Red Hat Advanced Cluster Management for Kubernetes 2.10

Clusters

集群管理

Last Updated: 2024-07-01

Red Hat Advanced Cluster Management for Kubernetes 2.10 Clusters

集群管理

法律通告

摘要

使用集群生命周期（也称为 multicluster engine operator），您可以创建和管理集群。在本指南中，您可以访问集群管理任务、发行注记和故障排除信息。

目录

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。

访问 支持列表，了解 hub 集群和受管集群要求和支持。

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。访问 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 安装和升级文档。
- 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
    - 安装和升级多集群引擎 Operator
    - 管理凭证
    - 集群生命周期简介
    - 发现服务简介
    - 托管 control plane
    - 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。

注： Red Hat Advanced Cluster Management 的 2.6 和更早的版本已 *从服务中删除*，且不再被支持。2.6 及更早的版本文档没有更新。其文档可能仍然可用，但不再有任何新的勘误或其他更新。

- 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. 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：

- 集群生命周期
- 凭证
- 托管 control plane
- Red Hat Advanced Cluster Management 集成

1.1.1.1. 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，您可以为集群代理附加组件配置代理设置，以允许受管集群通过 HTTP 和 HTTPS 代理服务器与 hub 集群通信。请参阅 为集群代理附加组件配置代理设置 以了解更多信息。
- **ManagedServiceAccount** 附加组件现在默认启用。如果要从 multicluster engine operator 版本 2.4 升级，请参阅启用 ManagedServiceAccount 附加组件。
- **技术预览：** 现在，您可以自定义服务器 URL 和 hub 集群 API CA 捆绑包，以便您可以在 multicluster engine operator hub 集群上导入受管集群，即使您有中间组件。请参阅 自定义服务器 URL 和 hub 集群 API CA 捆绑包（技术预览） 以了解更多信息。
- 现在，您可以在每个请求中传递 HTTP/HTTPS 标头和查询参数来获取操作系统镜像。如需更多信息，请参阅 在断开连接的环境中启用中央基础架构管理。
- 现在，您可以使用自签名或第三方 CA 证书存储并下载启用了 TLS 的 HTTPS **osImages** 进行验证。如需更多信息，请参阅 在断开连接的环境中启用中央基础架构管理。

1.1.1.2. 凭证

- 现在，您可以使用集成的控制台在 VMware vSphere 上为断开连接的安装配置 *Cluster OS image* 字段来创建凭证。如需更多信息，请参阅使用控制台管理凭证。

1.1.1.3. 托管 control plane

- **技术预览：** 您可以使用非裸机代理机器置备托管的 control plane 集群。如需更多信息，请参阅使用非裸机代理机器配置托管的 control plane 集群。

- 现在，您可以使用控制台创建带有 KubeVirt 平台的托管集群。如需更多信息，请参阅使用 控制台创建托管集群。
- 现在，您可以配置额外网络，对虚拟机(VM)请求保证 CPU 访问，并管理为节点池调度 KubeVirt 虚拟机。如需更多信息，请参阅为节点池配置额外网络、保证 CPU 和虚拟机调度。

1.1.1.4. Red Hat Advanced Cluster Management 集成

如果在安装 Red Hat Advanced Cluster Management 后启用 Observability，您可以使用 Grafana 仪表板查看托管的 control plane 集群容量估计以及现有的托管的 control plane 资源使用率。请参阅 Red Hat Advanced Cluster Management 集成。

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. *nmstate*的限制

通过配置复制和粘贴功能来加快开发速度。要在 **assisted-installer** 中配置 **copy-from-mac** 功能，您必须将 **mac-address** 添加到 **nmstate** 定义接口和 **mac-mapping** 接口。**mac-mapping** 接口在 **nmstate** 定义接口之外提供。因此，您必须提供相同的 **mac-address** 两次。

1.1.2.1.2. prehook 失败不会出现托管集群创建失败

如果您使用自动化模板来创建托管集群，并且 prehook 任务失败，则托管集群创建仍然正在进行。这是正常的，因为托管集群的设计没有完全失败状态，因此它会继续尝试创建集群。

1.1.2.1.3. 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。

1.1.2.1.4. 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>**。

1.1.2.1.5. 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。

1.1.2.1.6. 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**。

1.1.2.1.7. 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。

1.1.2.1.8. 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。

1.1.2.1.9. 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。

1.1.2.1.10. 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。

1.1.2.1.11. 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**。

1.1.2.1.12. 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。

1.1.2.1.13. 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。

1.1.2.1.14. 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。

1.1.2.1.15. 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。

1.1.2.1.16. 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。

1.1.2.1.17. 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。

1.1.2.1.18. 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。

1.1.2.1.19. 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***。

1.1.2.1.20. 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。

1.1.2.1.21. 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。

1.1.2.1.22. 升级后，非 Red Hat OpenShift Container Platform 受管集群需要*ManagedServiceAccount* 或 *LoadBalancer* 用于 pod 日志

如果您使用全新的 Red Hat Advanced Cluster Management 2.10 或更新版本安装，Red Hat OpenShift Container Platform 和非 OpenShift Container Platform 集群都支持 pod 日志功能。

如果从 Red Hat Advanced Cluster Management 2.9 升级到 2.10，您必须手动启用 **ManagedServiceAccount** 附加组件，以便在非 OpenShift Container Platform 受管集群中使用 pod 日志功能。请参阅 ManagedServiceAccount 附加组件 以了解如何启用 **ManagedServiceAccount**。

另外，您可以使用 **LoadBalancer** 而不是 **ManagedServiceAccount** 在非 OpenShift Container Platform 受管集群中启用 pod 日志功能。

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。

1.1.2.1.23. 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。

1.1.2.1.24. 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'
```

1.1.2.1.25. 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/4.12/latest/rhcos-live-kernel-x86_64
    rootfs: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.12/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。

1.1.2.1.26. 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
```

1.1.2.1.27. 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
spec:
  ignitionConfigOverride: '{"ignition":{"version":"3.2.0"},"storage":{"files":
[{"path":"/etc/containers/policy.json","mode":420,"overwrite":true,"contents":
{"source":"data:text/plain;charset=utf-
8;base64,ewogICAgImRlZmF1bHQiOiBbCiAgICAgICAgewogICAgICAgICAgICAidHlwZSI6ICJpbnNlY3
VyZUFjY2VwdEFueXRoaW5nIgogICAgICAgIH0KICAgIF0sCiAgICAidHJhbnNwb3J0cyI6CiAgICAgICAg
ewogICAgICAgICAgICAiZG9ja2VyLWRhZW1vbiI6CiAgICAgICAgICAgICAgICB7CiAgICAgICAgICAgICAg
ICAgICAgICAgICAiIjogWwogICAgICAgICAgICAgICAgICAgICAgICB7CiAgICAgICAgICAgICAgICAgICAg
ICAgICAgICAgICAgICAgICAidHlwZSI6IFt7InR5cGUiOiJpbnNlY3VyZUFjY2VwdEFueXRoaW5nIn1dCiAgICAgI
CB9CiAgICAgICAgICAgICAgICAgICAgICAgIF0KICAgICAgICAgICAgICAgIH0KICAgICAgICAgICAgfQogICAgI
H0K"}}]}}'
```

以下示例显示了创建的未限制文件：

```
{
    "default": [
        {
            "type": "insecureAcceptAnything"
        }
    ],
    "transports": {
        "docker-daemon": {
        "": [
        {
            "type": "insecureAcceptAnything"
        }
        ]
    }
    }
}
```

更改此设置后，集群可以安装。

1.1.2.1.28. 受管集群在部署后处于 *Pending* 状态

如果 Assisted Installer 代理启动缓慢，且部署受管集群，受管集群可能会处于 **Pending** 状态，且没有任何代理资源。您可以通过禁用聚合流来解决这个问题。完成以下步骤：

1. 在 hub 集群中创建以下 ConfigMap：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-assisted-service-config
  namespace: multicluster-engine
data:
  ALLOW_CONVERGED_FLOW: "false"
```

2. 运行以下命令来应用 ConfigMap：

```
oc annotate --overwrite AgentServiceConfig agent unsupported.agent-
install.openshift.io/assisted-service-configmap=my-assisted-service-config
```

1.1.2.1.29. ManagedClusterSet API 规格限制

使用 Clustersets API 时不支持 **selectorType: LaberSelector** 设置。支持 **selectorType: ExclusiveClusterSetLabel** 设置。

1.1.2.1.30. hub 集群通信限制

如果 hub 集群无法访问或与受管集群通信，则会出现以下限制：

- 您不能使用控制台创建新的受管集群。您仍然可以使用命令行界面或使用控制台中的 **手动运行 import 命令** 导入受管集群。
- 如果您使用控制台部署 Application 或 ApplicationSet，或者您将受管集群导入到 ArgoCD 中，hub 集群 ArgoCD 控制器会调用受管集群 API 服务器。您可以使用 AppSub 或 ArgoCD pull 模型来解决这个问题。
- pod 日志的控制台页面无法正常工作，并显示类似如下的错误消息：

```
Error querying resource logs:
Service unavailable
```

1.1.2.1.31. 受管服务帐户附加组件限制

以下是 **managed-serviceaccount** 附加组件的已知问题和限制：

1.1.2.1.31.1. *installNamespace* 字段只能有一个值

启用 **managed-serviceaccount** 附加组件时，**ManagedClusterAddOn** 资源中的 **installNamespace** 字段必须有 **open-cluster-management-agent-addon** 值。其他值将被忽略。**managed-serviceaccount** 附加组件代理总是在受管集群上的 **open-cluster-management-agent-addon** 命名空间中部署。

1.1.2.1.31.2. *tolerations* 和 *nodeSelector* 设置不会影响 *managed-serviceaccount* 代理

在 **MultiClusterEngine** 和 **MultiClusterHub** 资源中配置的 **tolerations** 和 **nodeSelector** 设置不会影响本地集群中部署的 **managed-serviceaccount** 代理。本地集群中并不总是需要 **managed-serviceaccount** 附加组件。

如果需要 **managed-serviceaccount** 附加组件，您可以通过完成以下步骤来临时解决这个问题：

1. 创建 **addonDeploymentConfig** 自定义资源。
2. 为本地集群和 **managed-serviceaccount** 代理设置 **tolerations** 和 **nodeSelector** 值。
3. 更新本地集群命名空间中的 **managed-serviceaccount ManagedClusterAddon**，以使用您创建的 **addonDeploymentConfig** 自定义资源。

请参阅为 klusterlet 附加组件配置 nodeSelectors 和 tolerations，以了解如何使用 **addonDeploymentConfig** 自定义资源为附加组件配置 **tolerations（容限）** 和 **nodeSelector**。

1.1.2.1.32. KubeVirt 托管集群上的批量销毁选项不会销毁托管集群

在 KubeVirt 托管集群的控制台中使用批量销毁选项不会破坏 KubeVirt 托管的集群。

使用行操作下拉菜单来销毁 KubeVirt 托管的集群。

1.1.2.1.33. Cluster curator 不支持 OpenShift Container Platform Dedicated 集群

当使用 **ClusterCurator** 资源升级 OpenShift Container Platform Dedicated 集群时，升级会失败，因为 Cluster curator 不支持 OpenShift Container Platform Dedicated 集群。

1.1.2.1.34. 自定义入口域无法正确应用

您可以在安装受管集群时使用 **ClusterDeployment** 资源指定自定义 ingress 域，但更改仅在使用 **SyncSet** 资源安装后才会生效。因此，**clusterdeployment.yaml** 文件中的 **spec** 字段显示您指定的自定义入口域，但 **status** 仍然会显示默认域。

1.1.2.2. 托管 control plane

1.1.2.2.1. 控制台将托管集群显示为 Pending import

如果注解和 **ManagedCluster** 名称不匹配，控制台会将集群显示为 **Pending import**。集群不能被 multicluster engine operator 使用。如果没有注解，而 **ManagedCluster** 名称与 **HostedCluster** 资源的 **Infra-ID** 值不匹配时会出现相同的问题。"

1.1.2.2.2. 当将节点池添加到托管集群时，控制台可能会多次列出同一版本

当使用控制台向现有托管集群添加新节点池时，同一版本的 OpenShift Container Platform 可能会在选项列表中出现多次。您可以在列表中为您想要的版本选择任何实例。

1.1.2.2.3. Web 控制台列出了节点，即使它们从集群中移除并返回基础架构环境

当节点池缩减为 0 个 worker 时，控制台中的主机列表仍然会显示处于 **Ready** 状态的节点。您可以通过两种方式验证节点数：

- 在控制台中，进入节点池，并验证它是否具有 0 个节点。
- 在命令行界面中运行以下命令：
  - 运行以下命令，验证 0 节点是否在节点池中：

    ```
    oc get nodepool -A
    ```

  - 运行以下命令，验证 0 个节点是否在集群中：

```
oc get nodes --kubeconfig
```

- 运行以下命令，验证 0 代理是否报告为绑定到集群：

```
oc get agents -A
```

1.1.2.2.4. 为双栈网络配置的托管集群中潜在的 DNS 问题

当您在使用双栈网络的环境中创建托管集群时，您可能会遇到以下与 DNS 相关的问题：

- **service-ca-operator** pod 中的 **CrashLoopBackOff** 状态：当 pod 试图通过托管的 control plane 访问 Kubernetes API 服务器时，pod 无法访问服务器，因为 **kube-system** 命名空间中的 data plane 代理无法解析请求。出现这个问题的原因是，前端使用 IP 地址，后端使用 pod 无法解析的 DNS 名称。

- Pod 处于 **ContainerCreating** 状态 ：出现这个问题，因为 **openshift-service-ca-operator** 无法生成 DNS pod 需要 DNS 解析的 **metrics-tls** secret。因此，pod 无法解析 Kubernetes API 服务器。

要解决这个问题，请按照 为双堆栈网络配置 DNS 中的指南来配置 DNS 服务器设置。

1.1.2.2.5. 在裸机平台上，代理资源可能无法拉取 ignition

在裸机(Agent)平台上，托管的 control plane 功能会定期轮转 Agent 用来拉取 ignition 的令牌。错误会导致不会传播新令牌。因此，如果您有一个在一些时间前创建的 Agent 资源，则可能无法拉取 ignition。

作为临时解决方案，在 Agent 规格中，删除 **IgnitionEndpointTokenReference** 属性引用的 secret，然后在 Agent 资源中添加或修改任何标签。然后，系统可以检测 Agent 资源是否已修改，并使用新令牌重新创建 secret。

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。

1.1.3.1. Errata 2.5.3

- 在 KubeVirt creation 向导中添加了一个字段，将托管 control plane 集群的默认模式设置为 **HighAvailability** 模式。(ACM-10580)

- 为一个或多个产品容器镜像提供更新。

1.1.3.2. Errata 2.5.2

- 修复了在运行备份恢复场景并为 Red Hat OpenShift Data Foundation (ODF)使用 Regional-DR 解决方案时可能会导致数据丢失的问题。(ACM-10407)

- 为一个或多个产品容器镜像提供更新。

1.1.3.3. Errata 2.5.1

- 为一个或多个产品容器镜像提供更新。

1.1.4. 弃用和删除集群生命周期

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
ManagedServiceAccount	**v1alpha1** API 被升级到 **v1beta1**，因为 **v1alpha1** 已被弃用。	2.9	使用 **v1beta1**。	None

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. 启用

*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
集群生命周期	在 Red Hat Virtualization 上创建集群	2.9	无	无
集群生命周期	klusterlet OLM Operator	2.4	无	无

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。

请参阅 支持列表 以了解 hub 集群和受管集群要求和支持。如需支持信息，以及以下文档：

- 控制台概述
- 用于 Kubernetes operator 基于角色的访问控制的多集群引擎
- 网络配置

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. 控制台概述

OpenShift Container Platform 控制台插件包括在 OpenShift Container Platform 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： 在启用插件后，您可以通过从下拉菜单中选择 **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。

1.2.2. 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

1.2.2.1. 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：

1.2.2.1.1. 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** 资源的管理员访问权限，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。

1.2.2.2. 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。

1.2.2.2.1. 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** 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

1.2.2.2.2. 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
klusterletaddonconfigs.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

1.2.2.2.3. 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。

1.2.3. 网络配置

将您的网络设置配置为允许连接。

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。

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

方向	协议	连接	端口（如果指定）
出站		置备的受管集群的 Kubernetes API 服务器	6443
从 OpenShift Container Platform 受管集群到 hub 集群的外向流量	TCP	hub 集群上的 ironic 代理和裸机 Operator 之间的通信	6180、6183、6385 和 5050
从 hub 集群到受管集群的 Ironic Python 代理 (IPA)的外向流量	TCP	运行 IPA 的裸机节点与 Ironic 编排器服务之间的通信	9999
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 operator 集群的多集群引擎的 Kubernetes API 服务器	6443

注：受管集群必须能够访问 hub 集群 control plane 节点 IP 地址。

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。

访问 支持列表，了解 hub 集群和受管集群要求和支持。

重要：如果您使用 Red Hat Advanced Cluster Management 版本 2.5 或更高版本，则集群中已安装了 Kubernetes operator 的多集群引擎。

请参阅以下文档：

- 在 断开连接的网络上安装在线.0.2 安装
- 卸装
- MultiClusterEngine 高级配置
- Red Hat Advanced Cluster Management 集成

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。

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。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 operator 的多集群引擎创建的 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 operator 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 4.13 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 4.13
- 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: {}
```

6. 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

1. 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 operator 命名空间的多集群引擎的名称。在 OpenShift Container Platform 环境中，**namespace** 的值可能被称为 Project（项目）。

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

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

```
oc project <namespace>
```

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

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
```

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

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

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

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

```
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: multicluster-engine
spec:
  sourceNamespace: openshift-marketplace
  source: redhat-operators
  channel: stable-2.1
  installPlanApproval: Automatic
  name: multicluster-engine
```

注： 要在基础架构节点上安装 Kubernetes operator 的多集群引擎，请参阅 Operator Lifecycle Manager 订阅额外配置 部分。

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

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

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

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

注： 要在基础架构节点上安装多集群引擎 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 订阅和 **MultiClusterEngine** 自定义资源。

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 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 4.13 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 4.13 或更高版本。

  请参阅 OpenShift Container Platform 版本 4.13 。
- 您的 Red Hat OpenShift Container Platform CLI 需要版本 4.13 或更高版本，并配置为运行 **oc** 命令。
- 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 4.13 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，您可能会在部署过程中收到 **ImagePullBackOff** 错误。

按照以下步骤在断开连接的环境中安装多集群引擎 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 operator 的多集群引擎可以访问 registry。

   a. 复制 registry 中的证书信息。

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

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

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

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

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

   - Container Security Operator 策略：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/gatekeeper** 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。*

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

如需了解所需步骤，请参阅在线安装。

1.3.3. 高级配置

multicluster engine operator 使用部署所有所需组件的 operator 安装。multicluster engine operator 可以在安装过程中或安装后进一步配置。了解有关高级配置选项的更多信息。

1.3.3.1. 部署的组件

*在 **MultiClusterEngine** 自定义资源中添加一个或多个属性：*

表 1.3. 部署的组件表列表

Name	描述	Enabled
assisted-service	使用最小基础架构先决条件和全面的 pre-flight 验证安装 OpenShift Container Platform	True
cluster-lifecycle	为 OpenShift Container Platform 和 Kubernetes hub 集群提供集群管理功能	True

cluster-manager	在集群环境中管理各种与集群相关的操作	True
cluster-proxy-addon	使用反向代理服务器在 hub 和受管集群中自动安装 **apiserver-network-proxy**	True
console-mce	启用 multicluster engine operator 控制台插件	True
discovery	在 OpenShift Cluster Manager 中发现并标识新集群	True
hive	置备并执行 OpenShift Container Platform 集群的初始配置	True
hypershift	以成本和时间效率以及跨云可移植性以大规模托管 OpenShift Container Platform control plane	True
hypershift-local-hosting	为本地集群环境启用本地托管功能	True
local-cluster	启用部署 multicluster engine operator 的本地 hub 集群的导入和自我管理	True
managedserviceacccount	将服务帐户同步到受管集群，并将令牌作为 secret 资源收集回 hub 集群	False
server-foundation	为多集群环境中的服务器端操作提供基础服务	True

当您将 multicluster engine operator 安装到集群中时，不是所有列出的组件都默认启用。

*您可以通过在 **MultiClusterEngine** 自定义资源中添加一个或多个属性来进一步配置多集群引擎 Operator。继续阅读以了解有关您可以添加属性的信息。*

1.3.3.2. 控制台和组件配置

*以下示例显示了可用于启用或禁用组件的 **spec.overrides** 默认模板：*

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: <name> 1
      enabled: true
```

1. 使用组件的名称替换 **name**。

或者，您可以运行以下命令。将 **namespace** 替换为项目的名称，将 **name** 替换为组件的名称：

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

1.3.3.3. 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**。

1.3.3.4. 自定义镜像 pull secret

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

OpenShift Container Platform 集群的 secret 要求由 Kubernetes operator 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>
```

1.3.3.5. 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>
```

1.3.3.6. 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。

重要： 如果您在单节点 OpenShift 集群中使用 multicluster engine operator，请将 **spec.availabilityConfig** 设置为 **Basic**。

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"
```

1.3.3.7. 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: ""
```

1.3.3.8. 容限（tolerations）

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

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

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

1.3.3.9. ManagedServiceAccount 附加组件

ManagedServiceAccount 附加组件允许您在受管集群中创建或删除服务帐户。要在启用此附加组件的环境中安装，请在 **spec.overrides** 的 **MultiClusterEngine** 规格中包括以下内容：

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

在创建 **MultiClusterEngine** 后，可以在命令行中编辑资源并将 **managedserviceaccount** 组件设置为 **enabled: true** 来启用 **ManagedServiceAccount** 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":"managedserviceaccount","enabled":true}}]'
```

1.3.4. 卸装

卸载 Kubernetes operator 的多集群引擎时，您会看到两个不同的过程级别：删除自定义资源 和完整的 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。

1.3.4.1. 先决条件：分离启用的服务

在卸载 Kubernetes operator 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，方法是在 集群创建 中选择您的供应商的信息。

1.3.4.2. 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。

1.3.4.3. 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 operator 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。

1.3.4.4. 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
```

如需更多信息，请参阅 断开连接的安装镜像。

1.3.5. Red Hat Advanced Cluster Management 集成

如果您在启用了 Red Hat Advanced Cluster Management 中使用 multicluster engine operator，您可以利用进一步的多集群管理功能。

1.3.5.1. 先决条件

请参阅以下先决条件与 Red Hat Advanced Cluster Management 功能集成。

- 您需要安装 Red Hat Advanced Cluster Management。请参阅 安装和升级 以安装。

1.3.5.2. Observability 集成

启用 **multicluster-observability** pod 后，您可以使用 Red Hat Advanced Cluster Management Observability Grafana 仪表板查看有关托管 control plane 的以下信息：

- **ACM** > **Hosted Control Planes Overview** 仪表板，以查看托管 control plane 的集群容量估算、相关集群资源以及现有托管 control plane 的列表和状态。

- **ACM** > **Resources** > **Hosted Control Plane** 仪表板，您可以从 Overview 页面中访问，以查看所选托管的 control plane 的资源使用。

要启用，请参阅 Observability 服务简介。

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。凭据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据，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。凭据存储为 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。请参阅了解并获取您的安全凭证。
- 允许在 AWS 上安装集群的帐户权限。有关如何配置 AWS 帐户 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。请参阅 登录到 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。
4. 输入 Red Hat OpenShift pull secret。请参阅下载 Red Hat OpenShift pull secret 以下载您的 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。

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。

1.4.1.2.1. 创建 S3 secret

要创建 Amazon Simple Storage Service (S3) secret，请在控制台中完成以下任务：

1. 点 **Add credential** > **AWS** > **S3 Bucket**。如果您点 **For Hosted Control Plane**，则会提供名称和命名空间。
2. 输入提供的以下字段信息：
   - **bucket 名称** ：添加 S3 存储桶的名称。
   - **aws_access_key_id**: 为您的 AWS 帐户添加 AWS 访问密钥 ID。登录到 AWS 以查找您的 ID。
   - **aws_secret_access_key**: 为您的新 AWS Secret 访问密钥提供内容。
   - **区域** ：输入您的 AWS 区域。

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。*
- *在凭证中添加标签以在控制台中查看您的 secret。例如，以下 AWS S3 Bucket* ***oc label secret*** *附加到* ***type=awss3*** *和* ***credentials --from-file=....***:

```
oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster "cluster.open-cluster-management.io/type=awss3"
oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster "cluster.open-cluster-management.io/credentials=credentials="
```

1.4.1.4. 其他资源

- *请参阅了解和获取您的安全凭证。*
- *请参阅配置 AWS 帐户。*
- *登录到 AWS。*
- *下载 Red Hat OpenShift pull secret 。*
- *请参阅生成 SSH 私钥并将其添加到代理中，以了解有关如何生成密钥的更多信息。*
- *请参阅在 Amazon Web Services 上创建集群*
- *请参阅在 Amazon Web Services GovCloud 上创建集群 。*
- *返回到为 Amazon Web Services 创建凭证 。*

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 Microsoft Azure 门户以获取您的登录凭证。
- 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 **appId** 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。
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。

9. 输入您的 Red Hat OpenShift pull secret 。请参阅下载 Red Hat OpenShift pull secret 以下载您的 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。

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。

1.4.2.4. 其他资源

- 请参阅 Microsoft Azure Portal。
- 请参阅如何配置云服务。
- 请参阅配置 Azure 帐户。
- 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。
- 下载 Red Hat OpenShift pull secret 。
- 请参阅生成 SSH 私钥并将其添加到代理中，以了解有关如何生成密钥的更多信息。

- 请参阅在 Microsoft Azure 上创建集群
- 返回到为 Microsoft Azure 创建凭证。

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 。请参阅登录到 GCP 以检索您的设置。
2. 添加 Google Cloud Platform 服务帐户 JSON 密钥 。请参阅 Create service account 文档，以创建您的服务帐户 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 Red Hat OpenShift pull secret。请参阅下载 Red Hat OpenShift pull secret 以下载您的 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。

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。

1.4.3.4. 其他资源

- 请参阅创建和管理项目。
- 请参阅配置 GCP 项目。
- 登录到 GCP。
- 请参阅创建服务账户来创建服务帐户 JSON 密钥。
- 下载 Red Hat OpenShift pull secret 。
- 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。
- 请参阅在 Google Cloud Platform 上创建集群

返回到 为 Google Cloud Platform 创建凭证。

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。

需要的访问权限： Edit

备注：

- 您必须先为 VMware vSphere 创建凭证，然后才能使用 multicluster engine operator 创建集群。
- OpenShift Container Platform 版本 4.13 及更新的版本被支持。

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 4.13 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： 通过运行 **cat certs/lin/*.0 > ca.crt** 命令，将多个带有 **.0** 扩展的证书链接到一起。

d. 添加 VMware vSphere 集群名称。

e. 添加 VMware vSphere 数据中心。

f. 添加 VMware vSphere 默认数据存储。

g. 添加 VMware vSphere 磁盘类型。

h. 添加 VMware vSphere 文件夹。

i. 添加 VMware vSphere 资源池。

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。请参阅下载 Red Hat OpenShift pull secret 以下载您的 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。

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。

1.4.4.4. 其他资源

- 请参阅使用自定义在 vSphere 上安装集群。
- 下载 Red Hat OpenShift pull secret 。
- 如需更多信息，请参阅为集群节点 SSH 访问生成密钥对。
- 请参阅在 VMware vSphere 上创建集群。
- 返回到 为 VMware vSphere 创建凭证。

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。

备注：

- *您必须先为 Red Hat OpenStack Platform 创建凭证，然后才能使用 multicluster engine operator 创建集群。*
- *仅支持 OpenShift Container Platform 版本 4.13 及更新的版本。*

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 4.13 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。要增强安全性和方便性，您可以创建一个命名空间，专门用于托管凭证。*

1. ***可选：**您可以为凭证添加基本 DNS 域。如果您添加了基本 DNS 域，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. *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，它是一个紧接在 **username** 后面的一个新行。*
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. ***可选** ：对于使用内部证书颁发机构的配置，请在 **Internal CA certificate** 字段中输入您的证书，以使用证书信息自动更新 **clouds.yaml**。*
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。
7. 输入 Red Hat OpenShift pull secret。请参阅下载 Red Hat OpenShift pull secret 以下载您的 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。
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。*

1.4.5.4. 其他资源

- *请参阅使用自定义在 OpenStack 上安装集群。*
- *下载 Red Hat OpenShift pull secret 。*
- *如需更多信息，请参阅为集群节点 SSH 访问生成密钥对。*
- *请参阅在 Red Hat OpenStack Platform 上创建集群。*
- *返回到 为 Red Hat OpenStack 创建凭证。*

1.4.6. 为 Red Hat Virtualization 创建凭证

***弃用：** Red Hat Virtualization 凭证和集群创建功能已弃用，并不再被支持。*

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 4.13 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。添加一个星号 * 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（已弃用）。

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。创建集群时会自动输入此 pull secret 并指定此凭证。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. 输入您的 **SSH 公钥**。创建集群时，也会自动输入此 **SSH 公钥** 并指定此凭证。
5. 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，它增强了集群集管理，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*
- *集群创建*
  - *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（已弃用）*
  - *在内部环境中创建集群*
  - *在代理环境中创建集群*
- *集群导入*
  - *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*
- *休眠创建的集群*
- *升级集群*

- 升级断开连接的集群
- 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
  - 使用污点和容限放置受管集群
  - 从 ManagedClusterSet 中删除受管集群
- Placement
- 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

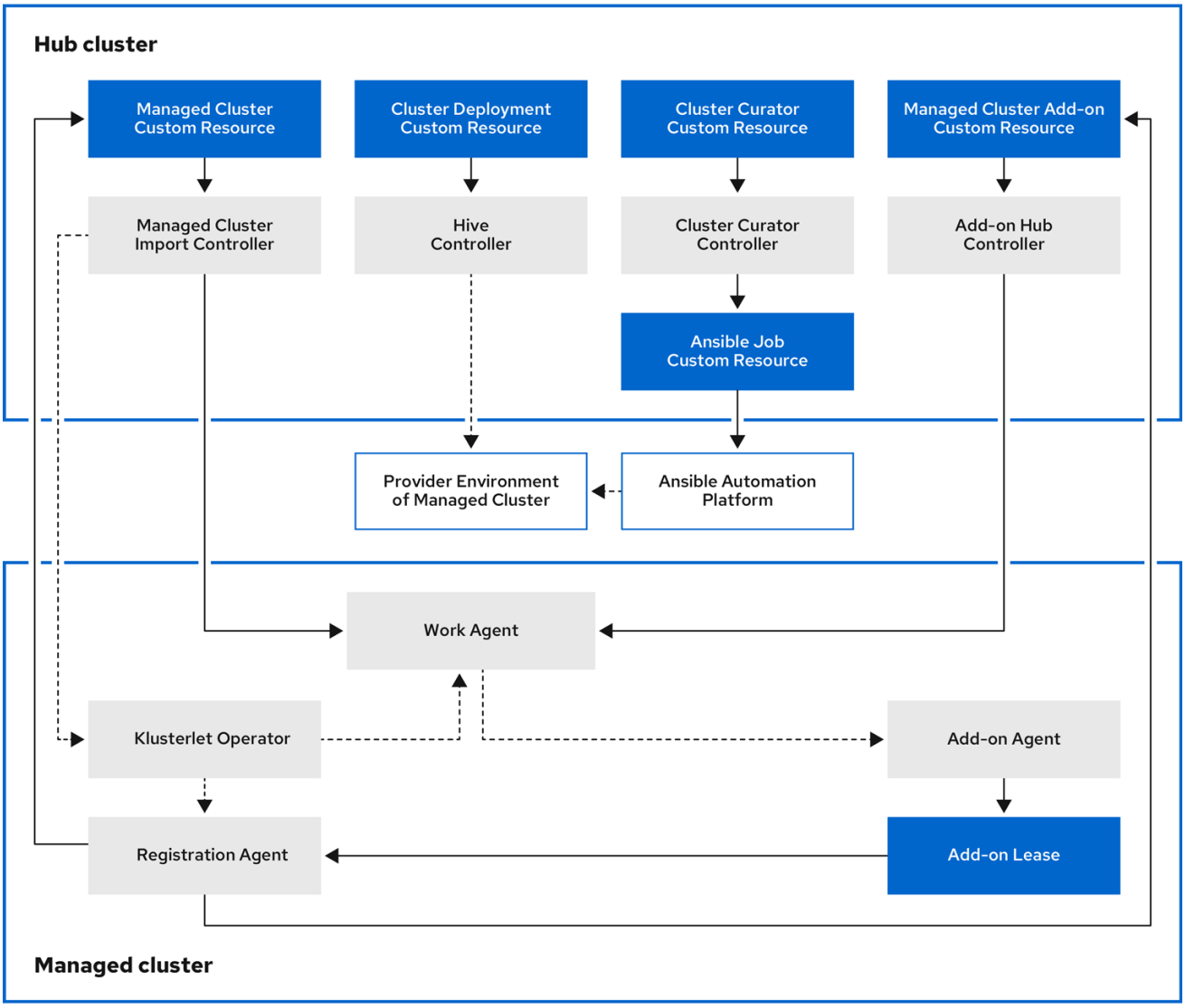
1.5.1. 集群生命周期架构

集群生命周期需要两种类型的集群：hub 集群和受管集群。

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：



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 **get** 或 **update coordination.k8s.io** 租期
  - 允许代理 **get** 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

构建集群时，请使用发行镜像指定的 Red Hat OpenShift Container Platform 版本。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。

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 for Kubernetes operator 在供应商处创建集群时，请指定用于新集群的发行镜像。要指定发行镜像，请参阅以下主题：

- 查找 ClusterImageSets
- 配置 ClusterImageSets
- 创建发行镜像以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 YAML 文件在 acm-hive-openshift-releases GitHub 仓库中维护。文件用于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:4.13.1-x86_64

控制台会显示最新版本，可帮助您使用最新发行镜像创建集群。如果您需要创建特定版本的集群，则也提供旧的发行镜像版本。

注： 您只能在控制台中创建集群时选择具有 **visible: 'true'** 标签的镜像。**ClusterImageSet** 资源中的此标签示例在以下内容中提供。将 **4.x.1** 替换为产品的当前版本：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.x.1-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.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，请运行以下命令：

```
oc get clusterimageset
```

存储库具有 **clusterImageSets** 目录，这是使用发行镜像时使用的目录。**clusterImageSets** 目录有以下目录：

- fast ：包含引用每个支持的 OpenShift Container Platform 版本最新版本的发行镜像的文件。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(stable、fast 和 candidate 频道)的所有发行镜像的文件
  注意：这些发行版本尚未全部经过测试并确定是稳定的。
- stable ：包含引用每个支持的 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，当前发行镜像列表每小时更新一次。升级产品后，列表最多可能需要一小时才能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：要在控制台中创建集群，请指定您想要我们的特定 **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，请参阅以下示例条目：

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

- 选项2：手动创建，应用来自 **acm-hive-openshift-releases** GitHub 仓库的 **ClusterImageSets** YAML 文件。
- 选项3：要从 fork 的 GitHub 仓库启用 **ClusterImageSets** 的自动更新，请遵循 cluster-image-set-controller GitHub 仓库中的 **README.md**。

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具有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 4.13 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。使用当前版本替换 **4.x.0** ：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.x.0-multi-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.0-multi
```

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

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. 运行以下命令，从 Quay 存储库拉取 环境中这两个架构的清单列表。将 **4.x.1** 替换为产品的当前版本：

   ```
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-x86_64
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-ppc64le
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-s390x
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-aarch64
   ```

   b. 运行以下命令，登录维护镜像的私有存储库。将 **<private-repo** > 替换为存储库的路径：

   ```
   podman login <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。将 **4.x.1** 替换为产品的当前版本。将 **<private-repo** > 替换为存储库的路径：

   ```
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-x86_64 <private-
   repo>/ocp-release:4.x.1-x86_64
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-ppc64le <private-
   repo>/ocp-release:4.x.1-ppc64le
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-s390x <private-
   repo>/ocp-release:4.x.1-s390x
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-aarch64 <private-
   repo>/ocp-release:4.x.1-aarch64
   ```

   d. 运行以下命令，为新信息创建清单：

   ```
   podman manifest create mymanifest
   ```

   e. 运行以下命令，将两个发行镜像的引用添加到清单列表中。将 **4.x.1** 替换为产品的当前版本。将 **<private-repo** > 替换为存储库的路径：

   ```
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-x86_64
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-ppc64le
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-s390x
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-aarch64
   ```

f. 运行以下命令，将清单列表中的列表与现有清单合并。将 **<private-repo>** 替换为存储库的路径。将 **4.x.1** 替换为当前版本：

```
podman manifest push mymanifest docker://<private-repo>/ocp-release:4.x.1
```

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。将 **<private-repo>** 替换为存储库的路径。将 **4.x.1** 替换为当前版本：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: "true"
  name: img4.x.1-appsub
spec:
  releaseImage: <private-repo>/ocp-release:4.x.1
```

b. 在 hub 集群中运行以下命令以应用更改。将 &lt **;file-name>** 替换为您在上一步中创建的 YAML 文件的名称：

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

3. 在创建 OpenShift Container Platform 集群时选择新的发行镜像。

4. 如果您使用 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。

1.5.2.1.4. 其他资源

- 有关引用发行镜像的 YAML 文件，请参阅 acm-hive-openshift-releases GitHub 仓库。
- 请参阅 cluster-image-set-controller GitHub 仓库 GitHub 存储库，以了解如何从 fork 的 GitHub 仓库启用 **ClusterImageSets** 的自动更新。

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 进行分叉。

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，但将 **2.x** 替换为 2.5：

```
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.x>
  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。

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: img4.13.8-x86-64-appsub
spec:
  releaseImage: IMAGE_REGISTRY_IPADDRESS_or_DNSNAME/REPO_PATH/ocp-release:4.13.8-x86_64
```

确保在 YAML 文件中引用的离线镜像 registry 中载入镜像。

注： 如果您要为 Red Hat OpenShift Virtualization 创建托管集群，则 **spec.releaseImage** 名称必须以 **-release** 结尾，才能使 OpenShift Container Platform 版本出现在控制台 **的发行镜像** 下拉列表中。

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。

注： Red Hat Advanced Cluster Management 2.9 及更新的版本，以及中央基础架构管理支持使用 **AgentClusterInstall** 支持 Nutanix 平台，这需要通过创建 Nutanix 虚拟机来提供额外的配置。如需更多信息，请参阅辅助安装程序文档中的 可选：在 Nutanix 上安装。

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 4.13 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"
```

注： 您必须将 **mirror-by-digest-only** 设置为 **true**，因为发行镜像是使用摘要指定的。

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. 编写代表您要随每个 **osImage** 请求发送的标头和查询参数的密钥对。如果您不需要这两个参数，则只对标头或查询参数写入密钥对。

重要： 标头和查询参数仅在使用 HTTPS 时加密。确保使用 HTTPS 来避免安全问题。

a. 创建名为 **headers** 的文件并添加类似以下示例的内容：

```
{
  "header1": "header1value",
  "header2": "header2value",
}
```

b. 创建名为 **query_params** 的文件并添加类似以下示例的内容：

```
{
  "param1": "value1",
  "param2": "value2",
}
```

1. 运行以下命令，从您创建的参数文件创建 secret。如果您只创建一个参数文件，请删除您没有创建的文件的参数：

```
oc create secret generic -n multicluster-engine os-images-http-auth --from-file=./query_params --from-file=./headers
```

2. 如果要将 HTTPS **osImages** 与自签名或第三方 CA 证书搭配使用，请将证书添加到 **image-service-additional-ca ConfigMap** 中。要创建证书，请运行以下命令：

```
oc -n multicluster-engine create configmap image-service-additional-ca --from-file=tls.crt
```

3. 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> (1)
  unauthenticatedRegistries:
    - <unauthenticated_registry> (2)
  imageStorage:
```

```
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: <img_volume_size> 3
OSImageAdditionalParamsRef:
  name: os-images-http-auth
OSImageCACertRef:
  name: image-service-additional-ca
osImages:
  - openshiftVersion: "<ocp_version>" 4
    version: "<ocp_release_version>" 5
    url: "<iso_url>" 6
    cpuArchitecture: "x86_64"
```

1 1 将 **mirror_config** 替换为包含您的镜像 registry 配置详情的 **ConfigMap** 名称。

2 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。

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。

4 将 **ocp_version** 替换为要安装的 OpenShift Container Platform 版本，如 **4.13**。

5 将 **ocp_release_version** 替换为特定的安装版本，例如：**49.83.202103251640-0**。

6 使用 ISO url 替换 **iso_url**，例如 **https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.13/4.13.3/rhcos-4.13.3-x86_64-live.x86_64.iso**。您可以在 *4.12.3 依赖项* 中找到其他值。

如果您使用带有自签名或第三方 CA 证书的 HTTPS **osImages**，请在 **OSImageCACertRef** spec 中引用证书。

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 **2-3G**。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。

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

```
metadata:
```

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

8. 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
```

9. 将 URL 中的 **apps** 替换为 **nlb-apps**，以匹配新 **IngressController** 中配置的域。

10. 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
```

11. 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：

1. 在控制台中，导航到 **Infrastructure** > **Host inventory**，再点 **Create infrastructure environment**。

2. 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
- 请参阅使用命令行界面创建主机清单
- 如果您完成此流程作为在裸机上配置托管 control plane 的过程的一部分，则后续步骤是完成以下步骤：
  - 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

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sapce。

   (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 1.4. 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。

字段	可选或必需的	描述
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。
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。

字段	可选或必需的	描述
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。
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。

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 1.5. 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。

字段	可选或必需的	描述
some-key	必需	在 **InfraEnv** 资源中添加一个或多个与 **nmStateConfigLabelSelector** 匹配的标签。
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 *Declarative Network API*。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> 1
type: Opaque
data:
  username: <username>
  password: <password>
```

1 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> 1
  annotations:
    inspect.metal3.io: disabled
  labels:
    infraenvs.agent-install.openshift.io: <your-infraenv> 2
spec:
  online: true
  automatedCleaningMode: disabled 3
  bootMACAddress: <your-mac-address> 4
  bmc:
    address: <machine-address> 5
    credentialsName: <bmc-secret-name> 6
  rootDeviceHints:
    deviceName: /dev/sda 7
```

-

1. 命名空间必须与 **InfraEnv** 的命名空间相同。
2. 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. 使用命令行界面删除受管集群节点

要从受管集群中删除受管集群，需要一个运行 OpenShift Container Platform 版本 4.13 或更高版本的 hub 集群。节点启动所需的任何静态网络配置都必须可用。在删除代理和裸机主机时，确保不会删除 **NMStateConfig** 资源。

1.5.3.7.4.1. 使用裸机主机删除受管集群节点

如果您在 hub 集群中有一个裸机主机，并希望从受管集群中删除受管集群，请完成以下步骤：

1. 在您要删除的节点的 **BareMetalHost** 资源中添加以下注解：

```
bmac.agent-install.openshift.io/remove-agent-and-node-on-delete: true
```

2. 运行以下命令来删除 **BareMetalHost** 资源。将 **<bmh-name>** 替换为 **BareMetalHost** 的名称：

```
oc delete bmh <bmh-name>
```

1.5.3.7.4.2. 在没有裸机主机的情况下删除受管集群节点

如果您在 hub 集群中没有裸机主机，并希望从受管集群中删除受管集群，请按照 OpenShift Container Platform 文档中的 Deleting 节点 说明操作。

1.5.3.7.5. 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*
- *要了解更多有关扩展计算机器的信息，请参阅 OpenShift Container Platform 文档中的手动扩展计算机器集。https://docs.openshift.com/container-platform/4.13/machine_management/manually-scaling-machineset.html*

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***
  *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,aGVscGltdHJhcHBlZGluYXN3YWdnZXJzcGVj"}}]}}
  ```

- ***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 resouce 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. 集群创建

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（已弃用）
- 在内部环境中创建集群
- 托管 control plane

1.5.4.1. 使用 CLI 创建集群

Kubernetes operator 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，但将 2.x 替换为 2.5：

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/backplane-<2.x>
```

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。

注：如果您使用 Nutanix 平台，请确保将 ClusterImageSet 资源中的 releaseImage 使用 x86_64 架构，并将 可见的 标签值设置为 'true'。请参见以下示例：

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: stable
    visible: 'true'
  name: img4.x.47-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.47-x86_64
```

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
BKWjJRzeUVuZs4yxSy4eu45xiANFIlbwE3e1aPzGD58x/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。如果您有一个要导入的现有集群，请参阅集群 导入。

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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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 VPC 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 multicluster 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，请参阅集群 导入 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，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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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，请参阅集群 导入 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 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，请参阅为 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。有关发行镜像的更多信息，请参阅发行镜像。

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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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，请参阅集群 导入 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 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。有关发行镜像的更多信息，请参阅发行镜像。

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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群

klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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 4.13 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：
  - 以下 API 基域必须指向静态 API VIP：

```
api.<cluster_name>.<base_domain>
```

  - 以下应用程序基域必须指向 Ingress VIP 的静态 IP 地址：

```
*.apps.<cluster_name>.<base_domain>
```

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，请参阅集群 导入 以了解这些步骤。

如果您需要创建凭证，请参阅为 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。有关发行镜像的更多信息，请参阅发行镜像。

注：支持 OpenShift Container Platform 版本 4.13 及更新的版本的发行镜像。

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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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，请参阅为 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
  - 外部网络的网络名称，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：
  - 以下 API 基域必须指向 API 的浮动 IP 地址：

```
api.<cluster_name>.<base_domain>
```

  - 以下应用程序基域必须指向 ingress:app-name 的浮动 IP 地址：

```
*.apps.<cluster_name>.<base_domain>
```

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，请参阅集群 导入 以了解这些步骤。

如果您需要创建凭证，请参阅为 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 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。有关发行镜像的更多信息，请参阅发行镜像。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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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（已弃用）

弃用：Red Hat Virtualization 凭证和集群创建功能已弃用，并不再被支持。

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（已弃用）：

- 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 Pull secret 页面下载 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。

- 您需要以下 DNS 记录：
  - 以下 API 基域必须指向静态 API VIP：

```
api.<cluster_name>.<base_domain>
```

  - 以下应用程序基域必须指向 Ingress VIP 的静态 IP 地址：

```
*.apps.<cluster_name>.<base_domain>
```

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，请参阅集群 导入 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。有关发行镜像的更多信息，请参阅发行镜像。

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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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。集群可以是单节点 OpenShift 集群、多节点集群，以及 VMware vSphere、Red Hat OpenStack、Red Hat Virtualization Platform （已弃用）、Nutanix 或裸机环境中的紧凑三节点集群。

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。当您要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。

- 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 4.13 或更高版本上部署了 hub 集群。
- 您需要配置了主机主机清单的基础架构环境。
- 您必须对 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。请参阅使用镜像 pull secret。
- 您需要以下 DNS 记录：
  - 以下 API 基域必须指向静态 API VIP：

```
api.<cluster_name>.<base_domain>
```

- 以下应用程序基域必须指向 Ingress VIP 的静态 IP 地址：

```
*.apps.<cluster_name>.<base_domain>
```

1.5.4.10.2. 使用控制台创建集群

要从控制台创建集群，请完成以下步骤：

1. 进入 Infrastructure > Clusters。
2. 在 Clusters 页面上，点 Create cluster 并完成控制台中的步骤。
3. 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。

注： 选择 YAML: On 在控制台中输入信息时查看内容更新。

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。请参阅 发行镜像 以了解更多信息。

当您选择受支持的 OpenShift Container Platform 版本时，会显示 安装单节点 OpenShift 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。请参阅 将主机扩展到基础架构环境，以了解有关在创建节点后向单节点 OpenShift 集群添加节点的更多信息。

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，单节点 OpenShift 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。

如果您使用 Red Hat Advanced Cluster Management for Kubernetes，并希望将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

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' 1
```

1

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-v4.13.0
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.13.0-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) 使用 AgentClusterInstall 资源的名称。

(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-v4.13.0 (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, None, VSphere, Nutanix, 或 External.

2

使用与 ClusterDeployment 资源相同的名称。

3

使用在生成一个ClusterImageSet 中生成的 ClusterImageSet。

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: 0.0.0.0/0
        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"
```

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>
  nmStateConfigLabelSelector:
    matchLabels:
      demo-nmstate-label: value
  proxy:
    httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
    httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
    noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

(1) *替换 Create the ClusterDeployment 中的 clusterDeployment 资源名称。*

(2) *替换 Create the ClusterDeployment 中的 clusterDeployment 资源命名空间。*

1.5.4.10.3.7.1. InfraEnv 字段表

字段	可选或必需的	描述
sshAuthorizedKey	选填	您可以指定 SSH 公钥，这可让您在从发现 ISO 镜像引导时访问主机。

字段	可选或必需的	描述
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。
proxy	选填	您可以在 proxy 部分的发现期间指定主机所需的代理设置。

注：当使用 IPv6 置备时，您无法在 noProxy 设置中定义 CIDR 地址块。您必须单独定义每个地址。

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。

1.5.4.10.4. 其他资源

- 有关使用 CLI 在 Nutanix 平台创建集群时，请参阅 Red Hat OpenShift Container Platform 文档中的使用 API 和 Nutanix 安装后配置 在 Nutanix 中添加主机。
- 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
- 请参阅为内部环境创建凭证
- 请参阅 发行镜像
- 请参阅使用 Discovery 镜像将主机添加到主机清单中

1.5.4.11. 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，您必须使用策略。如需更多信息，请参阅 **Pod** 策略。

1.5.4.11.1. 其他资源

- 请参阅 集群创建 来选择您的供应商。
- 请参阅 Pod 策略以 了解如何在部署集群后进行配置更改。
- 如需了解更多主题，请参阅 集群生命周期简介。
- 返回到 在代理环境中创建集群。

1.5.5. 集群导入

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。通常，您可以在可以访问 hub 集群和目标受管集群的任何位置完成导入任务，除非另有指定。

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。

如果删除了 hub 集群并尝试再次导入它，您必须将 local-cluster:true 标签添加到 ManagedCluster 资源中。

阅读以下主题以了解有关导入集群的更多信息，以便您可以管理它：

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
- 使用代理注册导入受管集群

- 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
- 在 Red Hat OpenShift Dedicated 环境中手动运行导入命令
- 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 4.13 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合中，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。如果要在 OpenShift Container Platform Dedicated 环境中导入集群并运行导入命令，请在控制台中提供信息 前手动在 OpenShift Container Platform Dedicated 环境中运行导入命令的步骤。
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。

要将受管集群 klusterlet 配置为在特定节点上运行，请参阅 可选：将 klusterlet 配置为在特定节点上运行。

1.5.5.1.3.1. 在 OpenShift Container Platform Dedicated 环境中手动运行导入命令

注：Klusterlet OLM Operator 和以下步骤已弃用。

如果要在 OpenShift Container Platform Dedicated 环境中导入集群并手动运行导入命令，则必须完成一些额外的步骤。

1. ***登录到您要导入的集群的 OpenShift Container Platform 控制台。***
2. ***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。***
3. ***在 OpenShift Container Platform 目录中找到 klusterlet Operator。***
4. ***在 open-cluster-management 命名空间中安装 klusterlet Operator。***

   重要：不要在 open-cluster-management-agent 命名空间中安装 Operator。
5. ***通过完成以下步骤，从导入命令中提取 bootstrap secret：***

a. 将导入命令粘贴到您创建的名为 import-command 的文件中。

b. 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
```

c. 在输出中找到并复制名为 bootstrap-hub-kubeconfig 的 secret。

d. 将 secret 应用到受管集群上的 open-cluster-management-agent 命名空间。

e. 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。

6. 选择 Cluster > Import cluster 后在控制台中提供信息。

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，将 cluster-name 替换为受管集群的名称：

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

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> (1)
```

(1) 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。

1.5.5.1.3.3. 可选：将 klusterlet 配置为在特定节点上运行

您可以通过为受管集群配置 nodeSelector 和 tolerations 注解来指定您希望受管集群 klusterlet 运行哪些节点。完成以下步骤以配置这些设置：

1. 从控制台的集群页面中选择要更新的受管集群。

2. 将 YAML 开关设置为 On 以查看 YAML 内容。

   注： YAML 编辑器仅在导入或创建集群时可用。要在导入或创建后编辑受管集群 YAML 定义，您必须使用 OpenShift Container Platform 命令行界面或 Red Hat Advanced Cluster Management 搜索功能。

3. 将 nodeSelector 注解添加到受管集群 YAML 定义。此注解的密钥是：open-cluster-management/nodeSelector。此注解的值是带有 JSON 格式的字符串映射。

4. 将 tolerations 条目添加到受管集群 YAML 定义中。此注解的键为：open-cluster-management/tolerations。此注解的值代表带有 JSON 格式的 容限 列表。生成的 YAML 可能类似以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    open-cluster-management/nodeSelector: '{"dedicated":"acm"}'
    open-cluster-management/tolerations:
'[{"key":"dedicated","operator":"Equal","value":"acm","effect":"NoSchedule"}]'
```

您还可以使用 KlusterletConfig 为受管集群配置 nodeSelector 和 tolerations。完成以下步骤以配置这些设置：

注：如果您使用 KlusterletConfig，受管集群将使用 KlusterletConfig 设置中的配置，而不是受管集群注解中的设置。

1. *应用以下示例 YAML 内容。根据需要替换值：*

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: <klusterletconfigName>
spec:
  nodePlacement:
    nodeSelector:
      dedicated: acm
    tolerations:
      - key: dedicated
        operator: Equal
        value: acm
        effect: NoSchedule
```

2. *将 agent.open-cluster-management.io/klusterlet-config: '<klusterletconfigName& gt; 注解添加到受管集群，将 & lt;klusterletconfigName > 替换为 KlusterletConfig 的名称。*

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 集群来自己管理。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，则必须在 OpenShift Container Platform 版本 4.13 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 4.13 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& gt; 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.crds\\.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. 使用代理注册导入受管集群

为 Kubernetes operator 安装多集群引擎后，就可以导入集群并使用代理注册端点管理它。继续阅读以下主题以了解如何使用代理注册端点导入受管集群。

- 先决条件
- 支持的构架
- 导入集群

1.5.5.3.1. 先决条件

- 已部署 hub 集群。如果要导入裸机集群，则必须在 OpenShift Container Platform 版本 4.13 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。

  如果需要创建新 secret，请参阅 创建新的 pull secret。

1.5.5.3.2. 支持的构架

- *Linux (x86_64, s390x, ppc64le)*
- *macOS*

1.5.5.3.3. 导入集群

要使用代理注册端点导入受管集群，请完成以下步骤：

1. *在 hub 集群中运行以下命令来获取代理注册服务器 URL：*

```
export agent_registration_host=$(oc get route -n multicluster-engine agent-registration -o=jsonpath="{.spec.host}")
```

注：如果您的 hub 集群使用集群范围的proxy，请确保使用受管集群可访问的 URL。

2. *运行以下命令来获取 cacert ：*

```
oc get configmap -n kube-system kube-root-ca.crt -o=jsonpath="{.data['ca\.crt']}" > ca.crt_
```

3. *通过应用以下 YAML 内容来获取代理注册者的令牌：*

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: managed-cluster-import-agent-registration-sa
  namespace: multicluster-engine
---
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: managed-cluster-import-agent-registration-sa-token
  namespace: multicluster-engine
  annotations:
    kubernetes.io/service-account.name: "managed-cluster-import-agent-registration-sa"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: managedcluster-import-controller-agent-registration-client
```

```
rules:
- nonResourceURLs: ["/agent-registration/*"]
  verbs: ["get"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: managed-cluster-import-agent-registration
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: managedcluster-import-controller-agent-registration-client
subjects:
  - kind: ServiceAccount
    name: managed-cluster-import-agent-registration-sa
    namespace: multicluster-engine
```

4. 运行以下命令来导出令牌：

```
export token=$(oc get secret -n multicluster-engine managed-cluster-import-agent-registration-sa-token -o=jsonpath='{.data.token}' | base64 -d)
```

5. 运行以下命令，启用自动批准并将内容修补到 **cluster-manager** ：

```
oc patch clustermanager cluster-manager --type=merge -p '{"spec":
{"registrationConfiguration":{"featureGates":[
{"feature": "ManagedClusterAutoApproval", "mode": "Enable"}], "autoApproveUsers":
["system:serviceaccount:multicluster-engine:agent-registration-bootstrap"]}}}'
```

注： 您还可以禁用自动批准，并从受管集群手动批准证书签名请求。

6. 运行以下命令切换到受管集群并获取 **cacert** ：

```
curl --cacert ca.crt -H "Authorization: Bearer $token" https://$agent_registration_host/agent-registration/crds/v1 | oc apply -f -
```

7. 运行以下命令将受管集群导入到 **hub** 集群：

将 **<clusterName >** 替换为集群的名称。

可选：将 **<klusterletconfigName >** 替换为 **KlusterletConfig** 的名称。

```
curl --cacert ca.crt -H "Authorization: Bearer $token" https://$agent_registration_host/agent-registration/manifests/<clusterName>?klusterletconfig=<klusterletconfigName> | oc apply -f -
```

1.5.5.4. 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。在安装 multicluster engine operator 时，hub 集群会自动导入，因此您可以在不完成以下步骤的情况下将节点添加到 hub 集群中。继续阅读以下主题以了解更多信息：

- 先决条件
- 导入集群

1.5.5.4.1. 先决条件

- 启用中央基础架构管理服务。

1.5.5.4.2. 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@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
```

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
```

   注：如果您的集群 API 通过自定义域访问，您必须首先通过在 certificate-authority-data 字段中添加自定义证书，并通过更改 server 字段以匹配自定义域来编辑此 kubeconfig。

   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 1.6. InfraEnv 字段表

字段	可选或必需的	描述
clusterRef	选填	如果您使用晚绑定，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
   ```

2. ***可选： 如果要在 OpenShift Container Platform 集群中使用 Red Hat Advanced Cluster Management 功能，如策略，请创建一个 ManagedCluster 资源。确保 ManagedCluster 资源的名称与 ClusterDeplpoyment 资源的名称匹配。如果您缺少 ManagedCluster 资源，则控制台中的集群状态会 分离。***

1.5.5.5. 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> 1
  pullSecret:
    name: <pullSecretName> 2
  registries: 3
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
```

1 使用选择一组受管集群的同一命名空间中的放置名称替换。

2 使用用于从自定义镜像 registry 中拉取镜像的 pull secret 的名称替换。

3 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
```

重要： 如果要使用代理注册导入受管集群，您必须创建一个包含镜像 registry 的 KlusterletConfig。请参见以下示例。根据需要替换值：

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: <klusterletconfigName>
spec:
  pullSecret:
    namespace: <pullSecretNamespace>
    name: <pullSecretName>
  registries:
    - mirror: <mirrored-image-registry-address>
      source: <image-registry-address>
    - mirror: <mirrored-image-registry-address>
      source: <image-registry-address>
```

请参阅 使用代理注册端点导入受管集群 以了解更多信息。

1.5.5.5.1. 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. 在将 open-cluster-management.io/image-registry=myNamespace.myImageRegistry 标签添加到受管集群后，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。如果您使用安装程序置备的基础架构进行集群部署，请参阅以下选项：

- 使用 MachinePool 扩展

如果要对集群部署使用中央基础架构管理，请参阅以下选项：

- 在 OpenShift Container Platform 集群中添加 worker 节点
- 将 control plane 节点添加到受管集群

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。

1.5.7.1.4. 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. 如果代理注册成功，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。

1.5.7.2. 将 control plane 节点添加到受管集群

您可以通过将 control plane 节点添加到健康或不健康的受管集群来替换失败的 control plane。

需要的访问权限: Administrator

1.5.7.2.1. 将 control plane 节点添加到健康的受管集群

完成以下步骤，将 control plane 节点添加到健康的受管集群：

1. 为新的 control plane 节点添加 worker 节点到 OpenShift Container Platform 集群的步骤。
2. 运行以下命令，在批准代理前将代理设置为 master ：

```
oc patch agent <AGENT-NAME> -p '{"spec":{"role": "master"}}' --type=merge
```

注： CSR 不会被自动批准。

3. 按照 OpenShift Container Platform 的 Assisted Installer 文档中的在健康集群中安装主 control plane 节点 的步骤

1.5.7.2.2. 将 control plane 节点添加到不健康的受管集群

完成以下步骤，将 control plane 节点添加到不健康的受管集群：

1. 删除不健康的 control plane 节点的代理。

2. 如果您使用零对部署置备流，请删除裸机主机。

3. 为新的 control plane 节点添加 worker 节点到 OpenShift Container Platform 集群的步骤。

4. 运行以下命令，在批准代理前将代理设置为 master ：

```
oc patch agent <AGENT-NAME> -p '{"spec":{"role": "master"}}' --type=merge
```

注： CSR 不会被自动批准。

5. 按照 OpenShift Container Platform 的 Assisted Installer 文档中的在不健康集群中安装主 control plane 节点 的步骤

1.5.8. 休眠创建的集群

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

1.5.8.1. 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。

1.5.8.2. 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。

1.5.8.3. 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。

1.5.8.4. 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。

1.5.9. 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。

1.5.9.1. 选择一个频道

您可以使用控制台为 OpenShift Container Platform 上的集群升级选择频道。选择频道后，会自动提醒勘误版本和发行版本可用的集群升级。

要为集群选择频道，请完成以下步骤：

1. 在导航栏中，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。

1.5.9.2. 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：

- 监测何时有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

1.5.9.2.1. 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：

- 在 Red Hat OpenShift Container Platform 版本 4.13 或更高版本上运行的 hub 集群，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。

1.5.9.2.2. 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 (1)

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}
```

(1) 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。

1.5.9.2.3. 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。

1.5.9.2.4. 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) (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 graph data 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。

1.5.9.2.5. 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。

3. 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-----
```

4. 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。

1.5.9.2.6. 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：

1. 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：

   a. 在 OpenShift Container Platform hub 集群控制台导航菜单中，选择 Administration > Namespaces。

   b. 点 Create Namespace。

c. 添加命名空间的名称以及您的命名空间的任何其他信息。

d. 选择 Create 来创建命名空间。

2. 在 OpenShift Container Platform 控制台的 Installed Operators 部分中，选择 Red Hat OpenShift Update Service Operator。

3. 在菜单中选择 Create Instance。

4. 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。

1.5.9.2.7. 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
```

1.5.9.2.8. 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' 1
  displayName: My Operator Catalog
  publisher: grpc
```

1 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
```

1.5.9.2.9. 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 upstream 参数目前是默认公共 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-4.13
    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-4.13
    upstream: https://<hub-cincinnati-uri>/api/upgrades_info/v1/graph
```

1.5.9.2.10. 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。

1.5.9.2.11. 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。

1.5.9.2.12. 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。

1.5.10. 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： hub 集群中不能是集群范围的代理配置。

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"
   ```

1.5.10.1. 为集群代理附加组件配置代理设置

您可以为集群代理附加组件配置代理设置，以允许受管集群通过 **HTTP** 和 **HTTPS** 代理服务器与 **hub** 集群通信。如果集群代理附加组件代理需要通过代理服务器访问 **hub** 集群，您可能需要配置代理设置。

要为集群代理附加组件配置代理设置，请完成以下步骤：

1. 在 **hub** 集群上创建一个 **AddOnDeploymentConfig** 资源，并添加 **spec.proxyConfig** 参数。请参见以下示例：

   ```
   apiVersion: addon.open-cluster-management.io/v1alpha1
   kind: AddOnDeploymentConfig
   metadata:
   ```

```
  name: <name> (1)
  namespace: <namespace> (2)
spec:
  agentInstallNamespace: open-cluster-managment-addon-observability
  proxyConfig:
  httpsProxy: "http://<username>:<password>@<ip>:<port>" (3)
  noProxy: ".cluster.local,.svc,172.30.0.1" (4)
  caBundle: <value> (5)
```

(1) 添加附加组件部署配置名称。

(2) 添加受管集群名称。

(3) 指定 HTTP 代理或 HTTPS 代理。

(4) 添加 kube-apiserver 的 IP 地址。要获取 IP 地址，请在受管集群中运行以下命令：`oc -n default describe svc kubernetes | grep IP:`

(5) 如果您在 httpsProxy 字段中指定 HTTPS 代理，请设置代理服务器 CA 捆绑包。

2. 通过引用您创建的 AddOnDeploymentConfig 资源来更新 ManagedClusterAddOn 资源。请参见以下示例：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: cluster-proxy
  namespace: <namespace> (1)
spec:
installNamespace: open-cluster-managment-addon
configs:
  group: addon.open-cluster-management.io
  resource: AddonDeploymentConfig
  name: <name> (2)
  namespace: <namespace> (3)
```

(1)

2

添加附加组件部署配置名称。

3

添加受管集群名称。

1. 通过检查 open-cluster-managment-addon 命名空间中的集群代理 pod 是否有 HTTPS_PROXY 或 NO_PROXY 环境变量来验证代理设置。

1.5.11. 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

1.5.11.1. 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 4.13 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。

1.5.11.2. 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。

1.5.11.3. 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 菜单中点击 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。

5. 选择凭据后，您可以选择用于所有作业的 Ansible 清单。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 并完成为 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。

6. 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。

7. 在出现的模态中选择 Job template 或 Workflow job template。您还可以添加 job_tags、skip_tags 和工作流类型。

   - 使用 Extra variables 字段以 key=value 对的形式将数据传递给 AnsibleJob 资源。

   - 特殊键 cluster_deployment 和 install_config 会自动作为额外变量传递。它们包含有关集群的常规信息以及集群安装配置的详情。

8. 选择 prehook 和 posthook Ansible 作业的名称，以添加到集群的安装或升级中。

9. 如有必要，拖动 Ansible 作业以更改顺序。

10.

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。在升级集群时，您可以使用 Extra variables 字段以 key=value 对的形式将数据传递给 AnsibleJob 资源。除了 cluster_deployment 和 install_config 特殊密钥外，cluster_info 特殊键也会自动传递为包含 ManagedClusterInfo 资源数据的额外变量。

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。

1.5.11.4. 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。

1.5.11.5. 再次运行失败的 Ansible 作业

如果集群 prehook 或 posthook 失败，您可以从 Clusters 页面重试升级。

要节省时间，您还可以只运行属于集群自动化模板的失败 Ansible posthook。完成以下步骤，在不重试整个升级的情况下，只运行 posthook：

1. 在 ClusterCurator 资源的根目录中添加以下内容来再次运行安装 posthook：

```
operation:
  retryPosthook: installPosthook
```

2. 在 ClusterCurator 资源的根目录中添加以下内容来再次运行升级 posthook：

```
operation:
  retryPosthook: upgradePosthook
```

添加内容后，会创建一个新作业来运行 Ansible posthook。

1.5.11.6. 指定用于所有作业的 Ansible 清单

您可以使用 ClusterCurator 资源指定用于所有作业的 Ansible 清单。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: test-inno
  namespace: test-inno
spec:
  desiredCuration: upgrade
  destroy: {}
  install: {}
  scale: {}
  upgrade:
    channel: stable-4.13
    desiredUpdate: 4.13.1
    monitorTimeout: 150
    posthook:
    - extra_vars: {}
      clusterName: test-inno
      type: post_check
      name: ACM Upgrade Checks
    prehook:
    - extra_vars: {}
      clusterName: test-inno
      type: pre_check
      name: ACM Upgrade Checks
    towerAuthSecret: awx
```

要验证清单是否已创建，您可以检查 ClusterCurator 资源中的 status 字段，以了解指定所有作业是否已成功完成的消息。

1.5.12. 配置 Ansible Automation Platform 作业以便在托管集群中运行的

Red Hat Ansible Automation Platform 与 multicluster engine operator 集成，以便您可以在创建或更新托管集群前或之后创建 prehook 和 posthook Ansible Automation Platform 作业实例。

需要的访问权限：集群管理员

- 先决条件
- 运行 Ansible Automation Platform 作业来安装托管集群
- 运行 Ansible Automation Platform 作业以更新托管集群
- 运行 Ansible Automation Platform 作业以删除托管集群

1.5.12.1. 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 4.14 或更高版本
- 安装 Ansible Automation Platform Resource Operator，将 Ansible Automation Platform 作业连接到 Git 订阅的生命周期。当使用 Automation 模板启动 Ansible Automation Platform 作业时，请确保 Ansible Automation Platform 作业模板在运行时是幂等的。您可以在 OpenShift Container Platform OperatorHub 中找到 Ansible Automation Platform Resource Operator。

1.5.12.2. 运行 Ansible Automation Platform 作业来安装托管集群

要启动安装托管集群的 Ansible Automation Platform 作业，请完成以下步骤：

1. 创建 HostedCluster 和 NodePool 资源，包括 pausedUntil: true 字段。如果使用 hcp create cluster 命令行界面命令，您可以指定 --pausedUntil: true 标志。

   请参见以下示例：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: my-cluster
```

```
  namespace: clusters
spec:
  pausedUntil: 'true'
...
```

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  name: my-cluster-us-east-2
  namespace: clusters
spec:
  pausedUntil: 'true'
...
```

2. 创建 ClusterCurator 资源，其名称与 HostedCluster 资源相同，并在与 HostedCluster 资源相同的命名空间中。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
  desiredCuration: install
  install:
    jobMonitorTimeout: 5
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
          variable2: 2
      - name: Demo Job Template
    posthook:
      - name: Demo Job Template
    towerAuthSecret: toweraccess
```

3. 如果您的 Ansible Automation Platform Tower 需要身份验证，请创建一个 secret 资源。请参见以下示例：

```
apiVersion: v1
kind: Secret
metadata:
  name: toweraccess
  namespace: clusters
stringData:
  host: https://my-tower-domain.io
  token: ANSIBLE_TOKEN_FOR_admin
```

1.5.12.3. 运行 Ansible Automation Platform 作业以更新托管集群

要运行更新托管集群的 Ansible Automation Platform 作业，请编辑您要更新的托管集群的 ClusterCurator 资源。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
  desiredCuration: upgrade
  upgrade:
    desiredUpdate: 4.14.1 (1)
    monitorTimeout: 120
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
          variable2: 2
      - name: Demo Job Template
    posthook:
      - name: Demo Job Template
    towerAuthSecret: toweraccess
```

(1) 有关支持的版本的详情，请参阅 托管 control plane。

注：当您以这种方式更新托管集群时，您可以将托管的 control plane 和节点池更新至同一版本。不支持将托管的 control plane 和节点池更新至不同的版本。

1.5.12.4. 运行 Ansible Automation Platform 作业以删除托管集群

要运行删除托管集群的 Ansible Automation Platform 作业，请编辑您要删除的托管集群的 ClusterCurator 资源。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
```

```
  desiredCuration: destroy
  destroy:
    jobMonitorTimeout: 5
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
          variable2: 2
      - name: Demo Job Template
    posthook:
      - name: Demo Job Template
    towerAuthSecret: toweraccess
```

注： 不支持删除 AWS 上的托管集群。

1.5.12.5. 其他资源

- 有关托管 control plane 命令行界面 hcp 的更多信息，请参阅安装托管的 control plane 命令行界面。
- 有关托管集群的更多信息，包括支持的版本，请参阅 托管 control plane。

1.5.13. 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

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
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: 72001Mi
  clusterClaims:
    - name: id.k8s.io
      value: cluster1
    - name: kubeversion.open-cluster-management.io
      value: v1.18.3+6c42de8
    - name: platform.open-cluster-management.io
      value: AWS
    - name: product.open-cluster-management.io
      value: OpenShift
    - name: id.openshift.io
      value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    - name: consoleurl.openshift.io
      value: 'https://console-openshift-console.apps.xxxx.dev04.red-chesterfield.com'
    - name: version.openshift.io
      value: '4.13'
  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
```

1.5.13.1. 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
```

1.5.13.2. 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。

1.5.14. 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

1.5.14.1. 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
```

1.5.14.1.1. 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 > 替换为受管集群集的名称。

1.5.14.1.2. 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。

1.5.14.1.3. 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：如果要将受管集群从一个 ManagedClusterSet 移到另一个，则必须在两个受管集群集中都有该权限。

2.

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
```

1.5.14.2. 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。

集群集	访问权限	创建权限
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，请参阅从 ManagedCusterSets 过滤 ManagedClusters。

1.5.14.3. 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。

1.5.14.3.1. 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集群集绑定到的命名空间。

1.5.14.3.2. 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"]
```

1.5.14.4. 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。

1.5.14.4.1. 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。

1.5.14.4.2. 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'
```

1.5.14.4.3. 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，运算符是以下值之一：
  - equal - 运算符 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。

1.5.14.4.4. 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。

1.5.14.5. 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。

备注：

- 每个受管集群都必须分配给受管集群集。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。
- 如果受管集群上安装了 Submariner 附加组件，则必须在从 ManagedClusterSet 中删除受管集群前卸载附加组件。

1.5.14.5.1. 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，删除您要从集群集合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：如果要将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有所需的 RBAC 权限。

1.5.14.5.2. 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。

1.5.15. Placement

放置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：集群管理员，Cluster set administrator

继续阅读以了解更多有关如何使用放置的信息：

- *放置概述*
- *从 ManagedCusterSets过滤 ManagedClusters*
- *使用 PlacementDecisions检查所选 ManagedClusters*

1.5.15.1. 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 ManagedClusterSets 的一部分，该部分选择潜在 ManagedClusters 的工作集合。*
- *placements 使用 labelSelector 和 claimSelector 从 ManagedClusterSets 过滤 ManagedClusters。*
- *ManagedClusters 的放置可以使用污点和容限控制。*
- *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：

-

您必须通过在该命名空间中创建一个 ManagedClusterSetBinding 来至少将一个 ManagedClusterSet 绑定到一个命名空间。

- 您需要在 managedclustersets/bind 的虚拟子资源上对 CREATE 进行基于角色的访问权限。

1.5.15.1.1. 其他资源

- 如需更多信息，请参阅使用污点和容限放置受管集群。
- 请参阅 Placements API 以了解更多有关 API 的信息。
- 返回到 选择带有放置 的 ManagedClusters 。

1.5.15.2. 从 ManagedClusterSets过滤 ManagedClusters

您可以使用 labelSelector 或 claimSelector 选择要过滤的 ManagedClusters。请参阅以下示例以了解如何同时使用这两个过滤器：

- 在以下示例中，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
```

- 在以下示例中，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
```

- 您还可以使用 clusterSets 参数从特定集群集合中过滤 ManagedClusters。在以下示例中，claimSelector 仅与集群设置 clusterset1 和 clusterset2 ：

```
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
```

您还可以使用 numberOfClusters 参数选择要过滤的 ManagedClusters 数量。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 3 (1)
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
```

```
      operator: In
      values:
        - us-west-1
```

1

指定您要选择的 ManagedClusters 数量。前面的示例设置为 3。

1.5.15.2.1. 使用放置定义容限来过滤 ManagedClusters

要了解如何过滤带有匹配污点的 ManagedClusters，请参阅以下示例：

- 默认情况下，放置无法在以下示例中选择 cluster1 ：

```
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'
```

要选择 cluster1，您必须定义容限。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal
```

您还可以使用 tolerationSeconds 参数，为指定时间选择带有匹配污点的 ManagedClusters。tolerationSeconds 定义了一个容限保持与污点绑定的时长。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。

查看以下示例以了解如何使用 tolerationSeconds ：

- 在以下示例中，受管集群无法访问：

```
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，工作负载将传送到另一个可用的受管集群。请参见以下示例：

```
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)
```

(1) 指定您希望在多数秒以后将工作负载转移。

1.5.15.2.2. 通过使用放置定义 prioritizerPolicy 来优先考虑 ManagedClusters

查看以下示例，了解如何在放置中使用 prioritizerPolicy 参数来优先选择 ManagedClusters。

- 以下示例选择具有最大可分配内存的集群：

注：与 Kubernetes Node Allocatable 类似，"可分配"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
```

- 以下示例选择具有最大可分配 CPU 和内存的集群，并对资源更改敏感：

```
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
```

- 以下示例选择具有最大 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
```

1.5.15.2.3. 根据附加组件状态过滤 ManagedClusters

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，只有在受管集群上启用了特定的附加组件时，才能为放置选择受管集群。

在创建放置时，您可以为附加组件指定标签及其状态。如果受管集群上启用了附加组件，则会自动在 ManagedCluster 资源中创建一个标签。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>。

将 addon_name 替换为您要在所选受管集群中启用的附加组件的名称。

如果选择了受管集群，将 status_of_addon 替换为您要附加组件的状态。

下表列出了 status_of_addon 的值：

值	描述
可用	附加组件已启用并可用。
不健康	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，其中包含以下内容：

```
feature.open-cluster-management.io/addon-application-manager: available
```

请参阅以下示例，了解如何根据附加组件及其状态创建放置：

- 以下放置示例包括启用了 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
```

- 以下放置示例包括启用了 application-manager 且状态为 available 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"
```

- 以下放置示例包括禁用 application-manager 的所有受管集群：

```
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
```

1.5.15.2.4. 其他资源

- 如需了解更多详细信息，请参阅 Node Allocatable。

- 返回到 选择带有放置的 ManagedClusters，用于其他主题。

1.5.15.3. 使用 PlacementDecisions检查所选 ManagedClusters

创建一个或多个带有标签 cluster.open-cluster-management.io/placement={placement_name} 的 PlacementDecision kind 来代表放置选择的 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此 ManagedCluster 上应用工作负载。当 ManagedCluster 不再被选择并从 PlacementDecision 中删除后，在此 ManagedCluster 上应用的工作负载会被删除。请参阅 PlacementDecisions API 来了解更多有关 API 的信息。

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: ''
```

1.5.15.3.1. 其他资源

- 如需了解更多详细信息，请参阅 PlacementDecisions API。

1.5.16. 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，池会为它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。

1.5.16.1. 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

1.5.16.1.1. 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/createmanagedcluster: "false" 1
```

1 "false" 必须用引号括起来，表示它是一个字符串。

1.5.16.1.2. 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

1.5.16.2. 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

1.5.16.2.1. 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。

1.5.16.2.2. 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。

1.5.16.3. 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。

1.5.16.4. 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。

1.5.16.5. 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。

重要： 您只能销毁没有集群声明的集群池。

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，在确认框中输入 confirm，然后选择 Destroy。

   备注：

- 如果集群池有任何集群声明，则 Destroy 按钮被禁用。

- 包含集群池的命名空间不会被删除。删除命名空间会销毁从集群池声明的任何集群，因为这些集群的集群声明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。

1.5.17. 启用 ManagedServiceAccount 附加组件

安装 multicluster engine operator 版本 2.5 或更高版本时，ManagedServiceAccount 附加组件会被默认启用。如果您从 multicluster engine operator 版本 2.4 升级 hub 集群，且在升级前没有启用 ManagedServiceAccount 附加组件，则必须手动启用附加组件。

ManagedServiceAccount 允许您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。

1.5.17.1. 先决条件

- Red Hat OpenShift Container Platform 版本 4.13 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。

1.5.17.2. 启用 ManagedServiceAccount

要为 hub 集群和受管集群启用 ManagedServiceAccount 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 ManagedServiceAccount 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: <managedserviceaccount_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
```

1.5.18. 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。

1.5.18.1. 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 API 服务器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：

- 您要将外部负载均衡器的默认 API 服务器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。

- 外部负载均衡器的命名证书是过期的，您需要替换它。如果旧证书和新证书都共享相同的 root CA 证书，虽然中间证书数量相同，您可以遵循 OpenShift Container Platform 文档中的 Adding API 服务器证书中的 指导来为新证书创建新 secret。然后，将主机名 api.

<cluster_name>.<base_domain> 的 serving 证书引用更新为 APIServer 自定义资源中的新 secret。否则，当旧证书和新证书有不同的 root CA 证书时，请完成以下步骤替换证书：

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& gt; 来访问 hub 集群。您不能使用配置了其他主机名的命名证书。

1.5.18.2. 配置 hub 集群和受管集群之间的代理

要将受管集群注册到 Kubernetes operator hub 集群的多集群引擎，您需要将受管集群传输到 multicluster engine operator hub 集群。有时您的受管集群无法直接访问多集群引擎 operator hub 集

群。在本例中，将代理设置配置为允许来自受管集群的通信通过 HTTP 或 HTTPS 代理服务器访问 multicluster engine operator hub 集群。

例如，多集群引擎 operator hub 集群位于公有云中，受管集群位于防火墙后面的私有云环境中。来自私有云的通信只能通过 HTTP 或 HTTPS 代理服务器进行。

1.5.18.2.1. 先决条件

- 您有一个支持 HTTP 隧道的 HTTP 或 HTTPS 代理服务器。例如，HTTP 连接方法。
- 您有一个可访问 HTTP 或 HTTPS 代理服务器的管理集群，代理服务器可以访问 multicluster engine operator hub 集群。

完成以下步骤以配置 hub 集群和受管集群之间的代理设置：

1. 使用代理设置创建 KlusterConfig 资源。
   a. 请参阅使用 HTTP 代理的以下配置：

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: http-proxy
spec:
  hubKubeAPIServerProxyConfig:
    httpProxy: "http://<username>:<password>@<ip>:<port>"
```

   b. 请参阅使用 HTTPS 代理的以下配置：

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: https-proxy
spec:
  hubKubeAPIServerProxyConfig:
    httpsProxy: "https://<username>:<password>@<ip>:<port>"
    caBundle: <user-ca-bundle>
```

注：当配置 HTTPS 代理服务器时，需要 CA 证书。如果指定了 HTTP 代理和 HTTPS 代理，则使用 HTTPS 代理。

2. 在创建受管集群时，通过添加引用 KlusterletConfig 资源的注解来选择 KlusterletConfig 资源。请参见以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    agent.open-cluster-management.io/klusterlet-config: <klusterlet-config-name>
  name:<managed-cluster-name>
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

注： 当您操作 multicluster engine operator 控制台时，您可能需要切换 YAML 视图，将注解添加到 ManagedCluster 资源。

1.5.18.2.2. 禁用 hub 集群和受管集群之间的代理

如果开发更改，您可能需要禁用 HTTP 或 HTTPS 代理。

1. 进入 ManagedCluster 资源。
2. 删除注解 agent.open-cluster-management.io/klusterlet-config。

1.5.18.2.3. 可选：将 klusterlet 配置为在特定节点上运行

当使用 Red Hat Advanced Cluster Management for Kubernetes 创建集群时，您可以通过为受管集群配置 nodeSelector 和 tolerations 注解来指定您要运行受管集群 klusterlet 的节点。完成以下步骤以配置这些设置：

1. 从控制台的集群页面中选择要更新的受管集群。
2. 将 YAML 开关设置为 On 以查看 YAML 内容。

注： YAML 编辑器仅在导入或创建集群时可用。要在导入或创建后编辑受管集群 YAML 定义，您必须使用 OpenShift Container Platform 命令行界面或 Red Hat Advanced Cluster Management 搜索功能。

3. 将 nodeSelector 注解添加到受管集群 YAML 定义。此注解的密钥是： open-cluster-management/nodeSelector。此注解的值是带有 JSON 格式的字符串映射。

4. 将 tolerations 条目添加到受管集群 YAML 定义中。此注解的键为： open-cluster-management/tolerations。此注解的值代表带有 JSON 格式的 容限 列表。生成的 YAML 可能类似以下示例：

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    open-cluster-management/nodeSelector: '{"dedicated":"acm"}'
    open-cluster-management/tolerations:
'[{"key":"dedicated","operator":"Equal","value":"acm","effect":"NoSchedule"}]'
```

1.5.18.3. 在导入受管集群时自定义 hub 集群 API 服务器的服务器 URL 和 CA 捆绑包（技术预览）

如果受管集群和 hub 集群之间存在中间组件，则可能无法在 multicluster engine operator hub 集群中注册受管集群。中间组件示例包括虚拟 IP、负载均衡器、反向代理或 API 网关。如果您有中间组件，则必须在导入受管集群时为 hub 集群 API 服务器使用自定义服务器 URL 和 CA 捆绑包。

1.5.18.3.1. 先决条件

- 您必须配置中间组件，以便受管集群可以访问 hub 集群 API 服务器。
- 如果中间组件终止了受管集群和 hub 集群 API 服务器之间的 SSL 连接，则必须桥接 SSL 连接，并将原始请求的身份验证信息传递给 hub 集群 API 服务器的后端。您可以使用 Kubernetes API 服务器的用户 Impersonation 功能来桥接 SSL 连接。

  中间组件从原始请求中提取客户端证书，将证书主题的通用名称(CN)和机构(O)添加为模拟标头，然后将修改后的模拟请求转发到 hub 集群 API 服务器的后端。

  注： 如果您桥接 SSL 连接，集群代理附加组件无法正常工作。

1.5.18.3.2. 自定义服务器 URL 和 hub 集群 CA 捆绑包

要在导入受管集群时使用自定义 hub API 服务器 URL 和 CA 捆绑包，请完成以下步骤：

1. 使用自定义 hub 集群 API 服务器 URL 和 CA 捆绑包创建一个 KlusterConfig 资源。请参

见以下示例：

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: 1
spec:
  hubKubeAPIServerURL: "https://api.example.com:6443" 2
  hubKubeAPIServerCABundle: "LS0tLS1CRU...LS0tCg==" 3
```

1 *添加 klusterlet 配置名称。*

2 *添加自定义服务器 URL。*

3 *添加自定义 CA 捆绑包。*

2. *通过添加引用该资源的注解在创建受管集群时，选择 KlusterletConfig 资源。请参见以下示例：*

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    agent.open-cluster-management.io/klusterlet-config: 1
  name: 2
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

1 *添加 klusterlet 配置名称。*

2 *添加集群名称。*

注：如果使用控制台，您可能需要启用 YAML 视图来将注解添加到 ManagedCluster 资源。

1.5.18.4. 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
- 为集群代理附加组件配置代理设置

1.5.19. 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。
- 当您分离或销毁受管集群时，相关的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

1.5.19.1. 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。

1.5.19.2. 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'setting 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。

1.5.19.3. 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. 使用以下命令删除 klusterlet 自定义资源：

```
oc get klusterlet | grep klusterlet | awk '{print $1}' | xargs oc patch klusterlet --type=merge -p '{"metadata":{"finalizers": []}}'
```

4. 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": []}}'
```

5. 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
```

1.5.19.4. 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。

1.5.19.4.1. 先决条件

- 安装 OpenShift CLI（oc）。
- 以具有 cluster-admin 特权的用户身份登录。

1.5.19.4.2. 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:
    - "4.13"
```

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: 4.13
status: Stale
```

1.7. 托管 CONTROL PLANE

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。

OpenShift Container Platform 托管 control plane 在裸机和 Red Hat OpenShift Virtualization 上提供，并作为 Amazon Web Services (AWS)上的 技术预览功能 提供。您可以在 OpenShift Container Platform 版本 4.14 上托管 control plane。托管的 control plane 功能默认启用。

1.7.1. 要求

下表显示每个平台都支持哪些 OpenShift Container Platform 版本。在表中，Hosting OpenShift Container Platform 版本 指的是启用了 multicluster engine operator 的 OpenShift Container Platform 版本：

平台	托管 OpenShift Container Platform 版本	托管 OpenShift Container Platform 版本
裸机	4.14 - 4.15	4.14 - 4.15 （仅限）
Red Hat OpenShift Virtualization	4.14 - 4.15	4.14 - 4.15 （仅限）
AWS （技术预览）	4.11 - 4.15	4.14 - 4.15 （仅限）

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*

要使用托管的 control plane，请从以下文档开始：

1. *托管 control plane 大小指导*
2. *安装托管的 control plane 命令行界面*
3. *分发托管集群工作负载*

然后，查看与计划使用的平台相关的文档：

- *在 AWS 上配置托管的 control plane 集群（技术预览）*
- *在裸机上配置托管的 control plane 集群*
- *在 64 位 x86 OpenShift Container Platform 集群上配置托管集群，为 IBM Power 计算节点创建托管的 control plane （技术预览）*
- *为 IBM Z 计算节点在 x86 裸机上配置托管集群（技术预览）*

- 在 OpenShift Virtualization 中管理托管的 control plane 集群
- 在断开连接的环境中配置托管的 control plane
- 禁用托管的控制功能

要为节点池配置额外网络、保证 CPU 和虚拟机调度，请参阅以下文档：

- 为节点池配置额外网络、保证 CPU 和虚拟机调度

有关托管 control plane 的其他资源，请参阅以下 OpenShift Container Platform 文档：

- 托管 control plane 的架构
- 托管控制平面（control plane）的常见概念和用户角色表
- 为托管集群创建监控仪表板
- 托管的 control plane 的备份、恢复和灾难恢复

1.7.2. 托管 control plane 大小指导

许多因素（包括托管集群工作负载和 worker 节点数）会影响到特定数量的 control-plane 节点中可以容纳多少个托管集群。使用此大小指南来帮助进行托管集群容量规划。这个指南假设高度可用的托管 control plane 拓扑。在裸机集群中测量基于负载的大小示例。基于云的实例可能具有不同的限制因素，如内存大小。有关高可用性托管 control plane 拓扑的更多信息，请参阅 分配托管集群工作负载。

您可以覆盖以下资源利用率大小测量并禁用指标服务监控。如需更多信息，请参阅附加资源部分中的覆盖资源利用率测量。

请参阅以下高可用性托管的 control plane 要求，这些要求已使用 OpenShift Container Platform 版

本 4.12.9 及更新版本进行测试：

- *78 个 pod*
- *etcd 三个 8 GiB PV*
- *最小 vCPU：大约 5.5 个内核*
- *最小内存：大约 19 GiB*

1.7.2.1. Pod 限值

每个节点的 maxPods 设置会影响 control-plane 节点可以容纳多少个托管集群。务必要记录所有 control-plane 节点上的 maxPods 值。为每个高可用性托管的 control plane 大约 75 个 pod 计划。

对于裸机节点，默认的 maxPods 设置可能是一个限制因素，因为通常三个托管的 control plane 适合每个节点，即使机器有可备用的资源。通过配置 KubeletConfig 值将 maxPods 值设置为 500，以获得更大的托管 control plane 密度，这有助于利用额外的计算资源。如需更多信息，请参阅 OpenShift Container Platform 文档中的配置每个节点的最大 pod 数量。

1.7.2.2. 基于请求的资源限值

集群可以托管的 control plane CPU 和来自 pod 的内存请求的最大数量。

高可用性托管的 control plane 由请求 5 个 vCPU 和 18 GB 内存的 78 个 pod 组成。这些基准号与集群 worker 节点资源容量进行比较，以估算托管 control plane 的最大数量。

1.7.2.3. 基于负载的限制

当某些工作负载放在托管 control plane Kubernetes API 服务器上时，集群可以托管的 control plane pod CPU 和内存使用率计算集群可以托管的 control plane 的最大 control plane 数量。

以下方法用于在工作负载增加时测量托管的 control plane 资源利用率：

-

具有 9 个 worker 的托管集群，每个 worker 使用 8 个 vCPU 和 32 GiB，同时使用 KubeVirt 平台

- 根据以下定义，配置为专注于 API control-plane stress 的工作负载测试配置集：
  - 为每个命名空间创建对象，扩展至 100 个命名空间总数
  - 通过持续删除和创建对象的其他 API 压力
  - 工作负载查询每秒(QPS)和 Burst 设置设置高，以删除任何客户端节流

当负载增加 1000 QPS 时，托管的 control plane 资源使用率会增加 9 个 vCPU 和 2.5 GB 内存。

出于常规大小，请考虑 1000 QPS API 速率，它是 中型 托管集群负载，以及一个 重度 托管的集群负载的 2000 QPS API。

注： 此测试提供了一个估算因素，用于根据预期的 API 负载增加计算资源利用率。确切利用率率可能会因集群工作负载的类型和速度而异。

表 1.7. 载入表

托管 control plane 资源使用率扩展	VCPU	内存(GiB)
没有负载的资源使用率	2.9	11.1
使用 1000 QPS 的资源利用率	9.0	2.5

当负载增加 1000 QPS 时，托管的 control plane 资源使用率会增加 9 个 vCPU 和 2.5 GB 内存。

出于常规大小，请考虑 1000 QPS API 速率作为中型托管集群负载，以及 2000 QPS API 作为大量托管集群负载。

以下示例显示了工作负载和 API 速率定义的托管 control plane 资源扩展：

表 1.8. API 速率表

QPS (API rate)	vCPU 用量	内存用量(GiB)
低负载(Less than 50 QPS)	2.9	11.1
中型负载(1000 QPS)	11.9	13.6
高负载(2000 QPS)	20.9	16.1

托管 control plane 大小是 control-plane 负载，以及导致大量 API 活动、etcd 活动或两者的工作负载。侧重于数据平面负载的托管 pod 工作负载（如运行数据库）可能会导致 API 速率高。

1.7.2.4. 大小计算示例

这个示例为以下情况提供了大小调整指导：

- *三个标记为 hypershift.openshift.io/control-plane 节点的裸机 worker*
- *maxPods 值设为 500*
- *根据基于负载的限制，预期的 API 速率为 medium 或大约 1000*

表 1.9. 限制输入

限制描述	服务器 1	服务器 2
worker 节点上的 vCPU 数量	64	128
worker 节点上的内存(GiB)	128	256
每个 worker 的最大 pod	500	500
用于托管 control plane 的 worker 数量	3	3
最大 QPS 目标率（每秒的 API 请求）	1000	1000

表 1.10. 大小计算示例

根据 worker 节点大小和 API 速率计算值	服务器 1	服务器 2	计算备注
根据 vCPU 请求，每个 worker 的最大托管 control plane	12.8	25.6	每个托管的 control plane 的 worker vCPU total 5 个 vCPU 请求数
根据 vCPU 用量，每个 worker 的最大托管 control plane	5.4	10.7	vCPUS HBAC (2.9 测量闲置 vCPU 用量 +（QPS 目标速率 maximum maximum 1000) measured vCPU 用量（每 1000 QPS 测量的 vCPU 使用量）
根据内存请求，每个 worker 的最大托管 control plane	7.1	14.2	worker 内存 GiB 每个托管的 control plane 内存请求总数
根据内存用量，每个 worker 的最大托管 control plane	9.4	18.8	worker 内存 GiB HBAC (11.1 measured idle memory usage +(QPS target rate packagemanifests 1000) measured 2.5 measured memory usage per 1000 QPS increase)
根据每个节点的 pod 限制，每个 worker 的最大托管 control plane	6.7	6.7	每个托管的 control plane 500 **maxPods** 75 个 pod
前面提到的最大值	5.4	6.7	
	vCPU 限制因素	**maxPods** 限制因素	
管理集群中托管 control plane 的最大数量	16	20	前面提到的最少 3 个 control-plane worker

表 1.11. 托管 control plane 容量指标

Name	描述
mce_hs_addon_request_based_hcp_capacity_gauge	估算集群可以基于高可用性托管 control plane 资源请求的最大托管 control plane 数量。
mce_hs_addon_low_qps_based_hcp_capacity_gauge	如果所有托管的 control plane 都大约为 50 QPS 到集群 Kube API 服务器，则预测集群可以托管的最大 control plane 数量。

mce_hs_addon_medium_qps_based_hcp_capacity_gauge	如果所有托管的 control plane 都对集群 Kube API 服务器执行超过 1000 QPS，则预测集群可以托管的最大 control plane 数量。
mce_hs_addon_high_qps_based_hcp_capacity_gauge	如果所有托管的 control plane 都围绕 2000 QPS 到集群 Kube API 服务器，则预测集群可以托管的最大 control plane 数量。
mce_hs_addon_average_qps_based_hcp_capacity_gauge	根据托管 control plane 的现有平均 QPS 估算集群可以托管的 control plane 的最大数量。如果您没有活跃的托管 control plane，您可以预期低 QPS。

1.7.2.5. 其他资源

- *分发托管集群工作负载*
- *配置每个节点的最大 pod 数量*
- *覆盖资源使用率测量*

1.7.3. 覆盖资源使用率测量

资源利用率的基准测量集合可能会因每个集群而异。如需更多信息，请参阅 托管 control plane 大小指导。

您可以根据集群工作负载的类型和速度覆盖资源利用率测量。完成以下步骤：

1. *运行以下命令来创建 ConfigMap 资源：*

```
oc create -f <your-config-map-file.yaml>
```

将 <your-config-map-file.yaml > 替换为包含 hcp-sizing-baseline 配置映射的 YAML 文件的名称。

2. *在 local-cluster 命名空间中创建 hcp-sizing-baseline 配置映射，以指定您要覆盖的测量。您的配置映射可能类似以下 YAML 文件：*

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: hcp-sizing-baseline
  namespace: local-cluster
data:
  incrementalCPUUsagePer1KQPS: "9.0"
  memoryRequestPerHCP: "18"
  minimumQPSPerHCP: "50.0"
```

3. 运行以下命令，删除 **hypershift-addon-agent** 部署来重启 **hypershift-addon-agent pod**：

```
oc delete deployment hypershift-addon-agent -n open-cluster-management-agent-addon
```

4. 观察 **hypershift-addon-agent pod** 日志。运行以下命令，验证配置映射中是否已更新覆盖的测量：

```
oc logs hypershift-addon-agent -n open-cluster-management-agent-addon
```

您的日志可能类似以下输出：

```
2024-01-05T19:41:05.392Z INFO agent.agent-reconciler agent/agent.go:793 setting
cpuRequestPerHCP to 5
2024-01-05T19:41:05.392Z INFO agent.agent-reconciler agent/agent.go:802 setting
memoryRequestPerHCP to 18
2024-01-05T19:53:54.070Z INFO agent.agent-reconciler
agent/hcp_capacity_calculation.go:141 The worker nodes have 12.000000 vCPUs
2024-01-05T19:53:54.070Z INFO agent.agent-reconciler
agent/hcp_capacity_calculation.go:142 The worker nodes have 49.173369 GB memory
```

如果没有在 **hcp-sizing-baseline** 配置映射中正确更新覆盖的测量，您可能会在 **hypershift-addon-agent pod** 日志中看到以下出错信息：

```
2024-01-05T19:53:54.052Z ERROR agent.agent-reconciler agent/agent.go:788 failed to get
configmap from the hub. Setting the HCP sizing baseline with default values. {"error":
"configmaps \"hcp-sizing-baseline\" not found"}
```

1.7.3.1. 禁用指标服务监控

启用 **hypershift-addon** 受管集群附加组件后，指标服务监控会被默认配置，以便 **OpenShift Container Platform** 监控可以从 **hypershift-addon** 收集指标。您可以通过完成以下步骤来禁用指标服务监控：

1. 运行以下命令登录到您的 hub 集群：

```
oc login
```

2. 运行以下命令，打开 hypershift-addon-deploy-config 附加组件部署配置规格以进行编辑：

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

3. 在规格中添加 disableMetrics=true 自定义变量，如下例所示：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  customizedVariables:
  - name: hcMaxNumber
    value: "80"
  - name: hcThresholdNumber
    value: "60"
  - name: disableMetrics
    value: "true"
```

4. 保存更改。disableMetrics=true 自定义变量为新的和现有的 hypershift-addon 受管集群附加组件禁用指标服务监控配置。

1.7.3.2. 其他资源

- 托管 control plane 大小指导

1.7.4. 安装托管的 control plane 命令行界面

您可以通过完成以下步骤来安装托管的 control plane 命令行界面 hcp ：

1. 在 OpenShift Container Platform 控制台中点 Help 图标 > Command Line Tools。

2. 为您的平台点 **Download hcp CLI**。

3. 运行以下命令来解包下载的归档：

```
tar xvzf hcp.tar.gz
```

4. 运行以下命令使二进制文件可执行：

```
chmod +x hcp
```

5. 运行以下命令，将二进制文件移到路径的目录中：

```
sudo mv hcp /usr/local/bin/.
```

现在，您可以使用 **hcp create cluster** 命令来创建和管理托管集群。要列出可用的参数，请输入以下命令：

```
hcp create cluster <platform> --help (1)
```

(1) 支持的平台包括 **aws**、**代理** 和 **kubevirt**。

1.7.4.1. 使用 CLI 安装托管的 control plane 命令行界面

您可以通过完成以下步骤，使用 CLI 安装托管的 control plane 命令行界面 **hcp** ：

1. 运行以下命令，获取下载 **hcp** 二进制文件的 URL：

```
oc get ConsoleCLIDownload hcp-cli-download -o json | jq -r ".spec"
```

2. 运行以下命令下载 **hcp** 二进制文件：

```
wget <hcp_cli_download_url> (1)
```

1 将 hcp_cli_download_url 替换为您在上一步中获取的 URL。

3. 运行以下命令来解包下载的归档：

```
tar xvzf hcp.tar.gz
```

4. 运行以下命令使二进制文件可执行：

```
chmod +x hcp
```

5. 运行以下命令，将二进制文件移到路径的目录中：

```
sudo mv hcp /usr/local/bin/.
```

1.7.4.2. 使用内容网关安装托管的 control plane 命令行界面

您可以使用内容网关安装托管的 control plane 命令行界面 hcp。完成以下步骤：

1. 导航到 内容网关 并下载 hcp 二进制文件。

2. 运行以下命令来解包下载的归档：

```
tar xvzf hcp.tar.gz
```

3. 运行以下命令使二进制文件可执行：

```
chmod +x hcp
```

4. 运行以下命令，将二进制文件移到路径的目录中：

```
sudo mv hcp /usr/local/bin/.
```

现在，您可以使用 hcp create cluster 命令来创建和管理托管集群。要列出可用的参数，请输入以下命令：

```
hcp create cluster <platform> --help (1)
```

(1) 支持的平台包括 aws、代理 和 kubevirt。

1.7.5. 分发托管集群工作负载

在开始为 OpenShift Container Platform 托管 control plane 之前，您必须标记节点，将托管的集群 pod 调度到基础架构节点。

节点标签确保以下功能：

- 高可用性和正确的工作负载部署。例如，您可以设置 node-role.kubernetes.io/infra 标签，以避免将 control-plane 工作负载计数设置为 OpenShift Container Platform 订阅。
- 使 control plane 工作负载与管理集群中的其他工作负载分开。

重要：不要为您的工作负载使用管理集群。工作负载不得在 control plane 运行节点上运行。

1.7.5.1. 管理集群节点的标签和污点

作为管理集群管理员，在管理集群节点中使用以下标签和污点来调度 control plane 工作负载：

- HyperShift.openshift.io/control-plane: true ：使用此标签和污点来专用节点来运行托管的 control plane 工作负载。通过设置 true，您可以避免将 control plane 节点与其他组件共享。
- HyperShift.openshift.io/cluster: <hosted-control-plane-namespace > ：当您想要将节点专用于单个托管集群时使用此标签和污点。

在主机 control-plane pod 的节点上应用以下标签：

- ***node-role.kubernetes.io/infra ：使用此标签以避免将 control-plane 工作负载计数设置为您的订阅。***

- ***topology.kubernetes.io/zone: 在管理集群节点上使用此标签，在故障域中部署高可用性集群。区域可能是设置该区域的节点的位置、机架名称或主机名。***

要将每个机架用作管理集群节点的可用区，请输入以下命令：

+

```
oc label node/<management_node1_name> node/<management_node2_name>
topology.kubernetes.io/zone=<rack_name>
```

托管集群的 Pod 具有容限，调度程序使用关联性规则来调度它们。调度程序将 pod 调度到标记为 hypershift.openshift.io/control-plane 和 hypershift.openshift.io/cluster: <hosted_control_plane_namespace > 的节点。

对于 ControllerAvailabilityPolicy 选项，使用 HighlyAvailable，这是托管 control planes 命令行界面 hcp 的默认值。当使用该选项时，您可以通过将 topology.kubernetes.io/zone 设置为拓扑键，在不同的故障域中为每个部署调度 pod。不支持没有高可用性的 control plane。

1.7.5.2. 为托管集群标记节点

重要： 在部署托管的 control plane 前，您必须将标签添加到节点。

要将在托管集群中运行的 pod 调度到基础架构节点，请在 HostedCluster 自定义资源(CR)中添加 role.kubernetes.io/infra: "" 标签。请参见以下示例：

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

1.7.5.3. 优先级类

四个内置优先级类会影响托管集群 pod 的优先级和抢占。您可以按照从高到低的顺序在管理集群中创建 pod：

- *HyperShift-operator: HyperShift Operator pod。*
- *HyperShift-etcd: etcd 的 Pod。*
- *HyperShift-api-critical ： API 调用和资源准入所需的 Pod 才能成功。此优先级类包括 kube-apiserver、聚合 API 服务器和 web hook 等 pod。*
- *HyperShift-control-plane: control plane 中的 Pod 不是 API-critical，但仍需要提升优先级，如集群版本 Operator。*

1.7.5.4. 其他资源

有关托管 control plane 的更多信息，请参阅以下主题：

- *在裸机上配置托管的 control plane 集群*
- *在 OpenShift Virtualization 中管理托管的 control plane 集群*
- *在 AWS 上配置托管的 control plane 集群（技术预览）*

1.7.6. 在 AWS 上配置托管的 control plane 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。hypershift-addon 受管集群附加组件默认为 multicluster engine operator 2.5 和 Red Hat Advanced Cluster Management 2.10 hub 集群中的 local-cluster 受管集群启用。

托管的集群 是一个 OpenShift Container Platform 集群，其 API 端点和托管在托管集群中的 control plane。托管的集群包括控制平面和它的对应的数据平面。您可以使用 multicluster engine operator 控制台或托管的 control plane 命令行界面 hcp 创建托管集群。托管的集群自动导入为受管集群。如果要禁用此自动导入功能，请参阅禁用将托管集群的自动导入到 multicluster engine operator。

重要：

- 每个托管集群都必须具有集群范围的唯一名称。托管的集群名称不能与任何现有受管集群相同，以便 multicluster engine operator 管理它。

- 不要使用 集群 作为托管的集群名称。

- 在托管 control plane 的同一平台上运行 hub 集群和 worker。

- 无法在 multicluster engine operator 受管集群的命名空间中创建托管集群。

1.7.6.1. 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 2.5 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 multicluster engine operator 2.5 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
```

- AWS 命令行界面

- 托管 control plane 命令行界面

有关托管 control plane 的其他资源，请参阅以下文档：

- 要禁用托管的 control plane 功能，或者已经禁用了它并希望手动启用它，请参阅 启用或禁用托管的 control plane 功能。

-

要通过运行 Red Hat Ansible Automation Platform 作业来管理托管集群，请参阅配置 Ansible Automation Platform 作业以便在托管集群中运行。

- 要部署 SR-IOV Operator，请参阅为托管的 control plane 部署 SR-IOV Operator。

1.7.6.2. 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：

```
aws s3api create-bucket --bucket <your-bucket-name>
aws s3api delete-public-access-block --bucket <your-bucket-name>
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*"
        }
    ]
}' | envsubst > policy.json
aws s3api put-bucket-policy --bucket <your-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：

```
aws s3api create-bucket --bucket <your-bucket-name> \
  --create-bucket-configuration LocationConstraint=<region> \
  --region <region>
aws s3api delete-public-access-block --bucket <your-bucket-name>
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*"
        }
```

```
    ]
}' | envsubst > policy.json
aws s3api put-bucket-policy --bucket <your-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	包含对托管集群的主机 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=<path>/.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
```

1.7.6.3. 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：

```
aws route53 create-hosted-zone --name <your-basedomain> --caller-reference $(whoami)-$(date --rfc-3339=date)
```

将 your-basedomain 替换为您的基域，例如 www.example.com。

1.7.6.4. 启用外部 DNS

control plane 和数据平面在托管的 control plane 中分离。您可以在两个独立区域中配置 DNS：

- 托管集群中工作负载的 Ingress，如以下域：.. apps.service-consumer-domain.com
- 管理集群中的服务端点的 Ingress，如 API 或 OAUTH 端点，通过服务供应商域：... service-provider-domain.com

hostedCluster.spec.dns 的输入管理托管集群中工作负载的入口。hostedCluster.spec.services.servicePublishingStrategy.route.hostname 的输入管理管理集群中服务端点的入口。

外部 DNS 为托管的集群服务创建名称记录，用于指定 LoadBalancer 或 Route 的发布类型，并为该发布类型提供主机名。对于使用 Private 或 PublicAndPrivate 端点访问类型的托管集群，只有 APIServer 和 OAuth 服务支持主机名。对于私有集群，DNS 记录解析为 VPC 中 Virtual Private Cloud (VPC)端点的专用 IP 地址。

托管的 control plane 会公开以下服务：

- APIServer
- OAuthServer
- Konnectivity

- Ignition
- OVNSbDb
- OIDC

您可以使用 HostedCluster 规格中的 servicePublishingStrategy 字段公开这些服务。默认情况下，对于 LoadBalancer 和 Route 类型的 servicePublishingStrategy，您可以使用以下方法之一发布该服务：

- 使用处于 LoadBalancer 类型的 Service 状态的负载均衡器的主机名
- 使用 Route 资源的 status.host 字段

但是，当您在受管服务上下文中部署托管的 control plane 时，这些方法可以公开底层管理集群的 ingress 子域，以及管理集群生命周期和灾难恢复的限制选项。

当在 LoadBalancer 和 Route 发布类型上分层 DNS 间接时，受管服务操作员可以使用服务级别域发布所有公共托管的集群服务。此架构允许在 DNS 名称上重新映射到新的 LoadBalancer 或 Route，且不会公开管理集群的 ingress 域。托管 control plane 使用外部 DNS 实现该间接层。

您可以在管理集群的 hypershift 命名空间中部署 external-dns 和 HyperShift Operator。外部 DNS 监视具有 external-dns.alpha.kubernetes.io/hostname 注解的服务或路由。该注解用于创建指向服务的 DNS 记录，如一个记录，或路由，如一个 CNAME 记录。

您只能在云环境中使用外部 DNS。对于其他环境，您需要手动配置 DNS 和服务。

有关外部 DNS 的更多信息，请参阅 外部 DNS。

1.7.6.4.1. 先决条件

在为托管的 control plane 设置外部 DNS 前，您必须满足以下先决条件：

- *您创建了外部公共域*
- *您可以访问 AWS Route53 管理控制台*

1.7.6.4.2. 为托管的 control plane 设置外部 DNS

要使用服务级别 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 是否具有所需字段：*

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。	当使用 AWS 密钥时可选
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

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=<domain_name> --from-file=credentials=<path_to_aws_credentials_file> -n local-cluster
```

注：不会自动启用 secret 的恢复备份。要备份灾难恢复的 secret，请输入以下命令添加 hypershift-operator-external-dns-credentials ：

```
oc label secret hypershift-operator-external-dns-credentials -n local-cluster cluster.open-cluster-management.io/backup=""
```

1.7.6.4.3. 创建公共 DNS 托管区

External DNS Operator 使用公共 DNS 托管区来创建公共托管集群。

您可以创建公共 DNS 托管区，以用作外部 DNS domain-filter。在 AWS Route 53 管理控制台中完成以下步骤：

1. 在 Route 53 管理控制台中，单击 Create hosted zone。
2. 在 Hosted zone configuration 页面上，输入域名，验证 Publish 托管区域 是否选为类型，然后单击 Create hosted zone。
3. 创建区域后，在 Records 选项卡上，记录 Value/Route 流量到 列中的值。
4. 在主域中，创建一个 NS 记录，将 DNS 请求重定向到委派的区域。在 Value 字段中，输入您在上一步中记下的值。
5. 点 Create records。
6. 通过在新的子区中创建测试条目并使用类似以下示例的 dig 命令测试 DNS 托管区来验证 DNS 托管区是否正常工作：

```
dig +short test.user-dest-public.aws.kerberos.com
192.168.1.1
```

7. 要创建为 LoadBalancer 和 Route 服务设置主机名的托管集群，请输入以下命令：

```
hcp create cluster aws --name=<hosted_cluster_name> --endpoint-
access=PublicAndPrivate --external-dns-domain=<public_hosted_zone> ...
```

将 <public_hosted_zone > 替换为您创建的公共托管区。

本例显示了为托管集群生成的 服务 块：

```
  platform:
    aws:
      endpointAccess: PublicAndPrivate
...
  services:
  - service: APIServer
    servicePublishingStrategy:
      route:
        hostname: api-example.service-provider-domain.com
      type: Route
  - service: OAuthServer
    servicePublishingStrategy:
      route:
        hostname: oauth-example.service-provider-domain.com
      type: Route
  - service: Konnectivity
    servicePublishingStrategy:
      type: Route
  - service: Ignition
    servicePublishingStrategy:
      type: Route
```

Control Plane Operator 创建 Services 和 Routes 资源，并使用 external-dns.alpha.kubernetes.io/hostname 注解为它们添加注解。对于 Services 和 Routes，Control Plane Operator 在服务端点的 servicePublishingStrategy 字段中使用 hostname 参数的值。要创建 DNS 记录，您可以使用一个机制，如 external-dns 部署。

您只能为公共服务配置服务级别 DNS 间接。您无法为私有服务设置主机名，因为它们使用 hypershift.local 私有区。

下表包括在什么时候它可以用来对于服务和端点组合设置主机名：

服务	公开	PublicAndPrivate	私有
APIServer	Y	Y	N
OAuthServer	Y	Y	N
Konnectivity	Y	N	N
Ignition	Y	N	N

1.7.6.4.4. 使用命令行界面和外部 DNS 部署集群

要使用 PublicAndPrivate 或 Public publish 策略创建托管集群，您必须在管理集群中配置以下工件：

- *公共 DNS 托管区*
- *External DNS Operator*
- *HyperShift Operator*

要使用命令行界面部署托管集群，请完成以下步骤：

1. *要访问您的管理集群，请输入以下命令：*

```
export KUBECONFIG=<path_to_management_cluster_kubeconfig>
```

2. *输入以下命令验证外部 DNS Operator 是否正在运行：*

```
oc get pod -n hypershift -lapp=external-dns
```

请参见以下示例输出：

```
NAME                            READY   STATUS    RESTARTS   AGE
external-dns-7c89788c69-rn8gp   1/1     Running   0          40s
```

3. *要使用外部 DNS 创建托管集群，请输入以下命令：*

```
hypershift create cluster aws \
  --aws-creds <path_to_aws_credentials_file> \ 1
  --instance-type <instance_type> \ 2
  --region <region> \ 3
  --auto-repair \
  --generate-ssh \
  --name <hosted_cluster_name> \ 4
  --namespace clusters \
  --base-domain <service_consumer_domain> \ 5
  --node-pool-replicas <node_replica_count> \ 6
  --pull-secret <path_to_your_pull_secret> \ 7
  --release-image quay.io/openshift-release-dev/ocp-release:<ocp_release_image> \
```

```
  --external-dns-domain=<service_provider_domain> \ 9
  --endpoint-access=PublicAndPrivate 10
```

1 指定 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

2 指定实例类型，如 m6i.xlarge。

3 指定 AWS 区域，如 us-east-1。

4 指定托管的集群名称，如 my-external-aws。

5 指定服务消费者拥有的公共托管区，如 service-consumer-domain.com。

6 指定节点副本数，例如 2。

7 指定 pull secret 文件的路径。

8 指定您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。

9 指定服务提供商拥有的公共托管区，如 service-provider-domain.com。

10 将 设置为 PublicAndPrivate。您只能在公共或 公共 配置中使用外部 DNS 和 PublicAndPrivate 配置。

1.7.6.5. 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	必需
aws-access-key-id	凭证访问密钥 id。	必需
aws-secret-access-key	凭证访问密钥 secret。	必需

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=""
```

1.7.6.6. 托管集群的灾难恢复

托管 control plane 在 multicluster engine operator hub 集群上运行。data plane 在您选择的独立平台上运行。当从灾难中恢复多集群引擎 operator hub 集群时，您可能还想恢复托管的 control plane。

请参阅 AWS 区域托管的集群的灾难恢复，以了解如何备份托管的 control plane 集群并在不同的集群中恢复它。

重要： 托管集群的灾难恢复只在 AWS 上可用。

1.7.6.7. 在 AWS 上部署托管集群

设置托管的 control plane 命令行界面后，hcp 并启用 local-cluster 作为托管集群，您可以通过完成以下步骤在 AWS 上部署托管集群。要部署私有集群，请参阅在 AWS 上部署私有集群。

1. 要查看每个变量的描述，请运行以下命令：

```
hcp create cluster aws --help
```

2. 验证您是否已登录到 hub 集群。

3. 运行以下命令来创建托管集群：

```
hcp create cluster aws \
  --name <hosted_cluster_name> \ 1
  --infra-id <infra_id> \ 2
  --base-domain <basedomain> \ 3
  --aws-creds <path_to_aws_creds> \ 4
  --pull-secret <path_to_pull_secret> \ 5
  --region <region> \ 6
  --generate-ssh \
  --node-pool-replicas <node_pool_replica_count> \ 7
  --namespace <hosted_cluster_namespace> 8
```

1. 指定托管集群的名称，例如。验证 & lt;hosted_cluster_name & gt; 和 <infra_id > 的值是否相同，否则集群可能无法在 Kubernetes operator 控制台的多集群引擎中正确显示。

2. 指定基础架构的名称，例如 clc-name-hs1。

3. 指定您的基域，如 dev09.red-chesterfield.com。

4.

指定 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

5

指定 pull secret 的路径，例如 /user/name/pullsecret。

6

指定 AWS 区域名称，如 us-east-1。

7

指定节点池副本数，例如 2。

8

指定托管集群的命名空间，例如 clusters。

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& gt; 参数，则会在您选择的命名空间中创建 HostedCluster 和 NodePool 自定义资源。

1. *您可以运行以下命令来检查托管集群的状态：*

```
oc get hostedclusters -n <hosted_cluster_namespace>
```

2. *您可以运行以下命令来检查节点池：*

```
oc get nodepools --namespace <hosted_cluster_namespace>
```

1.7.6.8. 在 AWS 的多个区中创建托管集群

输入以下命令创建集群，指定公共区的基域：

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
```

```
--aws-creds <path-to-aws-credentials> \ 5
--region <region> \ 6
--zones <zones> 7
```

1 指定托管集群的名称，例如。

2 指定节点池副本数，例如 2。

3 指定您的基域，如 example.com。

4 指定 pull secret 的路径，例如 /user/name/pullsecret。

5 指定 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

6 指定 AWS 区域名称，如 us-east-1。

7 指定 AWS 区域中的可用区，如 us-east-1a 和 us-east-1b。

对于每个指定区，会创建以下基础架构：

- 公共子网
- 专用子网
- NAT 网关

- 专用路由表（公共路由表在公共子网之间共享）

为每个区创建一个 NodePool 资源。节点池名称按区域名称后缀。区的专用子网在 spec.platform.aws.subnet.id 中设置。

1.7.6.8.1. 提供用于在 AWS 上创建托管集群的凭证

当使用 hcp create cluster aws 命令创建托管集群时，您需要提供具有为集群创建基础架构资源的 AWS 帐户凭证。基础架构资源示例包括 VPC、子网和 NAT 网关。您可以通过两种方式提供 AWS 凭证： 使用 --aws-creds 标志或使用 multicluster engine operator 中的 AWS 云供应商 secret。

1.7.6.8.1.1. 使用 --aws-creds 标志提供凭证

如果使用 --aws-creds 标志提供凭证，请将该标志与 AWS 凭证文件路径的值一起使用。

请参见以下示例：

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
--aws-creds <path-to-aws-credentials> \ 5
--region <region> 6
```

1. 指定托管集群的名称，例如。
2. 指定节点池副本数，例如 2。
3. 指定您的基域，如 example.com。
4. 指定 pull secret 的路径，例如 /user/name/pullsecret。

5

指定 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

6

指定 AWS 区域名称，如 us-east-1。

1.7.6.8.1.2. 使用 AWS 云供应商 secret 提供凭证

secret 包含 SSH 密钥、pull secret、基域和 AWS 凭证。因此，您可以使用 hcp create cluster aws 命令及 --secret-creds 标志来提供 AWS 凭证。请参见以下示例：

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--region <region> \ 2
--namespace <hosted-cluster-namespace> \ 3
--secret-creds <my-aws-cred> 4
```

1

指定托管集群的名称，例如。

2

指定 AWS 区域名称，如 us-east-1。

3

如果 secret 不在 default 集群 命名空间中，请指定托管集群命名空间。

4

指定 AWS secret 名称，如 my-aws-cred。

使用此 secret 时，以下标记变为可选。如果您使用 --secret-creds 标志指定这些标记，则这些标志优先于云供应商 secret 中的值：

- --aws-creds

- *--base-domain*
- *--pull-secret*
- *--ssh-key*

1. *要使用 {mce-shortF} 控制台创建 secret，从导航菜单中选择 Credentials 并按照控制台中的凭证创建步骤进行操作。*

2. *要在命令行中创建 secret，请输入以下命令：*

```
$ oc create secret generic <my-secret> -n <namespace> --from-literal=baseDomain=<your-basedomain> --from-literal=aws_access_key_id=<your-aws-access-key> --from-literal=aws_secret_access_key=<your-aws-secret-key> --from-literal=pullSecret='{"auths": {"cloud.openshift.com":{"auth":"<auth>", "email":"<your-email>"}, "quay.io":{"auth":"<auth>", "email":"<your-email>"} } }' --from-literal=ssh-publickey=<your-ssh-publickey> --from-literal=ssh-privatekey=<your-ssh-privatekey>
```

secret 具有以下格式：

```
apiVersion: v1
metadata:
  name: my-aws-cred (1)
  namespace: clusters (2)
type: Opaque
kind: Secret
stringData:
  ssh-publickey:         # Value
  ssh-privatekey:        # Value
  pullSecret:           # Value, required
  baseDomain:            # Value, required
  aws_secret_access_key: # Value, required
  aws_access_key_id:     # Value, required
```

1.7.6.8.2. 其他资源

有关在托管集群中安装 AWS Elastic File Service (EFS) CSI Driver Operator 的说明，请参阅使用安全令牌服务配置 AWS EFS CSI Driver Operator。

1.7.6.9. 在 ARM64 OpenShift Container Platform 集群上启用托管的 control plane （技术预览）

您可以启用 ARM64-hosted control plane，以便在管理集群环境中使用 OpenShift Container Platform ARM64 data plane。此功能仅适用于 AWS 上的托管 control plane。

1.7.6.9.1. 先决条件

您必须在 64 位 ARM 基础架构上安装 OpenShift Container Platform 集群。如需更多信息，请参阅创建 OpenShift 集群：AWS (ARM)。

1.7.6.9.2. 在 ARM64 OpenShift Container Platform 集群上运行托管集群

要在 ARM64 OpenShift Container Platform 集群上运行托管集群，请完成以下步骤：

1. 创建一个托管集群，以使用多架构发行镜像覆盖默认发行镜像。

   例如，通过托管的 control plane 命令行界面 hcp，输入以下命令，并将集群名称、节点池副本、基域、pull secret、AWS 凭证和区域替换为您的信息：

   ```
   hcp create cluster aws \
   --name $CLUSTER_NAME \
   --node-pool-replicas=$NODEPOOL_REPLICAS \
   --base-domain $BASE_DOMAIN \
   --pull-secret $PULL_SECRET \
   --aws-creds $AWS_CREDS \
   --region $REGION \
   --release-image quay.io/openshift-release-dev/ocp-release:4.13.0-rc.0-multi
   ```

   本例通过 **--node-pool-replicas** 标志添加默认的 NodePool 对象。

2. 将 64 位 x_86 NodePool 对象添加到托管集群。

   例如，通过托管的 control plane 命令行界面 hcp，输入以下命令，将集群名称、节点池名称和节点池副本替换为您的信息：

   ```
   hcp create nodepool aws \
   --cluster-name $CLUSTER_NAME \
   --name $NODEPOOL_NAME \
   --node-count=$NODEPOOL_REPLICAS
   ```

1.7.6.9.3. 在 AWS 托管的集群中创建 ARM NodePool 对象

您可以在同一托管的 control plane 的 64 位 ARM 和 AMD64 上调度应用程序工作负载(NodePool 对象)。为此，您可以在 NodePool 规格中定义 arch 字段，以设置 NodePool 对象所需的处理器架构。arch 字段的有效值如下：

- *arm64*
- *amd64*

如果没有为 arch 字段指定值，则默认使用 amd64 值。

要在 AWS 上的托管集群上创建 ARM NodePool 对象，请完成以下步骤：

1. *确保具有要使用的 HostedCluster 自定义资源的多架构镜像。您可以在每晚访问多架构 https://multi.ocp.releases.ci.openshift.org/。*

   一个多架构镜像类似以下示例：

```
% oc image info quay.io/openshift-release-dev/ocp-release-nightly@sha256:9b992c71f77501678c091e3dc77c7be066816562efe3d352be18128b8e8fce94 -a ~/pull-secrets.json

error: the image is a manifest list and contains multiple images - use --filter-by-os to select from:

  OS           DIGEST
  linux/amd64
sha256:c9dc4d07788ebc384a3d399a0e17f80b62a282b2513062a13ea702a811794a60
  linux/ppc64le
sha256:c59c99d6ff1fe7b85790e24166cfc448a3c2ac3ef3422fce3c7259e48d2c9aab
  linux/s390x
sha256:07fcd16d5bee95196479b1e6b5b3b8064dd5359dac75e3d81f0bd4be6b8fe208
  linux/arm64
sha256:1d93a6beccc83e2a4c56ecfc37e921fe73d8964247c1a3ec34c4d66f175d9b3d
```

2. *通过在托管 control plane 命令行界面(hcp)上输入以下命令来呈现 NodePool 对象：*

```
hcp create nodepool aws --cluster-name $CLUSTER_NAME --name $ARM64_NODEPOOL_NAME --node-count=$NODEPOOL_REPLICAS --render > arm_nodepool_spec.yml
```

该命令创建一个 YAML 文件，用于指定 NodePool 对象的 CPU 架构，如下例所示：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hypershift-arm-us-east-1a
  namespace: clusters
spec:
  arch: amd64
  clusterName: hypershift-arm
  management:
    autoRepair: false
    upgradeType: Replace
  nodeDrainTimeout: 0s
  platform:
    aws:
      instanceProfile: hypershift-arm-2m289-worker
      instanceType: m5.large
      rootVolume:
        size: 120
        type: gp3
      securityGroups:
      - id: sg-064ea63968d258493
      subnet:
        id: subnet-02c74cf1cf1e7413f
    type: AWS
  release:
    image: quay.io/openshift-release-dev/ocp-release-
nightly@sha256:390a33cebc940912a201a35ca03927ae5b058fbdae9626f7f4679786cab4f
b1c
  replicas: 3
status:
  replicas: 0
```

3. *输入以下命令修改 YAML 文件中的 arch 和 instanceType 值。在命令中，ARM 实例类型为 m6g.large，但任何 ARM 实例类型都可以正常工作：*

```
sed 's/arch: amd64/arch: arm64/g; s/instanceType: m5.large/instanceType: m6g.large/g'
arm_nodepool_spec.yml > temp.yml && mv temp.yml arm_nodepool_spec.yml
```

4. *输入以下命令将渲染的 YAML 文件应用到托管集群：*

```
oc apply -f arm_nodepool_spec.yml
```

1.7.6.10. 访问托管集群

您可以通过直接从资源获取 kubeconfig 文件和 kubeadmin 凭证，或使用 hcp 命令行界面生成 kubeconfig 文件来访问托管集群。

- *要通过直接从资源获取 kubeconfig 文件和凭证来访问托管集群，您需要熟悉托管 control plane 集群的访问 secret。secret 存储在托管的集群（托管）命名空间中。托管的集群（托管）命名空间包含托管的集群资源，托管的 control plane 命名空间是托管的 control plane 运行的位置。*

  secret 名称格式如下：

  - *kubeconfig secret: & lt;hosted-cluster-namespace>-<name>-admin-kubeconfig (clusters-hypershift-demo-admin-kubeconfig)*
  - *kubeadmin password secret: & lt;hosted-cluster-namespace>-<name>-kubeadmin-password (clusters-hypershift-demo-kubeadmin-password)*

  kubeconfig secret 包含 Base64 编码的 kubeconfig 字段，您可以使用以下命令解码并保存到文件中：

  ```
  oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
  ```

  kubeadmin 密码 secret 也采用 Base64 编码。您可以解码它，并使用密码登录到托管的集群的 API 服务器或控制台。

- *要使用 hcp CLI 访问托管集群来生成 kubeconfig 文件，请执行以下步骤：*

  1. *输入以下命令生成 kubeconfig 文件：*

     ```
     hcp create kubeconfig --namespace <hosted-cluster-namespace> --name <hosted-cluster-name> > <hosted-cluster-name>.kubeconfig
     ```

  2. *保存 kubeconfig 文件后，您可以输入以下示例命令来访问托管集群：*

     ```
     oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
     ```

1.7.6.10.1. 其他资源

访问托管集群后，您可以扩展节点池或为托管集群启用节点自动扩展。如需更多信息，请阅读以下主题：

- 扩展节点池
- 为托管集群启用节点自动扩展

要为托管集群配置节点性能优化，请查看以下主题：

- 在托管集群中配置节点性能优化
- 通过设置内核引导参数，对托管集群进行高级节点调整。

1.7.6.11. 在 AWS 上部署私有集群（技术预览）

设置托管的 control plane 命令行界面后，hcp 并启用 local-cluster 作为托管集群，您可以在 AWS 上部署托管集群或私有托管集群。要在 AWS 上部署公共托管集群，请参阅在 AWS 上部署托管集群。

默认情况下，托管的 control plane 客户机集群可以通过公共 DNS 和管理集群的默认路由器进行公开访问。

对于 AWS 上的私有集群，所有与客户机集群的通信都通过 AWS PrivateLink 进行。要为 AWS 上的私有集群配置托管的 control plane，请执行以下步骤。

重要：虽然公共集群可以在任何区域中创建，但只能在 --aws-private-region 指定的区域中创建私有集群。

- 先决条件
- 在 AWS 上创建私有集群
- 访问 AWS 上的私有托管集群

- *其他资源*

1.7.6.11.1. 先决条件

要为 AWS 启用私有托管集群，您必须首先启用 AWS PrivateLink。如需更多信息，请参阅启用 AWS PrivateLink。

1.7.6.11.2. 在 AWS 上创建私有集群

1. *输入以下命令来创建私有集群 IAM 策略文档：*

```
cat << EOF >> policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpcEndpointServiceConfiguration",
        "ec2:DescribeVpcEndpointServiceConfigurations",
        "ec2:DeleteVpcEndpointServiceConfigurations",
        "ec2:DescribeVpcEndpointServicePermissions",
        "ec2:ModifyVpcEndpointServicePermissions",
        "ec2:CreateTags",
        "elasticloadbalancing:DescribeLoadBalancers"
      ],
      "Resource": "\*"
    }
  ]
}
```

2. *输入以下命令在 AWS 中创建 IAM 策略：*

```
aws iam create-policy --policy-name=hypershift-operator-policy --policy-
document=file://policy.json
```

3. *输入以下命令来创建 hypershift-operator IAM 用户：*

```
aws iam create-user --user-name=hypershift-operator
```

4. *输入以下命令将策略附加到 hypershift-operator 用户，将 &lt ;policy-arn > 替换为您创建的策略的 ARN：*

-

```
aws iam attach-user-policy --user-name=hypershift-operator --policy-arn=<policy-arn>
```

5. 输入以下命令为用户创建 IAM 访问密钥：

```
aws iam create-access-key --user-name=hypershift-operator
```

6. 输入以下命令创建私有集群，根据需要将变量替换为您的值：

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
--aws-creds <path-to-aws-credentials> \ 5
--region <region> \ 6
--endpoint-access Private 7
```

1 1 指定托管集群的名称，例如。

2 2 指定节点池副本数，例如 3。

3 指定您的基域，如 example.com。

4 指定 pull secret 的路径，例如 /user/name/pullsecret。

5 指定 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

6 指定 AWS 区域名称，如 us-east-1。

7

定义集群是公共还是私有。

集群的 API 端点可以通过私有 DNS 区域访问：

- api.<hosted-cluster-name>.hypershift.local
- *.apps.<hosted-cluster-name>.hypershift.local

1.7.6.11.3. 访问 AWS 上的私有托管集群

您可以使用堡垒实例访问私有集群。

1. 输入以下命令启动堡垒实例：

```
hypershift create bastion aws --aws-creds=<aws-creds> --infra-id=<infra-id> --region=
<region> --ssh-key-file=<ssh-key>
```

将 <ssh-key > 替换为连接到堡垒的 SSH 公钥文件。SSH 公钥文件的默认位置为 ~/.ssh/id_rsa.pub。将 <aws-creds > 替换为 AWS 凭证文件的路径，例如 /user/name/.aws/credentials。

注： hypershift CLI 不适用于下载。使用以下命令，使用 hypershift 命名空间中存在的 HyperShift Operator pod 来提取它。将 <hypershift-operator-pod-name& gt; 替换为您的 HyperShift Operator pod 名称。

```
oc project hypershift
oc rsync <hypershift-operator-pod-name>:/usr/bin/hypershift-no-cgo .
mv hypershift-no-cgo hypershift
```

1. 输入以下命令在集群节点池中查找节点的专用 IP：

```
aws ec2 describe-instances --filter="Name=tag:kubernetes.io/cluster/<infra-
id>,Values=owned" | jq '.Reservations[] | .Instances[] | select(.PublicDnsName=="") |
.PrivateIpAddress'
```

2. 输入以下命令为集群创建 kubeconfig 文件，它可以复制到节点：

```
hcp create kubeconfig > <cluster-kubeconfig>
```

3. 输入以下命令使用 create bastion 命令打印的 IP 通过堡垒 SSH 到其中一个节点：

```
ssh -o ProxyCommand="ssh ec2-user@<bastion-ip> -W %h:%p" core@<node-ip>
```

4. 在 SSH shell 中，输入以下命令将 kubeconfig 文件内容复制到节点上的文件中：

```
mv <path-to-kubeconfig-file> <new-file-name>
```

5. 输入以下命令导出 kubeconfig 文件：

```
export KUBECONFIG=<path-to-kubeconfig-file>
```

6. 输入以下命令观察客户机集群状态：

```
oc get clusteroperators clusterversion
```

1.7.6.11.4. 其他资源

有关在 AWS 上部署公共托管集群的更多信息，请参阅在 AWS 上部署托管集群。

1.7.6.12. 为托管 control plane 管理 AWS 基础架构和 IAM 权限（技术预览）

当您在 AWS 上为 Red Hat OpenShift Container Platform 使用托管的 control plane 时，基础架构要求会因您的设置而异。

- 先决条件
- AWS 基础架构要求
- 身份和访问管理权限

- 单独创建 AWS 基础架构和 IAM 资源

1.7.6.12.1. 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，请参阅在 AWS 上配置托管的 control plane 集群（技术预览）。

1.7.6.12.2. AWS 基础架构要求

当您在 AWS 上使用托管的 control plane 时，基础架构要求适合以下类别：

- 在任意 AWS 帐户中为 HyperShift Operator 预必需和非受管基础架构
- 托管的集群 AWS 帐户中的 Prerequired 和 unmanaged 基础架构
- 在管理 AWS 帐户中托管 control planes 管理的基础架构
- 在托管的集群 AWS 帐户中托管 control plane 管理的基础架构
- 托管集群 AWS 帐户中的 Kubernetes 管理的基础架构

Prerequired 表示托管 control plane 需要 AWS 基础架构才能正常工作。Unmanaged 意味着没有 Operator 或控制器为您创建基础架构。以下小节包含有关创建 AWS 资源的详细信息。

1.7.6.12.2.1. 在任意 AWS 帐户中为 HyperShift Operator 预必需和非受管基础架构

任意 AWS 帐户取决于托管的 control plane 服务的供应商。

在自我管理的托管 control plane 中，集群服务提供商控制 AWS 帐户。集群服务提供商 是托管集群 control plane 的管理员，它负责正常运行时间。在托管的托管 control plane 中，AWS 帐户属于红帽。

在 HyperShift Operator 的预必需且非受管基础架构中，以下基础架构要求适用于管理集群 AWS 帐户：

- 一个 S3 Bucket
  - OpenID Connect(OIDC)
- 路由 53 托管区域
  - 托管托管集群的专用和公共条目的域

1.7.6.12.2.2. 托管的集群 AWS 帐户中的 Prerequired 和 unmanaged 基础架构

当您的基础架构在托管集群 AWS 帐户中预必需且非受管时，所有访问模式的基础架构要求如下：

- 一个 VPC
- 一个 DHCP 选项
- 两个子网
  - 是内部数据平面子网的专用子网
  - 允许从数据平面访问互联网的公共子网
- 一个互联网网关
- 一个弹性 IP
- 一个 NAT 网关

- 一个安全组(worker 节点)
- 两个路由表（一个私有和一个公共）
- 两个 Route 53 托管区域
- 有足够的配额用于以下项目：
  - 公共托管集群的一个 Ingress 服务负载均衡器
  - 私有托管集群的一个私有链接端点

注： 要使私有链路网络正常工作，托管集群 AWS 帐户中的端点区域必须与管理集群 AWS 帐户中的服务端点解析的实例区匹配。在 AWS 中，区域名称是别名，如 us-east-2b，它们不一定映射到不同帐户中的同一区域。因此，为了使私有链接正常工作，管理集群必须在其区域的所有区域中都有子网或 worker。

1.7.6.12.2.3. 在管理 AWS 帐户中托管 control planes 管理的基础架构

当您的基础架构由管理 AWS 帐户中的托管 control plane 管理时，基础架构要求因您的集群是公共、私有还是组合而不同。

对于具有公共集群的帐户，基础架构要求如下：

- 网络负载均衡器：负载均衡器 Kube API 服务器
  - Kubernetes 创建一个安全组
- 卷
  - 对于 etcd （取决于高可用性，一个或多个三个）

- 对于 OVN-Kube

对于带有私有集群的帐户，基础架构要求如下：

- 网络负载均衡器：负载均衡器私有路由器
- 端点服务（专用链接）

对于具有公共和私有集群的帐户，基础架构要求如下：

- 网络负载均衡器：负载均衡器公共路由器
- 网络负载均衡器：负载均衡器私有路由器
- 端点服务（专用链接）
- 卷：
  - 对于 etcd （取决于高可用性，一个或多个三个）
  - 对于 OVN-Kube

1.7.6.12.2.4. 在托管的集群 AWS 帐户中托管 control plane 管理的基础架构

当您的基础架构由托管的集群 AWS 帐户中的托管 control plane 管理时，基础架构要求会有所不同，具体取决于您的集群是公共、私有还是组合。

对于具有公共集群的帐户，基础架构要求如下：

- 节点池必须具有定义 Role 和 RolePolicy 的 EC2 实例。

对于带有私有集群的帐户，基础架构要求如下：

- 每个可用区的一个私有链接端点
- 用于节点池的 EC2 实例

对于具有公共和私有集群的帐户，基础架构要求如下：

- 每个可用区的一个私有链接端点
- 用于节点池的 EC2 实例

1.7.6.12.2.5. 托管集群 AWS 帐户中的 Kubernetes 管理的基础架构

当 Kubernetes 在托管集群 AWS 帐户中管理您的基础架构时，基础架构要求如下：

- 默认入口的网络负载均衡器
- registry 的 S3 存储桶

1.7.6.12.3. Identity and Access Management (IAM)权限

在托管 control plane 的上下文中，使用者负责创建 Amazon 资源名称(ARN)角色。使用者 是生成权限文件的自动过程。消费者可能是命令行界面或 OpenShift Cluster Manager。托管 control plane 尝试使粒度遵守最低特权组件的原则，这意味着每个组件都使用自己的角色来运行或创建 AWS 对象，并且角色仅限于产品正常正常工作所需的角色。

有关命令行界面如何创建 ARN 角色的示例，请参阅"单独创建 AWS 基础架构和 IAM 资源"。

托管的集群作为输入接收 ARN 角色，消费者为每个组件创建一个 AWS 权限配置。因此，组件可以

通过 STS 和预配置的 OIDC IDP 进行身份验证。

以下角色由 control plane 上运行的托管 control plane 的一些组件使用，并在 data plane 上运行：

- *controlPlaneOperatorARN*
- *imageRegistryARN*
- *ingressARN*
- *kubeCloudControllerARN*
- *nodePoolManagementARN*
- *storageARN*
- *networkARN*

以下示例显示了对来自托管集群的 IAM 角色的引用：

```
...
endpointAccess: Public
  region: us-east-2
  resourceTags:
  - key: kubernetes.io/cluster/example-cluster-bz4j5
    value: owned
rolesRef:
    controlPlaneOperatorARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-control-plane-
operator
    imageRegistryARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-openshift-image-
registry
    ingressARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-openshift-ingress
    kubeCloudControllerARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-cloud-controller
    networkARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-cloud-network-config-
controller
    nodePoolManagementARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-node-pool
```

```
    storageARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-aws-ebs-csi-driver-controller
type: AWS
...
```

托管 control plane 使用的角色在以下示例中显示：

- ***ingressARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers",
                "tag:GetResources",
                "route53:ListHostedZones"
            ],
            "Resource": "\*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53:ChangeResourceRecordSets"
            ],
            "Resource": [
                "arn:aws:route53:::PUBLIC_ZONE_ID",
                "arn:aws:route53:::PRIVATE_ZONE_ID"
            ]
        }
    ]
}
```

- ***imageRegistryARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:PutBucketTagging",
                "s3:GetBucketTagging",
                "s3:PutBucketPublicAccessBlock",
                "s3:GetBucketPublicAccessBlock",
                "s3:PutEncryptionConfiguration",
                "s3:GetEncryptionConfiguration",
                "s3:PutLifecycleConfiguration",
```

```
                "s3:GetLifecycleConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:ListBucketMultipartUploads",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "\*"
        }
    ]
}
```

- ***storageARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteSnapshot",
                "ec2:DeleteTags",
                "ec2:DeleteVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeSnapshots",
                "ec2:DescribeTags",
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications",
                "ec2:DetachVolume",
                "ec2:ModifyVolume"
            ],
            "Resource": "\*"
        }
    ]
}
```

- ***networkARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:UnassignIpv6Addresses",
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": "\*"
        }
    ]
}
```

- ***kubeCloudControllerARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeImages",
                "ec2:DescribeRegions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyVolume",
                "ec2:AttachVolume",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateRoute",
                "ec2:DeleteRoute",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DetachVolume",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:AttachLoadBalancerToSubnets",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerPolicy",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancerListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
```

```
            "elasticloadbalancing:DetachLoadBalancerFromSubnets",
            "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
            "elasticloadbalancing:ModifyLoadBalancerAttributes",
            "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
            "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
            "elasticloadbalancing:AddTags",
            "elasticloadbalancing:CreateListener",
            "elasticloadbalancing:CreateTargetGroup",
            "elasticloadbalancing:DeleteListener",
            "elasticloadbalancing:DeleteTargetGroup",
            "elasticloadbalancing:DescribeListeners",
            "elasticloadbalancing:DescribeLoadBalancerPolicies",
            "elasticloadbalancing:DescribeTargetGroups",
            "elasticloadbalancing:DescribeTargetHealth",
            "elasticloadbalancing:ModifyListener",
            "elasticloadbalancing:ModifyTargetGroup",
            "elasticloadbalancing:RegisterTargets",
            "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
            "iam:CreateServiceLinkedRole",
            "kms:DescribeKey"
        ],
        "Resource": [
            "\*"
        ],
        "Effect": "Allow"
    }
  ]
}
```

- ***nodePoolManagementARN***

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ec2:AllocateAddress",
        "ec2:AssociateRouteTable",
        "ec2:AttachInternetGateway",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateInternetGateway",
        "ec2:CreateNatGateway",
        "ec2:CreateRoute",
        "ec2:CreateRouteTable",
        "ec2:CreateSecurityGroup",
        "ec2:CreateSubnet",
        "ec2:CreateTags",
        "ec2:DeleteInternetGateway",
        "ec2:DeleteNatGateway",
        "ec2:DeleteRouteTable",
        "ec2:DeleteSecurityGroup",
        "ec2:DeleteSubnet",
        "ec2:DeleteTags",
        "ec2:DescribeAccountAttributes",
```

```
                "ec2:DescribeAddresses",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeInternetGateways",
                "ec2:DescribeNatGateways",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeNetworkInterfaceAttribute",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeVolumes",
                "ec2:DetachInternetGateway",
                "ec2:DisassociateRouteTable",
                "ec2:DisassociateAddress",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:ModifySubnetAttribute",
                "ec2:ReleaseAddress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RunInstances",
                "ec2:TerminateInstances",
                "tag:GetResources",
                "ec2:CreateLaunchTemplate",
                "ec2:CreateLaunchTemplateVersion",
                "ec2:DescribeLaunchTemplates",
                "ec2:DescribeLaunchTemplateVersions",
                "ec2:DeleteLaunchTemplate",
                "ec2:DeleteLaunchTemplateVersions"
            ],
            "Resource": [
                "\*"
            ],
            "Effect": "Allow"
        },
        {
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
                }
            },
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:*:iam::*:role/aws-service-
role/elasticloadbalancing.amazonaws.com/AWSServiceRoleForElasticLoadBalancing"
            ],
            "Effect": "Allow"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
```

```
            "Resource": [
                "arn:*:iam::*:role/*-worker-role"
            ],
            "Effect": "Allow"
        }
    ]
}
```

- ***controlPlaneOperatorARN***

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateVpcEndpoint",
                "ec2:DescribeVpcEndpoints",
                "ec2:ModifyVpcEndpoint",
                "ec2:DeleteVpcEndpoints",
                "ec2:CreateTags",
                "route53:ListHostedZones"
            ],
            "Resource": "\*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53:ChangeResourceRecordSets",
                "route53:ListResourceRecordSets"
            ],
            "Resource": "arn:aws:route53:::%s"
        }
    ]
}
```

1.7.6.12.4. 单独创建 AWS 基础架构和 IAM 资源

*默认情况下，**hcp create cluster aws** 命令创建带有托管集群的云基础架构并应用它。您可以单独创建云基础架构部分，以便 **hcp create cluster aws** 命令只能用于创建集群，或者呈现它，以便在应用它前修改它。*

*要单独创建云基础架构部分，您需要创建 **AWS** 基础架构，创建 **AWS Identity and Access (IAM)**资源，并创建集群。*

1.7.6.12.4.1. 创建 AWS 基础架构

运行以下命令来创建 AWS 基础架构：

```
hypershift create infra aws --name CLUSTER_NAME \ 1
  --aws-creds AWS_CREDENTIALS_FILE \ 2
  --base-domain BASEDOMAIN \ 3
  --infra-id INFRA_ID \ 4
  --region REGION \ 5
  --output-file OUTPUT_INFRA_FILE 6
```

1

将 CLUSTER_NAME 替换为您要创建的托管集群的名称。这个值用于为集群创建 Route 53 私有托管区。

2

将 AWS_CREDENTIALS_FILE 替换为 AWS 凭证文件的名称，该文件具有为集群创建基础架构资源的权限，如 VPC、子网和 NAT 网关。这个值必须与 worker 所在的客户机集群的 AWS 帐户对应。

3

将 BASEDOMAIN 替换为您计划用于托管集群 Ingress 的基域的名称。这个值必须与您可以在其中创建记录的 Route 53 公共区对应。

4

将 INFRA_ID 替换为使用标签标识基础架构的唯一名称。这个值供 Kubernetes 中的云控制器管理器和集群 API 管理器使用，以识别集群的基础架构。通常，这个值是集群的名称 (CLUSTER_NAME)，后缀附加到其中。

5

使用您要为集群创建基础架构的区域替换 REGION。

6

将 OUTPUT_INFRA_FILE 替换为您要以 JSON 格式存储基础架构的 ID 的文件名称。您可以使用此文件作为 hcp create cluster aws 命令的输入，来填充 HostedCluster 和 NodePool resouces 中的字段。

注： hypershift CLI 不适用于下载。使用以下命令，使用 hypershift 命名空间中存在的 HyperShift Operator pod 来提取它。将 <hypershift-operator-pod-name& gt; 替换为您的 HyperShift Operator pod 名称。

+

```
oc project hypershift
oc rsync <hypershift-operator-pod-name>:/usr/bin/hypershift-no-cgo .
mv hypershift-no-cgo hypershift
```

输入以下命令后，会创建以下资源：

- 一个 VPC
- 一个 DHCP 选项
- 一个专用子网
- 一个公共子网
- 一个互联网网关
- 一个 NAT 网关
- 一个用于 worker 节点的安全组
- 两个路由表：1 个私有和 1 个公共
- 两个私有托管区：1 用于集群 Ingress，1 代表 PrivateLink，如果您创建一个私有集群

所有这些资源都包含 kubernetes.io/cluster/INFRA_ID=owned 标签，其中 INFRA_ID 是您在命令中指定的值。

1.7.6.12.4.2. 创建 AWS IAM 资源

运行以下命令来创建 AWS IAM 资源：

```
hypershift create iam aws --infra-id INFRA_ID \ 1
  --aws-creds AWS_CREDENTIALS_FILE \ 2
  --oidc-storage-provider-s3-bucket-name OIDC_BUCKET_NAME \ 3
  --oidc-storage-provider-s3-region OIDC_BUCKET_REGION \ 4
  --region REGION \ 5
  --public-zone-id PUBLIC_ZONE_ID \ 6
  --private-zone-id PRIVATE_ZONE_ID \ 7
  --local-zone-id LOCAL_ZONE_ID \ 8
  --output-file OUTPUT_IAM_FILE 9
```

1

将 INFRA_ID 替换为您在 create infra aws 命令中指定的相同 ID。这个值标识与托管集群关联的 IAM 资源。

2

将 AWS_CREDENTIALS_FILE 替换为具有创建 IAM 资源权限的 AWS 凭证文件的名称，如角色。此文件不需要是您为创建基础架构指定的相同凭据文件，但它必须与相同的 AWS 帐户对应。

3

将 OIDC_BUCKET_NAME 替换为存储 OIDC 文档的存储桶的名称。此存储桶是作为安装托管 control plane 的先决条件创建的。bucket 的名称用于为这个命令创建的 OIDC 供应商构建 URL。

4

将 OIDC_BUCKET_REGION 替换为 OIDC 存储桶所在的区域。

5

将 REGION 替换为集群基础架构所在的区域。这个值用于为属于托管集群的机器创建 worker 实例配置集。

6

将 PUBLIC_ZONE_ID 替换为客户机集群公共区的 ID。这个值用于为 Ingress Operator 创建策略。您可以在 create infra aws 命令生成的 OUTPUT_INFRA_FILE 中找到这个值。

7

将 PRIVATE_ZONE_ID 替换为客户机集群的私有区 ID。这个值用于为 Ingress Operator 创建策略。您可以在 create infra aws 命令生成的 OUTPUT_INFRA_FILE 中找到这个值。

8

在创建私有集群时，将 LOCAL_ZONE_ID 替换为客户机集群的本地区 ID。这个值用于为 Control Plane Operator 创建策略，以便它可以管理 PrivateLink 端点的记录。您可以在 create infra aws 命令生成的 OUTPUT_INFRA_FILE 中找到这个值。

9

将 OUTPUT_IAM_FILE 替换为计划以 JSON 格式存储 IAM 资源的 ID 的文件名称。然后，您可以使用此文件作为 hcp create cluster aws 命令的输入，来填充 HostedCluster 和 NodePool 资源中的字段。

输入以下命令后，会创建以下资源：

- 一个 OIDC 供应商，需要启用 STS 验证
- 七家角色，每个组件都分开与提供程序交互，如 Kubernetes 控制器管理器、集群 API 供应商和 registry
- 一个实例配置集，它是分配给集群中的所有 worker 实例的配置集

1.7.6.12.4.3. 创建集群

运行以下命令来创建集群：

```
hcp create cluster aws \
  --infra-id INFRA_ID \ 1
  --name CLUSTER_NAME \ 2
  --aws-creds AWS_CREDENTIALS \ 3
  --pull-secret PULL_SECRET_FILE \ 4
  --generate-ssh \ 5
  --node-pool-replicas 3
```

1

将 INFRA_ID 替换为您在 create infra aws 命令中指定的相同 ID。这个值标识与托管集群关联的 IAM 资源。

2

将 CLUSTER_NAME 替换为您在 create infra aws 命令中指定的相同名称。

3 将 AWS_CREDENTIALS 替换为您在 create infra aws 命令中指定的相同值。

4 将 PULL_SECRET_FILE 替换为包含有效 OpenShift Container Platform pull secret 的文件的名称。

5 --generate-ssh 标志是可选的，但在您需要 SSH 到 worker 时包括。为您生成 SSH 密钥，并作为 secret 存储在与托管集群相同的命名空间中。

您还可以在命令中添加 --render 标志，并将输出重定向到可编辑资源的文件，然后再将它们应用到集群。

运行命令后，以下资源会应用到集群：

- 一个命名空间
- 带有 pull secret 的 secret
- A HostedCluster
- A NodePool
- control plane 组件的三个 AWS STS secret
- 如果您指定了 --generate-ssh 标志，则一个 SSH 密钥 secret。

1.7.6.13. 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，删除 multicluster engine operator 上的受管集群资源：

```
oc delete managedcluster <managed_cluster_name>
```

其中 <managed_cluster_name > 是受管集群的名称。

2. 运行以下命令来删除托管集群及其后端资源：

```
hcp destroy cluster aws --name <hosted_cluster_name> --infra-id <infra_id> --aws-creds <path_to_aws_creds> --base-domain <basedomain>
```

根据需要替换名称。

1.7.7. 在裸机上配置托管的 control plane 集群

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。托管集群也称为 管理集群。

注： 管理集群 与受管集群 不同。受管集群是一个 hub 集群管理的集群。

托管的 control plane 功能默认启用。

multicluster engine operator 2.5 仅支持默认的 local-cluster，它是一个托管的 hub 集群，以及 hub 集群作为托管集群。在 Red Hat Advanced Cluster Management 2.10 中，您可以使用受管 hub 集群（也称为 local-cluster ），作为托管集群。

托管的集群 是一个 OpenShift Container Platform 集群，其 API 端点和托管在托管集群中的 control plane。托管的集群包括控制平面和它的对应的数据平面。您可以使用 multicluster engine operator 控制台或托管的 control plane 命令行界面 hcp 创建托管集群。托管的集群自动导入为受管集群。如果要禁用此自动导入功能，请参阅禁用将托管集群的自动导入到 multicluster engine operator。

重要：

- 在托管 control plane 的同一平台上运行 hub 集群和 worker。
- 每个托管集群都必须具有集群范围的唯一名称。托管的集群名称不能与任何现有受管集群相同，以便 multicluster engine operator 管理它。
- 不要使用 集群 作为托管的集群名称。
- 无法在 multicluster engine operator 受管集群的命名空间中创建托管集群。
- 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。有关中央基础架构管理服务简介，请参阅 启用中央基础架构管理服务。
- 所有裸机主机都需要使用中央基础架构管理提供的发现镜像 ISO 手动引导。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。
- 当您根据节点池扩展副本时，会创建一个机器。对于每台机器，Cluster API 供应商找到并安装满足节点池规格中指定的要求的 Agent。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。在重复使用代理前，您必须使用 Discovery 镜像重启它们。
- 当您为托管 control plane 配置存储时，请考虑推荐的 etcd 实践。要确保您满足延迟要求，请将快速存储设备专用于每个 control-plane 节点上运行的所有托管的 control plane etcd 实例。您可以使用 LVM 存储为托管 etcd pod 配置本地存储类。如需更多信息，请参阅 OpenShift Container Platform 文档中的使用逻辑卷管理器存储 的建议 etcd 实践 和持久性存储。

1.7.7.1. 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。

- 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
```

- 您必须将 topology.kubernetes.io/zone 标签添加到管理集群中的裸机主机。否则，所有托管的 control plane pod 都会调度到单个节点上，从而导致单点故障。

- 您需要启用中央基础架构管理。如需更多信息，请参阅启用中央基础架构管理服务。

- 您需要安装托管的 control plane 命令行界面。

1.7.7.2. 裸机防火墙、端口和服务要求

您必须满足防火墙、端口和服务要求，以便端口可以在管理集群、control plane 和托管的集群间进行通信。

注：服务在其默认端口上运行。但是，如果您使用 NodePort 发布策略，服务在由 NodePort 服务分配的端口上运行。

使用防火墙规则、安全组或其他访问控制来限制对所需源的访问。除非必要，否则请避免公开端口。对于生产环境部署，请使用负载均衡器通过单个 IP 地址简化访问。

托管的 control plane 在裸机上公开以下服务：

- APIServer
  - APIServer 服务默认在端口 6443 上运行，并且需要入口访问才能在 control plane 组

件之间的通信。

- 如果使用 MetalLB 负载均衡，允许对用于负载均衡器 IP 地址的 IP 范围进行入口访问。

- OAuthServer
  - 当您使用路由和入口来公开服务时，OAuthServer 服务默认在端口 443 上运行。
  - 如果使用 NodePort 发布策略，请为 OAuthServer 服务使用防火墙规则。

- Konnectivity
  - 当您使用路由和入口来公开服务时，Konnectivity 服务默认在端口 443 上运行。
  - Konnectivity 代理（建立反向隧道来允许托管的集群中的双向通信）需要在端口 6443 上出口访问集群 API 服务器地址。通过该出口访问，代理可以访问 APIServer 服务。
  - 如果集群 API 服务器地址是一个内部 IP 地址，允许从工作负载子网访问端口 6443 上的 IP 地址。
  - 如果地址是一个外部 IP 地址，允许将端口 6443 上的出口从节点传输到该外部 IP 地址。

- Ignition
  - 当使用路由和入口来公开服务时，Ignition 服务默认在端口 443 上运行。
  - 如果使用 NodePort 发布策略，请为 Ignition 服务使用防火墙规则。

在裸机上不需要以下服务：

- OVNSbDb
- OIDC

1.7.7.3. 裸机基础架构要求

Agent 平台不创建任何基础架构，但它对基础架构有以下要求：

- Agent ：代理代表使用发现镜像引导的主机，并准备好作为 OpenShift Container Platform 节点置备。
- DNS ：API 和入口端点必须可以被路由。

有关裸机上托管 control plane 的其他资源，请参阅以下文档：

- 要了解 etcd 和 LVM 存储建议，请参阅使用逻辑卷管理器存储 的建议 etcd 实践 和持久性存储。
- 要在断开连接的环境中在裸机上配置托管的 control plane，请参阅在断开连接的环境中 配置托管的 control plane。
- 要禁用托管的 control plane 功能，或者已经禁用了它并希望手动启用它，请参阅 启用或禁用托管的 control plane 功能。
- 要通过运行 Red Hat Ansible Automation Platform 作业来管理托管集群，请参阅配置 Ansible Automation Platform 作业以便在托管集群中运行。
- 要部署 SR-IOV Operator，请参阅为托管的 control plane 部署 SR-IOV Operator。
- 如果要禁用自动导入功能，请参阅 将 托管集群的自动导入到 multicluster engine

operator。

1.7.7.4. 在裸机上配置 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入口 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
```

- 如果您要为 IPv6 网络上的断开连接的环境配置 DNS，请查看以下 DNS 配置示例：

```
api.example.krnl.es.    IN A 2620:52:0:1306::5
api.example.krnl.es.    IN A 2620:52:0:1306::6
api.example.krnl.es.    IN A 2620:52:0:1306::7
api-int.example.krnl.es.    IN A 2620:52:0:1306::5
api-int.example.krnl.es.    IN A 2620:52:0:1306::6
api-int.example.krnl.es.    IN A 2620:52:0:1306::7
`*`.apps.example.krnl.es. IN A 2620:52:0:1306::10
```

- 如果您要为双栈网络上的断开连接的环境配置 DNS，请确保同时包含 IPv4 和 IPv6 的 DNS 条目。请参见以下 DNS 配置示例：

```
host-record=api-int.hub-dual.dns.base.domain.name,192.168.126.10
host-record=api.hub-dual.dns.base.domain.name,192.168.126.10
address=/apps.hub-dual.dns.base.domain.name/192.168.126.11
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,192.168.126.20
dhcp-host=aa:aa:aa:aa:10:02,ocp-master-1,192.168.126.21
dhcp-host=aa:aa:aa:aa:10:03,ocp-master-2,192.168.126.22
dhcp-host=aa:aa:aa:aa:10:06,ocp-installer,192.168.126.25
dhcp-host=aa:aa:aa:aa:10:07,ocp-bootstrap,192.168.126.26

host-record=api-int.hub-dual.dns.base.domain.name,2620:52:0:1306::2
```

```
host-record=api.hub-dual.dns.base.domain.name,2620:52:0:1306::2
address=/apps.hub-dual.dns.base.domain.name/2620:52:0:1306::3
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,[2620:52:0:1306::5]
dhcp-host=aa:aa:aa:aa:10:02,ocp-master-1,[2620:52:0:1306::6]
dhcp-host=aa:aa:aa:aa:10:03,ocp-master-2,[2620:52:0:1306::7]
dhcp-host=aa:aa:aa:aa:10:06,ocp-installer,[2620:52:0:1306::8]
dhcp-host=aa:aa:aa:aa:10:07,ocp-bootstrap,[2620:52:0:1306::9]
```

接下来，在裸机上托管 control plane 创建主机清单。

1.7.7.5. 在裸机上创建托管集群

您可以在裸机上创建托管集群，或导入一个集群。有关导入托管集群的步骤，请参阅 导入托管集群。

1. 输入以下命令来创建托管的 control plane 命名空间：

```
oc create ns <hosted_cluster_namespace>-<hosted_cluster_name>
```

将 <hosted_cluster_namespace > 替换为托管的集群命名空间名称，例如 集群。将 <hosted_cluster_name > 替换为您的托管的集群名称。

2. 验证您是否为集群配置了默认存储类。否则，您可能会看到待处理的 PVC。运行以下命令：

```
hcp create cluster agent \
  --name=<hosted_cluster_name> \ 1
  --pull-secret=<path_to_pull_secret> \ 2
  --agent-namespace=<hosted_control_plane_namespace> \ 3
  --base-domain=<basedomain> \ 4
  --api-server-address=api.<hosted_cluster_name>.<basedomain> \
  --etcd-storage-class=<etcd_storage_class> \ 5
  --ssh-key  <path_to_ssh_public_key> \ 6
  --namespace <hosted_cluster_namespace> \ 7
  --control-plane-availability-policy SingleReplica \
  --release-image=quay.io/openshift-release-dev/ocp-release:<ocp_release_image>
8
```

1 指定托管集群的名称，例如。

2

指定 pull secret 的路径，例如 /user/name/pullsecret。

3

指定托管的 control plane 命名空间，如 cluster -example。使用 oc get agent -n <hosted_control_plane_namespace> 命令确保代理在这个命名空间中 可用。

4

指定您的基域，如 krnl.es。

5

指定 etcd 存储类名称，例如 lvm-storageclass。

6

指定 SSH 公钥的路径。默认文件路径为 ~/.ssh/id_rsa.pub。

7

指定托管集群的命名空间。

8

指定您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。如果您使用断开连接的环境，将 < ocp_release_image> 替换为摘要镜像。要提取 OpenShift Container Platform 发行镜像摘要，请参阅 提取 OpenShift Container Platform 发行镜像摘要。

3. *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
```

请参见以下示例输出：

```
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
```

1.7.7.5.1. 使用控制台在裸机上创建托管集群

1. 打开 OpenShift Container Platform Web 控制台，并通过输入管理员凭证登录。有关打开控制台的说明，请参阅 OpenShift Container Platform 文档中的 访问 Web 控制台。

2. 在控制台标头中，确保选择了 All Clusters。

3. 点 Infrastructure > Clusters。

4. 点 Create cluster > Host inventory > Hosted control plane。

   此时会显示 Create cluster 页面。

5. 在 Create cluster 页面中，按照提示输入集群、节点池、网络和自动化的详情。

   注： 当您输入集群详情时，您可能会发现以下有用提示：

   - 如果要使用预定义的值来自动填充控制台中的字段，您可以创建主机清单凭证。如需更多信息，请参阅 为内部环境创建凭证。
   - 在 Cluster details 页面中，pull secret 是用于访问 OpenShift Container Platform 资源的 OpenShift Container Platform pull secret。如果选择了主机清单凭证，则会自动填充 pull secret。
   - 在 Node pool 页面中，命名空间包含节点池的主机。如果使用控制台创建主机清单，控制台会创建一个专用命名空间。
   - 在 Networking 页面中，您可以选择 API 服务器发布策略。托管集群的 API 服务器可以通过使用现有的负载均衡器或 NodePort 类型的服务公开。必须存在 api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN} 设置的 DNS 条目，该设置指向可访问 API 服务器的目的地。此条目可以是指向管理集群中的一个节点的记录，或指向将传入的流量重定向到 Ingress pod 的负载均衡器的记录。

6. 检查您的条目并点 Create。

   此时会显示 Hosted 集群 视图。

7. 在 Hosted 集群视图中监控托管集群的部署。

8. 如果没有看到有关托管集群的信息，请确保选择了 All Clusters，然后点集群名称。

9. 等待 control plane 组件就绪。这个过程可能需要几分钟时间。

10. 要查看节点池状态，可滚动到 NodePool 部分。安装节点的过程大约需要 10 分钟。您还可以单击 Nodes 来确认节点是否加入托管集群。

1.7.7.5.2. 使用镜像 registry 在裸机上创建托管集群

您可以通过在 hcp create cluster 命令中指定 --image-content-sources 标志，使用镜像 registry 在裸机上创建托管集群。完成以下步骤：

1. 创建 YAML 文件以定义镜像内容源策略(ICSP)。请参见以下示例：

```
- mirrors:
  - brew.registry.redhat.io
  source: registry.redhat.io
- mirrors:
  - brew.registry.redhat.io
  source: registry.stage.redhat.io
- mirrors:
  - brew.registry.redhat.io
  source: registry-proxy.engineering.redhat.com
```

2. 将文件保存为 icsp.yaml。此文件包含您的镜像 registry。

3. 要使用您的镜像 registry 创建托管集群，请运行以下命令：

```
hcp create cluster agent \
    --name=<hosted_cluster_name> \ 1
    --pull-secret=<path_to_pull_secret> \ 2
    --agent-namespace=<hosted_control_plane_namespace> \ 3
    --base-domain=<basedomain> \ 4
    --api-server-address=api.<hosted_cluster_name>.<basedomain> \
    --image-content-sources icsp.yaml  \ 5
    --ssh-key  <path_to_ssh_key> \ 6
```

```
--namespace <hosted_cluster_namespace> \ 7
--release-image=quay.io/openshift-release-dev/ocp-release:<ocp_release_image>
8
```

1 指定托管集群的名称，例如。

2 指定 pull secret 的路径，例如 /user/name/pullsecret。

3 指定托管的 control plane 命名空间，如 cluster -example。使用 oc get agent -n <hosted-control-plane-namespace> 命令来确保代理在此命名空间中 可用。

4 指定您的基域，如 krnl.es。

5 指定定义 ICSP 和您的镜像 registry 的 icsp.yaml 文件。

6 指定 SSH 公钥的路径。默认文件路径为 ~/.ssh/id_rsa.pub。

7 指定托管集群的命名空间。

8 指定您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。如果您使用断开连接的环境，将 < ocp_release_image> 替换为摘要镜像。要提取 OpenShift Container Platform 发行镜像摘要，请参阅 提取 OpenShift Container Platform 发行镜像摘要。

1.7.7.5.3. 其他资源

- 要创建在使用控制台创建托管集群时可以重复使用 的凭证，请参阅为内部环境创建凭证。

- 要导入托管集群，请参阅 手动导入托管的 control plane 集群。
- 要访问托管集群，请参阅 访问托管集群。
- 要使用 Discovery Image 将主机添加到主机清单中，请参阅使用 Discovery Image 将主机添加到主机清单中。
- 要提取 OpenShift Container Platform 发行镜像摘要，请参阅 提取 OpenShift Container Platform 发行镜像摘要。

1.7.7.6. 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 <hosted-control-plane-namespace> secret/admin-kubeconfig --to=- >
kubeconfig-<hosted-cluster-name>
```

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-<hosted-cluster-name>
```

请参见以下示例输出：

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
console                              4.10.26   True        False         False      2m38s
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
```

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-<hosted-cluster-name>
```

请参见以下示例输出：

```
NAMESPACE                                 NAME                                              READY
STATUS             RESTARTS       AGE
kube-system                               konnectivity-agent-khlqv                          0/1
Running            0              3m52s
openshift-cluster-node-tuning-operator    tuned-dhw5p                                       1/1
Running            0              109s
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-console                         console-5cbf6c7969-6gk6z                          1/1
Running            0              119s
openshift-console                         downloads-7bcd756565-6wj5j                        1/1
Running            0              4m3s
openshift-dns-operator                    dns-operator-77d755cd8c-xjfbn                     2/2
Running            0              21m
openshift-dns                             dns-default-kfqnh                                 2/2
Running            0              113s
```

1.7.7.7. 为托管集群扩展 NodePool 对象

您可以通过将节点添加到托管集群来扩展 NodePool 对象。

1. 将 NodePool 对象扩展到两个节点：

```
oc -n <hosted-cluster-namespace> scale nodepool <nodepool-name> --replicas 2
```

Cluster API 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。代理按以下顺序传递状态：

- binding
- 发现
- insufficient
- 安装

- ***installing-in-progress***
- ***added-to-existing-cluster***

2. ***输入以下命令：***

```
oc -n <hosted-control-plane-namespace> get agent
```

请参见以下示例输出：

```
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
```

3. ***输入以下命令：***

```
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}'
```

请参见以下示例输出：

```
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
```

4. ***输入 extract 命令获取新托管集群的 kubeconfig ：***

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig -
-to=- > kubeconfig-<hosted-cluster-name>
```

5. ***在代理到达 add-to-existing-cluster 状态后，输入以下命令验证您可以在托管集群中看到 OpenShift Container Platform 节点：***

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get nodes
```

请参见以下示例输出：

```
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
```

集群 Operator 首先通过将工作负载添加到节点来协调。

6. 输入以下命令验证在扩展 NodePool 对象时是否创建了两台机器：

```
oc -n <hosted-control-plane-namespace> get machines
```

请参见以下示例输出：

```
NAME                          CLUSTER               NODENAME      PROVIDERID
PHASE     AGE   VERSION
hypercluster1-c96b6f675-m5vch   hypercluster1-b2qhl   ocp-worker-1   agent://da503cf1-
a347-44f2-875c-4960ddb04091   Running   15m   4.13z
hypercluster1-c96b6f675-tl42p   hypercluster1-b2qhl   ocp-worker-2   agent://4dac1ab2-
7dd5-4894-a220-6a3473b67ee6   Running   15m   4.13z
```

clusterversion 协调过程最终到达缺少 Ingress 和 Console 集群 Operator 的点。

7. 输入以下命令：

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get clusterversion,co
```

请参见以下示例输出：

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version             False       True          40m     Unable to apply
4.13z: 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
```

1.7.7.7.1. 添加节点池

您可以通过指定名称、副本数和任何其他信息（如代理标签选择器），为托管集群创建节点池。

1. 要创建节点池，请输入以下信息：

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"

hcp create nodepool agent \
  --cluster-name $CLUSTER_NAME \
  --name $NODEPOOL_NAME \
  --node-count $WORKER_COUNT \
  --agentLabelSelector '{"matchLabels": {"size": "medium"}}' 1
```

1 --agentLabelSelector 是可选的。节点池使用带有 "size" 标签的代理。

2. 通过列出 cluster 命名空间中的 nodepool 资源来检查节点池的状态：

```
oc get nodepools --namespace clusters
```

3. 输入以下命令提取 admin-kubeconfig secret：

```
oc extract -n <hosted-control-plane-namespace> secret/admin-kubeconfig --
to=./hostedcluster-secrets --confirm
```

请参见以下示例输出：

```
hostedcluster-secrets/kubeconfig
```

4. 一段时间后，您可以输入以下命令检查节点池的状态：

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

5. 输入以下命令验证可用节点池的数量是否与预期的节点池数量匹配：

```
oc get nodepools --namespace clusters
```

1.7.7.7.2. 其他资源

- 要将数据平面缩减为零，请参阅将数据平面缩减为零。

1.7.7.8. 处理裸机上托管的集群中的入口

每个 OpenShift Container Platform 集群都有一个默认的应用程序 Ingress Controller，它通常关联有外部 DNS 记录。例如，如果您创建一个名为 example 的托管集群，其基域为 krnl.es，您可以预期通配符 domain. apps.example.krnl.es 可以被路由。

要为 If apps 域设置负载均衡器和通配符 DNS 记录，请在客户机集群中执行以下操作：

1. 通过创建包含 MetalLB Operator 配置的 YAML 文件来部署 MetalLB：

```
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.

2. 将文件保存为 metallb-operator-config.yaml。

3. 输入以下命令应用配置：

```
oc apply -f metallb-operator-config.yaml
```

4. Operator 运行后，创建 MetalLB 实例：

   a. 创建包含 MetalLB 实例配置的 YAML 文件：

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb
```

   b. 将文件保存为 metallb-instance-config.yaml。

   c. 输入以下命令来创建 MetalLB 实例：

```
oc apply -f metallb-instance-config.yaml
```

5. 通过创建两个资源来配置 MetalLB Operator：

   - 具有单个 IP 地址的 IPAddressPool 资源。此 IP 地址必须与集群节点使用的网络位于同一个子网中。

   - 一个 BGPAdvertisement 资源，用于公告 IPAddressPool 资源通过 BGP 协议提供的负载均衡器 IP 地址。

     a. 创建 YAML 文件使其包含配置：

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: <ip_address_pool_name> 1
```

```
  namespace: metallb
spec:
  protocol: layer2
  autoAssign: false
  addresses:
    - <ingress_ip>-<ingress_ip> 2
---
apiVersion: metallb.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: <bgp_advertisement_name> 3
  namespace: metallb
spec:
  ipAddressPools:
    - <ip_address_pool_name> 4
```

1 4

指定 IPAddressPool 资源名称。

2

指定环境的 IP 地址，例如 192.168.122.23。

3

指定 BGPAdvertisement 资源名称。

a. 将文件保存为 ipaddresspool-bgpadvertisement-config.yaml。

b. 运行以下命令来创建资源：

```
oc apply -f ipaddresspool-bgpadvertisement-config.yaml
```

1. 创建 LoadBalancer 类型的服务后，MetalLB 为该服务添加一个外部 IP 地址。

c. 通过创建名为 metallb-loadbalancer-service.yaml 的 YAML 文件，配置将入口流量路由到 ingress 部署的新负载均衡器服务：

```
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
```

d. *保存 metallb-loadbalancer-service.yaml 文件。*

e. *输入以下命令应用 YAML 配置：*

```
oc apply -f metallb-loadbalancer-service.yaml
```

f. *输入以下命令访问 OpenShift Container Platform 控制台：*

```
curl -kI https://console-openshift-console.apps.example.krnl.es

HTTP/1.1 200 OK
```

g. *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
```

请参见以下示例输出：

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
clusterversion.config.openshift.io/version   4.x.y     True        False         3m32s   Cluster version is 4.x.y

NAME                                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console          4.x.y     True        False         False      3m50s
clusteroperator.config.openshift.io/ingress          4.x.y     True        False         False      53m
```

- + 将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。

1.7.7.8.1. 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。

1.7.7.9. 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的 worker 节点。

1. 要启用自动扩展，请输入以下命令：

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p
'[{"op": "remove", "path": "/spec/replicas"},{"op":"add", "path": "/spec/autoScaling", "value": {
"max": 5, "min": 2 }}]'
```

注意：在示例中，最少的节点数量为 2，最大为 5。您可以添加的最大节点数可能会受您的平台绑定。例如，如果您使用 Agent 平台，则最大节点数由可用代理数量绑定。

1. 创建需要新节点的工作负载。

   a. 使用以下示例创建一个包含工作负载配置的 YAML 文件：

```
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: {}
```

b. 将文件保存为 **workload-config.yaml**。

c. 输入以下命令应用 **YAML**：

```
oc apply -f workload-config.yaml
```

2. 输入以下命令提取 **admin-kubeconfig secret**：

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=./hostedcluster-secrets --confirm
```

请参见以下示例输出：

```
hostedcluster-secrets/kubeconfig
```

3. 您可以输入以下命令来检查新节点是否处于 **Ready** 状态：

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

4. 要删除节点，请输入以下命令删除工作负载：

```
oc --kubeconfig ./hostedcluster-secrets -n default delete deployment reversewords
```

5. 等待几分钟通过，无需额外容量。在 **Agent** 平台上，代理将停用，并可重复使用。您可以输入以下命令确认节点已被删除：

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

-

1.7.7.9.1. 为托管集群禁用节点自动扩展

要禁用节点自动扩展，请输入以下命令：

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p '[\
{"op":"remove", "path": "/spec/autoScaling"}, \{"op": "add", "path": "/spec/replicas", "value": <specify-
value-to-scale-replicas>]'
```

命令从 YAML 文件中删除 "spec.autoScaling"，添加 "spec.replicas"，并将 "spec.replicas" 设置为您指定的整数值。

1.7.7.10. 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。

1.7.7.10.1. 使用命令行销毁裸机上的托管集群

要销毁托管集群，请完成以下步骤：

- 运行以下命令来删除托管集群及其后端资源：

  ```
  hcp destroy cluster agent --name <hosted_cluster_name>
  ```

  将 <hosted_cluster_name > 替换为托管集群的名称。

1.7.8. 使用非裸机代理机器配置托管的 control plane 集群（技术预览）

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。托管集群也称为 管理集群。

注： 管理集群 与受管集群 不同。受管集群是一个 hub 集群管理的集群。

托管的 control plane 功能默认启用。

multicluster engine operator 2.5 仅支持默认的 local-cluster， 它是一个托管的 hub 集群，以及 hub 集群作为托管集群。在 Red Hat Advanced Cluster Management 2.10 中，您可以使用受管 hub 集群（也称为 local-cluster ）， 作为托管集群。

托管的集群 是一个 OpenShift Container Platform 集群， 其 API 端点和托管在托管集群中的 control plane。托管的集群包括控制平面和它的对应的数据平面。您可以使用 multicluster engine operator 控制台或托管的 control plane 命令行界面 hcp 创建托管集群。托管的集群自动导入为受管集群。如果要禁用此自动导入功能， 请参阅禁用将托管集群的自动导入到 multicluster engine operator。

重要：

- *每个托管集群都必须具有集群范围的唯一名称。托管的集群名称不能与任何现有受管集群相同，以便 multicluster engine operator 管理它。*
- *不要使用 集群 作为托管的集群名称。*
- *在托管 control plane 的同一平台上运行 hub 集群和 worker。*
- *无法在 multicluster engine operator 受管集群的命名空间中创建托管集群。*
- *您可以使用 Agent 平台将代理机器作为 worker 节点添加到托管集群。代理机器代表使用 Discovery Image 引导的主机，并准备好作为 OpenShift Container Platform 节点置备。Agent 平台是中央基础架构管理服务的一部分。如需更多信息， 请参阅启用中央基础架构管理服务。*
- *不是裸机的所有主机都需要使用中央基础架构管理提供的发现镜像 ISO 手动引导。*
-

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，会为每个副本创建一个机器。对于每台机器，Cluster API 供应商找到并安装一个经过批准的代理（通过验证）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。在重复使用代理前，您必须使用 Discovery 镜像重启它们。

- 当您为托管 control plane 配置存储时，请考虑推荐的 etcd 实践。要确保您满足延迟要求，请将快速存储设备专用于每个 control-plane 节点上运行的所有托管的 control plane etcd 实例。您可以使用 LVM 存储为托管 etcd pod 配置本地存储类。如需更多信息，请参阅 OpenShift Container Platform 文档中的使用逻辑卷管理器存储 的建议 etcd 实践 和持久性存储。

1.7.8.1. 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 2.5 及更新的版本，并在 OpenShift Container Platform 集群上安装。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。

- multicluster engine operator 必须至少有一个受管 OpenShift Container Platform 集群。local-cluster 会自动导入。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
```

- 您需要启用中央基础架构管理。如需更多信息，请参阅启用中央基础架构管理服务。

- 您需要安装托管的 control plane 命令行界面。

1.7.8.2. 非裸机代理机器的防火墙和端口要求

确保满足防火墙和端口要求，以便端口可以在管理集群、control plane 和托管的集群间进行通信：

- kube-apiserver 服务默认在端口 6443 上运行，需要 ingress 访问 control plane 组件之间的通信。
  - 如果使用 NodePort 发布策略，请确保公开给 kube-apiserver 服务的节点端口。
  - 如果使用 MetalLB 负载均衡，允许对用于负载均衡器 IP 地址的 IP 范围进行入口访问。
- 如果使用 NodePort 发布策略，请为 ignition-server 和 Oauth-server 设置使用防火墙规则。
- konnectivity 代理建立反向隧道，以允许托管集群上的双向通信，需要在端口 6443 上出口访问集群 API 服务器地址。通过该出口访问，代理可以访问 kube-apiserver 服务。
  - 如果集群 API 服务器地址是一个内部 IP 地址，允许从工作负载子网访问端口 6443 上的 IP 地址。
  - 如果地址是一个外部 IP 地址，允许将端口 6443 上的出口从节点传输到该外部 IP 地址。
- 如果您更改了 6443 的默认端口，请调整规则以反映该更改。
- 确保打开在集群中运行的工作负载所需的任何端口。
- 使用防火墙规则、安全组或其他访问控制来限制对所需源的访问。除非必要，否则请避免公开端口。
- 对于生产环境部署，请使用负载均衡器通过单个 IP 地址简化访问。

1.7.8.3. 非裸机代理机器的基础架构要求

Agent 平台不创建任何基础架构，但它对基础架构有以下要求：

- Agent ：代理代表使用发现镜像引导的主机，并准备好作为 OpenShift Container Platform 节点置备。
- DNS ：API 和入口端点必须可以被路由。

1.7.8.4. 在非裸机代理机器上配置 DNS

托管集群的 API 服务器作为 NodePort 服务公开。api.<hosted-cluster-name>.<basedomain> 必须存在 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入口 pod。

- 请参见以下 DNS 配置示例：

```
api-int.example.krnl.es.  IN A 192.168.122.22
`*`.apps.example.krnl.es. IN A 192.168.122.23
```

- 如果您要为 IPv6 网络上的断开连接的环境配置 DNS，请查看以下 DNS 配置示例：

```
api-int.example.krnl.es.  IN A 2620:52:0:1306::7
`*`.apps.example.krnl.es. IN A 2620:52:0:1306::10
```

- 如果您要为双栈网络上的断开连接的环境配置 DNS，请确保同时包含 IPv4 和 IPv6 的 DNS 条目。请参见以下 DNS 配置示例：

```
host-record=api-int.hub-dual.dns.base.domain.name,2620:52:0:1306::2
address=/apps.hub-dual.dns.base.domain.name/2620:52:0:1306::3
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,[2620:52:0:1306::5]
```

1.7.8.5. 在非裸机代理机器上创建托管集群

您可以创建托管集群或导入一个。有关导入托管集群的步骤，请参阅 导入托管集群。

1. 输入以下命令来创建托管的 control plane 命名空间：

```
oc create ns <hosted-cluster-namespace>-<hosted-cluster-name>
```

将 <hosted-cluster-namespace > 替换为托管的集群命名空间名称，例如 集群。将 <hosted-cluster-name > 替换为您的托管的集群名称。

2. 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。输入以下命令，将任何示例变量替换为您的信息：

```
hcp create cluster agent \
  --name=<hosted-cluster-name> \ 1
  --pull-secret=<path-to-pull-secret> \ 2
  --agent-namespace=<hosted-control-plane-namespace> \ 3
  --base-domain=<basedomain> \ 4
  --api-server-address=api.<hosted-cluster-name>.<basedomain> \
  --etcd-storage-class=<etcd-storage-class> \ 5
  --ssh-key  <path-to-ssh-key> \ 6
  --namespace <hosted-cluster-namespace> \ 7
  --control-plane-availability-policy SingleReplica \
  --release-image=quay.io/openshift-release-dev/ocp-release:<ocp-release> 8
```

1 指定托管集群的名称，例如。

2 指定 pull secret 的路径，例如 /user/name/pullsecret。

3 指定托管的 control plane 命名空间，如 cluster -example。使用 oc get agent -n <hosted-control-plane-namespace> 命令来确保代理在此命名空间中 可用。

4 指定您的基域，如 krnl.es。

5 指定 etcd 存储类名称，例如 lvm-storageclass。

6

指定 SSH 公钥的路径。默认文件路径为 ~/.ssh/id_rsa.pub。

7 指定托管集群的命名空间。

8 指定您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。

3. 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
```

请参见以下示例输出：

```
NAME                                     READY   STATUS    RESTARTS   AGE
catalog-operator-6cd867cc7-phb2q         2/2     Running   0          2m50s
control-plane-operator-f6b4c8465-4k5dh   1/1     Running   0          4m32s
```

1.7.8.5.1. 使用控制台在非裸机代理机器上创建托管集群

1. 打开 OpenShift Container Platform Web 控制台，并通过输入管理员凭证登录。有关打开控制台的说明，请参阅 OpenShift Container Platform 文档中的 访问 Web 控制台。

2. 在控制台标头中，确保选择了 All Clusters。

3. 点 Infrastructure > Clusters。

4. 点 Create cluster Host inventory > Hosted control plane。

   此时会显示 Create cluster 页面。

5. 在 Create cluster 页面中，按照提示输入集群、节点池、网络和自动化的详情。

注：当您输入集群详情时，您可能会发现以下有用提示：

- 如果要使用预定义的值来自动填充控制台中的字段，您可以创建主机清单凭证。如需更多信息，请参阅 为内部环境创建凭证。
- 在 Cluster details 页面中，pull secret 是用于访问 OpenShift Container Platform 资源的 OpenShift Container Platform pull secret。如果选择了主机清单凭证，则会自动填充 pull secret。
- 在 Node pool 页面中，命名空间包含节点池的主机。如果使用控制台创建主机清单，控制台会创建一个专用命名空间。
- 在 Networking 页面中，您可以选择 API 服务器发布策略。托管集群的 API 服务器可以通过使用现有的负载均衡器或 NodePort 类型的服务公开。必须存在 api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN} 设置的 DNS 条目，该设置指向可访问 API 服务器的目的地。此条目可以是指向管理集群中的一个节点的记录，或指向将传入的流量重定向到 Ingress pod 的负载均衡器的记录。

6. 检查您的条目并点 Create。

   此时会显示 Hosted 集群 视图。

7. 在 Hosted 集群 视图中监控托管集群的部署。如果没有看到有关托管集群的信息，请确保选择了 All Clusters，然后点集群名称。等待 control plane 组件就绪。这个过程可能需要几分钟时间。

8. 要查看节点池状态，可滚动到 NodePool 部分。安装节点的过程大约需要 10 分钟。您还可以单击 Nodes 来确认节点是否加入托管集群。

1.7.8.5.2. 其他资源

- 要创建在使用控制台创建托管集群时可以重复使用 的凭证，请参阅为内部环境创建凭证。
- 要导入托管集群，请参阅 手动导入托管的 control plane 集群。

- 要访问托管集群，请参阅 访问托管集群。

- 要使用 Discovery Image 将主机添加到主机清单中，请参阅使用 Discovery Image 将主机添加到主机清单中。

1.7.8.6. 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 <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=- > kubeconfig-<hosted-cluster-name>
```

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-<hosted_cluster_name>
```

请参见以下示例输出：

```
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
```

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-<hosted-cluster-name>
```

请参见以下示例输出：

```
NAMESPACE                                          NAME                                                  READY   STATUS    RESTARTS   AGE
kube-system                                        konnectivity-agent-khlqv                              0/1     Running   0          3m52s
openshift-cluster-samples-operator                 cluster-samples-operator-6b5bcb9dff-kpnbc             2/2     Running   0          20m
```

```
openshift-monitoring                    alertmanager-main-0                          6/6
Running     0          100s
openshift-monitoring                    openshift-state-metrics-677b9fb74f-qqp6g
3/3     Running     0          104s
```

1.7.8.7. 为托管集群扩展 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 <hosted-cluster-namespace> scale nodepool <nodepool-name> --replicas 2
```

Cluster API 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。代理按以下顺序传递状态：

- binding
- 发现
- insufficient
- 安装
- installing-in-progress
- added-to-existing-cluster

2. 输入以下命令：

```
oc -n <hosted-control-plane-namespace> get agent
```

请参见以下示例输出：

```
-
```

```
NAME                                   CLUSTER        APPROVED   ROLE         STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6   hypercluster1  true       auto-assign
```

3. 输入以下命令：

```
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}'
```

请参见以下示例输出：

```
BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: binding
BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: insufficient
```

4. 输入 **extract** 命令获取新托管集群的 **kubeconfig** ：

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig -
-to=- > kubeconfig-<hosted-cluster-name>
```

5. 在代理到达 **add-to-existing-cluster** 状态后，输入以下命令验证您可以在托管集群中看到 **OpenShift Container Platform** 节点：

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get nodes
```

请参见以下示例输出：

```
NAME          STATUS   ROLES    AGE     VERSION
ocp-worker-1  Ready    worker   5m41s   v1.24.0+3882f8f
```

集群 **Operator** 首先通过将工作负载添加到节点来协调。

6. 输入以下命令验证在扩展 **NodePool** 对象时是否创建了两台机器：

```
oc -n <hosted-control-plane-namespace> get machines
```

请参见以下示例输出：

```
NAME                    CLUSTER            NODENAME      PROVIDERID
PHASE     AGE   VERSION
hypercluster1-c96b6f675-m5vch   hypercluster1-b2qhl   ocp-worker-1   agent://da503cf1-
a347-44f2-875c-4960ddb04091   Running   15m   4.13z
```

clusterversion 协调过程最终到达缺少 Ingress 和 Console 集群 Operator 的点。

7. **输入以下命令：**

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get clusterversion,co
```

请参见以下示例输出：

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version             False       True          40m     Unable to apply
4.13z: the cluster operator console has not yet successfully rolled out

NAME                                                 VERSION   AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                  4.13z     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          4.13z     True        False
False      10m
clusteroperator.config.openshift.io/dns                      4.13z     True        False
False      9m16s
```

1.7.8.7.1. 添加节点池

您可以通过指定名称、副本数和任何其他信息（如代理标签选择器），为托管集群创建节点池。

1. **要创建节点池，请输入以下信息：**

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"

hcp create nodepool agent \
  --cluster-name $CLUSTER_NAME \
  --name $NODEPOOL_NAME \
  --node-count $WORKER_COUNT \
  --agentLabelSelector '{"matchLabels": {"size": "medium"}}' 1
```

1

--agentLabelSelector 是可选的。节点池使用带有 "size" 标签的代理。

2. 通过列出 cluster 命名空间中的 nodepool 资源来检查节点池的状态：

```
oc get nodepools --namespace clusters
```

3. 输入以下命令提取 admin-kubeconfig secret：

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=./hostedcluster-secrets --confirm
```

请参见以下示例输出：

```
hostedcluster-secrets/kubeconfig
```

4. 一段时间后，您可以输入以下命令检查节点池的状态：

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

5. 输入以下命令验证可用节点池的数量是否与预期的节点池数量匹配：

```
oc get nodepools --namespace clusters
```

1.7.8.7.2. 其他资源

- 要将数据平面缩减为零，请参阅将数据平面缩减为零。

1.7.8.8. 在非裸机代理机器上处理托管的集群中的入口

每个 OpenShift Container Platform 集群都有一个默认的应用程序 Ingress Controller，它通常关联有外部 DNS 记录。例如，如果您创建一个名为 example 的托管集群，其基域为 krnl.es，您可以预期通配符 domain. apps.example.krnl.es 可以被路由。

要为 If apps 域设置负载均衡器和通配符 DNS 记录，请在客户机集群中执行以下操作：

1. *通过创建包含 MetalLB Operator 配置的 YAML 文件来部署 MetalLB：*

```
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. *将文件保存为 metallb-operator-config.yaml。*

3. *输入以下命令应用配置：*

```
oc apply -f metallb-operator-config.yaml
```

4. *Operator 运行后，创建 MetalLB 实例：*

   a. *创建包含 MetalLB 实例配置的 YAML 文件：*

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb
```

   b.

b. 将文件保存为 metallb-instance-config.yaml。

c. 输入以下命令来创建 MetalLB 实例：

```
oc apply -f metallb-instance-config.yaml
```

5. 通过创建两个资源来配置 MetalLB Operator：

- 具有单个 IP 地址的 IPAddressPool 资源。此 IP 地址必须与集群节点使用的网络位于同一个子网中。
- 一个 BGPAdvertisement 资源，用于公告 IPAddressPool 资源通过 BGP 协议提供的负载均衡器 IP 地址。

a. 创建 YAML 文件使其包含配置：

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: <ip_address_pool_name> 1
  namespace: metallb
spec:
  protocol: layer2
  autoAssign: false
  addresses:
    - <ingress_ip>-<ingress_ip> 2
---
apiVersion: metallb.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: <bgp_advertisement_name> 3
  namespace: metallb
spec:
  ipAddressPools:
    - <ip_address_pool_name> 4
```

1 4 指定 IPAddressPool 资源名称。

2

指定环境的 IP 地址，例如 192.168.122.23。

3

指定 BGPAdvertisement 资源名称。

a. *将文件保存为 ipaddresspool-bgpadvertisement-config.yaml。*

b. *运行以下命令来创建资源：*

```
oc apply -f ipaddresspool-bgpadvertisement-config.yaml
```

1. *创建 LoadBalancer 类型的服务后，MetalLB 为该服务添加一个外部 IP 地址。*

c. *通过创建名为 metallb-loadbalancer-service.yaml 的 YAML 文件，配置将入口流量路由到 ingress 部署的新负载均衡器服务：*

```
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
```

d. *将文件保存为 metallb-loadbalancer-service.yaml。*

e. *输入以下命令应用 YAML 配置：*

```
oc apply -f metallb-loadbalancer-service.yaml
```

f. 输入以下命令访问 OpenShift Container Platform 控制台：

```
curl -kI https://console-openshift-console.apps.example.krnl.es

HTTP/1.1 200 OK
```

g. 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
```

请参见以下示例输出：

```
NAME                                         VERSION         AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.x.y           True        False         3m32s   Cluster
version is 4.x.y

NAME                                                 VERSION         AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                  4.x.y           True        False
False      3m50s
clusteroperator.config.openshift.io/ingress                  4.x.y           True        False
False      53m
```

+ 将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本，如 4.14.0-x86_64。

1.7.8.8.1. 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。

1.7.8.9. 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的 worker 节点。

1. 要启用自动扩展，请输入以下命令。在本例中，最少的节点数量为 2，最大为 5。您可以添加的最大节点数可能会受您的平台绑定。例如，如果您使用 Agent 平台，则最大节点数由可用代

理数量绑定：

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p
'[{"op": "remove", "path": "/spec/replicas"},{"op":"add", "path": "/spec/autoScaling", "value": {
"max": 5, "min": 2 }}]'
```

2. 创建需要新节点的工作负载。

   a. 使用以下示例创建一个包含工作负载配置的 YAML 文件：

```
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: {}
```

   b. 将文件保存为 workload-config.yaml。

   c. 输入以下命令应用 YAML：

```
oc apply -f workload-config.yaml
```

3. 输入以下命令提取 admin-kubeconfig secret：

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>admin-kubeconfig --to=./hostedcluster-secrets --confirm
```

请参见以下示例输出：

```
hostedcluster-secrets/kubeconfig
```

4. 您可以输入以下命令来检查新节点是否处于 Ready 状态：

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
```

5. 要删除节点，请输入以下命令删除工作负载：

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig -n default delete deployment reversewords
```

6. 等待几分钟通过，无需额外容量。在 Agent 平台上，代理将停用，并可重复使用。您可以输入以下命令确认节点已被删除：

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
```

1.7.8.9.1. 为托管集群禁用节点自动扩展

要禁用节点自动扩展，请输入以下命令：

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p '[\{"op":"remove", "path": "/spec/autoScaling"}, \{"op": "add", "path": "/spec/replicas", "value": <specify-value-to-scale-replicas>]'
```

命令从 YAML 文件中删除 "spec.autoScaling"，添加 "spec.replicas"，并将 "spec.replicas" 设置为您指定的整数值。

1.7.8.10. 在非裸机代理机器上销毁托管集群

您可以使用控制台销毁非裸机托管集群。完成以下步骤，在非裸机代理机器上销毁托管集群：

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。

1.7.8.10.1. 使用命令行销毁非裸机代理机器上的托管集群

要销毁托管集群，请完成以下步骤：

- 运行以下命令来删除托管集群及其后端资源：

```
hcp destroy cluster agent --name <hosted_cluster_name>
```

将 <hosted_cluster_name > 替换为托管集群的名称。

1.7.9. 在 64 位 x86 OpenShift Container Platform 集群上配置托管集群，为 IBM Power 计算节点创建托管的 control plane （技术预览）

技术预览： 在 64 位 x86 裸机中为 IBM Power (ppc64le)计算节点配置托管集群有有限的支持。

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。托管集群也称为 管理集群。

注： 管理集群 不是 受管集群。受管集群是一个 hub 集群管理的集群。

multicluster engine operator 2.5 仅支持默认的 local-cluster，它是一个托管的 hub 集群，以及 hub 集群作为托管集群。

重要：

- 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。有关中央基础架构管理服务简介，请参阅创建主机清单。

- 每个 IBM Power 系统主机都必须使用中央基础架构管理提供的发现镜像启动。每个主机启动后，它运行一个 Agent 进程来发现主机的详细信息并完成安装。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。

1.7.9.1. 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 2.5 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。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。

- multicluster engine operator 必须至少有一个受管 OpenShift Container Platform 集群。local-cluster 在 multicluster engine operator 2.5 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。

- 您需要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。

-

您需要安装托管的 control plane 命令行界面。请参阅 安装托管的 control plane 命令行界面。

1.7.9.2. IBM Power 基础架构要求

Agent 平台不创建任何基础架构，但对于基础架构需要以下内容：

- Agent ：代理代表使用发现镜像引导的主机，并准备好作为 OpenShift Container Platform 节点置备。
- DNS ：API 和入口端点必须可以被路由。

1.7.9.3. IBM Power 配置文档

满足先决条件后，请查看以下主题来在裸机上配置托管的 control plane：

1. 在 InfraEnv 资源中添加代理
2. 为 IBM Power 上托管的 control plane 配置 DNS
3. 在裸机上创建托管集群
4. 为 IBM Power 计算节点在 64 位 x86 裸机上托管 control plane 创建 InfraEnv 资源
5. 为 IBM Power 上托管集群扩展 NodePool 对象。

1.7.9.4. 在 InfraEnv 资源中添加代理

您可以通过手动将机器配置为使用 live ISO 来添加代理。

1. 下载 live ISO，并使用它来启动主机（裸机或虚拟机）。live ISO 的 URL 可以在 InfraEnv 资源（在 status.isoDownloadURL 字段中）中找到。在启动时，主机与 Assisted Service 通信，并注册为与 InfraEnv 资源相同的命名空间中的代理。

2. 要列出代理及其某些属性，请输入以下命令：

```
oc -n <hosted-control-plane-namespace> get agents
```

请参见以下示例输出：

```
NAME                                   CLUSTER   APPROVED   ROLE         STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218                 auto-assign
e57a637f-745b-496e-971d-1abbf03341ba                 auto-assign
```

3. 创建每个代理后，您可以选择在规格中设置 **installation_disk_id** 和 **hostname**，并输入以下命令批准代理：

```
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
```

4. 要验证代理是否已批准使用，请输入以下命令并检查输出：

```
oc -n <hosted-control-plane-namespace> get agents
```

请参见以下示例输出：

```
NAME                                   CLUSTER   APPROVED   ROLE         STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218             true       auto-assign
e57a637f-745b-496e-971d-1abbf03341ba             true       auto-assign
```

1.7.9.5. 为 IBM Power 上托管的 control plane 配置 DNS

托管集群的 API 服务器公开。**api.<hosted-cluster-name>.<base-domain>** 条目必须存在 DNS 条目，该条目指向可访问 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入口 pod。

请参阅以下 DNS 配置示例：

```
$ cat /var/named/<example.krnl.es.zone>
```

请参见以下示例输出：

```
$ TTL 900
@ IN  SOA bastion.example.krnl.es.com. hostmaster.example.krnl.es.com. (
     2019062002
     1D 1H 1W 3H )
  IN NS bastion.example.krnl.es.com.
;
;
api                IN A 1xx.2x.2xx.1xx (1)
api-int             IN A 1xx.2x.2xx.1xx
;
;
*.apps.<hosted-cluster-name>.<basedomain>          IN A 1xx.2x.2xx.1xx
;
;EOF
```

(1) *记录指的是 API 负载均衡器的 IP 地址，它处理托管的 control plane 的入口和出口流量。*

对于 IBM Power，添加与代理 IP 地址对应的 IP 地址。

```
compute-0          IN A 1xx.2x.2xx.1yy
compute-1          IN A 1xx.2x.2xx.1yy
```

1.7.9.6. 为 IBM Power 计算节点在 64 位 x86 裸机上托管 control plane 创建 InfraEnv 资源

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。

要创建 InfraEnv 资源，请完成以下步骤：

1. 创建包含配置的 YAML 文件。请参见以下示例：

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: <hosted-cluster-name>
  namespace: <hosted-control-plane-namespace>
spec:
  cpuArchitecture: ppc64le
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <ssh-public-key>
```

2. 将文件保存为 infraenv-config.yaml。

3. 输入以下命令应用配置：

```
oc apply -f infraenv-config.yaml
```

4. 要获取用于下载 live ISO 的 URL，它允许 IBM Power 机器加入代理，请输入以下命令：

```
oc -n <hosted-control-plane-namespace> get InfraEnv <hosted-cluster-name> -o json
```

1.7.9.7. 为 IBM Power 上托管集群扩展 NodePool 对象

在创建托管集群时，会创建 NodePool 对象。通过扩展 NodePool 对象，您可以将更多计算节点添加到托管的 control plane 中。

1. 运行以下命令来将 NodePool 对象扩展到两个节点：

```
oc -n <clusters_namespace> scale nodepool <nodepool_name> --replicas 2
```

Cluster API 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。代理按以下顺序传递转换阶段：

- binding

- *发现*
- ***insufficient***
- ***安装***
- ***installing-in-progress***
- ***added-to-existing-cluster***

2. *运行以下命令查看特定扩展代理的状态：*

```
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}'
```

请参见以下输出：

```
BMH: Agent: 50c23cda-cedc-9bbd-bcf1-9b3a5c75804d State: known-unbound
BMH: Agent: 5e498cd3-542c-e54f-0c58-ed43e28b568a State: insufficient
```

3. *运行以下命令查看转换阶段：*

```
oc -n <hosted_control_plane_namespace> get agent
```

请参见以下输出：

```
NAME                                   CLUSTER          APPROVED   ROLE          STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   hosted-forwarder   true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a                      true       auto-assign
da503cf1-a347-44f2-875c-4960ddb04091   hosted-forwarder   true       auto-assign
```

4. *运行以下命令以生成 kubeconfig 文件来访问托管集群：*

```
hcp create kubeconfig --namespace <clusters_namespace> --name
<hosted_cluster_namespace> > <hosted_cluster_name>.kubeconfig
```

5. 在代理到达 add-to-existing-cluster 状态后，输入以下命令验证您可以看到 OpenShift Container Platform 节点：

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get nodes
```

请参见以下输出：

```
NAME                          STATUS   ROLES    AGE      VERSION
worker-zvm-0.hostedn.example.com Ready    worker   5m41s    v1.24.0+3882f8f
worker-zvm-1.hostedn.example.com Ready    worker   6m3s     v1.24.0+3882f8f
```

6. 输入以下命令验证在扩展 NodePool 对象时是否创建了两台机器：

```
oc -n <hosted_control_plane_namespace> get machine.cluster.x-k8s.io
```

请参见以下输出：

```
NAME                           CLUSTER  NODENAME PROVIDERID     PHASE     AGE
VERSION
hosted-forwarder-79558597ff-5tbqp   hosted-forwarder-crqq5   worker-zvm-
0.hostedn.example.com   agent://50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   Running
41h   4.15.0
hosted-forwarder-79558597ff-lfjfk   hosted-forwarder-crqq5   worker-zvm-
1.hostedn.example.com   agent://5e498cd3-542c-e54f-0c58-ed43e28b568a   Running
41h   4.15.0
```

7. 运行以下命令检查集群版本和集群 Operator 状态：

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion
```

请参见以下输出：

```
NAME                                 VERSION      AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.15.0   True        False         40h     Cluster
version is 4.15.0
```

8. 运行以下命令来检查集群 Operator 状态：

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusteroperators
```

对于集群中的每个组件，输出显示以下集群 operator 状态： NAME,VERSION, AVAILABLE ,AVAILABLE,PROGRESSING,DEGRADED,SINCE, 和 MESSAGE。

如需输出示例，请参阅 OpenShift Container Platform 文档中的 Initial Operator 配置部分。

1.7.9.7.1. 其他资源

- 要将数据平面缩减为零，请参阅将数据平面缩减为零。

1.7.10. 为 IBM Z 计算节点在 x86 裸机上配置托管集群（技术预览）

技术预览： 在 IBM Z (s390x)计算节点的 x86 裸机中配置托管集群处于技术预览状态，且支持有限的。

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。托管集群也称为 管理集群。

注： 管理集群 不是 受管集群。受管集群是一个 hub 集群管理的集群。

您可以使用 hypershift 附加组件将受管集群转换为托管集群，以在该集群中部署 HyperShift Operator。然后，您可以开始创建托管集群。

multicluster engine operator 2.5 仅支持默认的 local-cluster，它是一个托管的 hub 集群，以及 hub 集群作为托管集群。

重要：

- 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。有关中央基础架构管理服务简介，请参阅 Kube API - 入门指南。

- 每个 IBM Z 系统主机都必须使用中央基础架构管理提供的 PXE 镜像启动。每个主机启动后，它运行一个 Agent 进程来发现主机的详细信息并完成安装。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，Hy HyperShift Operator 会在托管的 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 PXE 镜像引导集群，以更新节点的数量。

1.7.10.1. 先决条件

- Kubernetes operator 版本 2.5 或更高版本的多集群引擎必须安装在 OpenShift Container Platform 集群中。在安装 Red Hat Advanced Cluster Management 时，多集群引擎 operator 会被自动安装。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。

- multicluster engine operator 必须至少有一个受管 OpenShift Container Platform 集群。local-cluster 在 multicluster engine operator 2.5 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
```

- 您需要至少具有三个 worker 节点的托管集群来运行 HyperShift Operator。

- 您需要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。

- 您需要安装托管的 control plane 命令行界面。请参阅 安装托管的 control plane 命令行界面。

1.7.10.2. IBM Z 基础架构要求

Agent 平台不创建任何基础架构，但对于基础架构需要以下内容：

- *Agent ：代理代表使用发现镜像或 PXE 镜像引导的主机，并可置备为 OpenShift Container Platform 节点。*
- *DNS ：API 和 Ingress 端点必须可以被路由。*

托管的 control plane 功能默认启用。如果您禁用了这个功能并希望手动启用它，或者需要禁用这个功能，请参阅启用或禁用 托管的 control plane 功能。

1.7.10.3. IBM Z 配置文档

满足先决条件后，请查看以下主题来在裸机上配置托管的 control plane：

1. *使用 IBM Z 为托管的 control plane 配置 DNS*
2. *在裸机上创建托管集群*
3. *为 IBM Z 计算节点在 64 位 x86 裸机上托管 control plane 创建 InfraEnv 资源*
4. *在 InfraEnv 资源中添加 IBM Z 代理（技术预览）*
5. *为 IBM Z 上托管集群扩展 NodePool 对象*

1.7.10.4. 在 InfraEnv 资源中添加 IBM Z 代理（技术预览）

在 IBM Z 环境中添加代理需要额外的步骤，本节中详细介绍。

注： 除非另有说明，这些步骤适用于 IBM Z 和 IBM LinuxONE 上的 z/VM 和 RHEL KVM 安装。

1.7.10.4.1. 使用 KVM 为 IBM Z 添加代理

对于带有 KVM 的 IBM Z，运行以下命令，使用从 InfraEnv 资源下载的 PXE 镜像启动 IBM Z 环境。创建代理后，主机与 Assisted Service 通信，并注册到与管理集群上的 InfraEnv 资源相同的命名空间中。

```
virt-install \
  --name "<vm_name>" \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=4 \
  --location "<path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img" \
  --disk <qcow_image_path> \
  --network network:macvtap-net,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --wait=-1 \
  --extra-args "rd.neednet=1 nameserver=<nameserver>
coreos.live.rootfs_url=http://<http_server>/rootfs.img random.trust_cpu=on
rd.luks.options=discard ignition.firstboot ignition.platform.id=metal console=tty1
console=ttyS1,115200n8 coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8"
```

对于 ISO 引导，从 InfraEnv 资源下载 ISO，并运行以下命令来引导节点：

```
virt-install \
  --name "<vm_name>" \
  --autostart \
  --memory=16384 \
  --cpu host \
  --vcpus=4 \
  --network network:macvtap-net,mac=<mac_address> \
  --cdrom "<path_to_image.iso>" \
  --disk <qcow_image_path> \
  --graphics none \
  --noautoconsole \
  --os-variant <os_version> \
  --wait=-1 \
```

1.7.10.4.2. 使用 z/VM 为 IBM 添加代理

注：如果要为 z/VM 客户机使用静态 IP，您必须为 z/VM 代理配置 NMStateConfig 属性，以便 IP 参数在第二次引导中保留。

完成以下步骤，使用从 InfraEnv 资源下载的 PXE 镜像启动 IBM Z 环境。创建代理后，主机与 Assisted Service 通信，并注册到与管理集群上的 InfraEnv 资源相同的命名空间中。

1. 更新参数文件，以添加 rootfs_url、network_adaptor 和 disk_type 值。

   请参见以下示例参数文件：

```
rd.neednet=1 \
console=ttysclp0 \
coreos.live.rootfs_url=<rootfs_url> \
ip=<IP_guest_vm>::<nameserver>:255.255.255.0::<network_adaptor>:none \
nameserver=<nameserver> \
zfcp.allow_lun_scan=0 \ 1
rd.znet=qeth,<network_adaptor_range>,layer2=1 \
rd.<disk_type>=<storage> random.trust_cpu=on \ 2
rd.luks.options=discard \
ignition.firstboot ignition.platform.id=metal \
console=tty1 console=ttyS1,115200n8 \
coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8
```

   1. 对于使用 VSwitch 的安装，请添加 zfcp.allow_lun_scan=0。使用 OSA、Hipersockets 和 RoCE 安装省略此条目。

   2. 对于在 DASD 类型磁盘中安装，请使用 rd.dasd= 指定安装磁盘。对于在 FCP 类型磁盘中安装，请使用 rd.zfcp=。

2. 运行以下命令，将 initrd、内核镜像和参数文件移到客户机虚拟机中：

```
vmur pun -r -u -N kernel.img $INSTALLERKERNELLOCATION/<image name>
```

```
vmur pun -r -u -N generic.parm $PARMFILELOCATION/paramfilename
```

```
vmur pun -r -u -N initrd.img $INSTALLERINITRAMFSLOCATION/<image name>
```

3. 在客户机虚拟机控制台中运行以下命令：

```
cp ipl c
```

4. 要列出代理及其属性，请输入以下命令：

```
oc -n <hosted_control_plane_namespace> get agents
```

请参见以下示例输出：

```
NAME    CLUSTER APPROVED    ROLE    STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d    auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a    auto-assign
```

5. 运行以下命令来批准代理。可选： 您可以在规格中设置代理 ID < installation_disk_id> 和 <hostname > ：

```
oc -n <hosted_control_plane_namespace> patch agent 50c23cda-cedc-9bbd-bcf1-
9b3a5c75804d -p '{"spec":
{"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-zvm-
0.hostedn.example.com"}}' --type merge
```

6. 运行以下命令验证代理是否已批准：

```
oc -n <hosted_control_plane_namespace> get agents
```

请参见以下示例输出：

```
NAME                                    CLUSTER     APPROVED   ROLE         STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d              true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a              true       auto-assign
```

1.7.10.5. 为使用 IBM Z 托管 control plane 配置 DNS

托管集群的 API 服务器作为 'NodePort' 服务公开。api.<hosted-cluster-name>.<base-domain> 必须存在 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。

请参阅以下 DNS 配置示例：

```
$ cat /var/named/<example.krnl.es.zone>
```

请参见以下示例输出：

```
$ TTL 900
@ IN  SOA bastion.example.krnl.es.com. hostmaster.example.krnl.es.com. (
      2019062002
      1D 1H 1W 3H )
  IN NS bastion.example.krnl.es.com.
;
;
api                IN A 1xx.2x.2xx.1xx 1
api-int             IN A 1xx.2x.2xx.1xx
;
;
*.apps        IN A 1xx.2x.2xx.1xx
;
;EOF
```

1

记录指的是 API 负载均衡器的 IP 地址，它处理托管的 control plane 的入口和出口流量。

对于 IBM z/VM，添加与代理 IP 地址对应的 IP 地址。

```
compute-0            IN A 1xx.2x.2xx.1yy
compute-1            IN A 1xx.2x.2xx.1yy
```

1.7.10.6. 为 IBM Z 计算节点的 x86 裸机上托管 control plane 创建 InfraEnv 资源

InfraEnv 是一个环境，使用 PXE 镜像引导的主机可以作为代理加入。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。

请参阅以下流程来创建 InfraEnv 资源：

1. 创建包含配置的 YAML 文件。请参见以下示例：

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: <hosted-cluster-name>
  namespace: <hosted-control-plane-namespace>
spec:
  cpuArchitecture: s390x
```

```
pullSecretRef:
  name: pull-secret
sshAuthorizedKey: <ssh-public-key>
```

2. 将文件保存为 **infraenv-config.yaml**。

3. 输入以下命令应用配置：

```
oc apply -f infraenv-config.yaml
```

4. 要获取用于下载 PXE 镜像的 URL，如 initrd.img、kernel.img 或 rootfs.img，它允许 IBM Z 机器作为代理加入，请输入以下命令：

```
oc -n <hosted-control-plane-namespace> get InfraEnv <hosted-cluster-name> -o json
```

1.7.10.7. 为 IBM Z 上托管集群扩展 NodePool 对象

在创建托管集群时，会创建 NodePool 对象。通过扩展 NodePool 对象，您可以将更多计算节点添加到托管的 control plane 中。

1. 运行以下命令来将 NodePool 对象扩展到两个节点：

```
oc -n <clusters_namespace> scale nodepool <nodepool_name> --replicas 2
```

Cluster API 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。代理按以下顺序传递转换阶段：

- binding
- 发现
- insufficient
- 安装

- *installing-in-progress*
- *added-to-existing-cluster*

2. *运行以下命令查看特定扩展代理的状态：*

```
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}'
```

请参见以下输出：

```
BMH: Agent: 50c23cda-cedc-9bbd-bcf1-9b3a5c75804d State: known-unbound
BMH: Agent: 5e498cd3-542c-e54f-0c58-ed43e28b568a State: insufficient
```

3. *运行以下命令查看转换阶段：*

```
oc -n <hosted_control_plane_namespace> get agent
```

请参见以下输出：

```
NAME                                   CLUSTER          APPROVED   ROLE          STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   hosted-forwarder   true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a                      true       auto-assign
da503cf1-a347-44f2-875c-4960ddb04091   hosted-forwarder   true       auto-assign
```

4. *运行以下命令以生成 kubeconfig 文件来访问托管集群：*

```
hcp create kubeconfig --namespace <clusters_namespace> --name
<hosted_cluster_namespace> > <hosted_cluster_name>.kubeconfig
```

5. *在代理到达 add-to-existing-cluster 状态后，输入以下命令验证您可以看到 OpenShift Container Platform 节点：*

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get nodes
```

请参见以下输出：

```
NAME                          STATUS   ROLES    AGE     VERSION
worker-zvm-0.hostedn.example.com Ready    worker   5m41s   v1.24.0+3882f8f
worker-zvm-1.hostedn.example.com Ready    worker   6m3s    v1.24.0+3882f8f
```

集群 Operator 首先通过将工作负载添加到节点来协调。

6. 输入以下命令验证在扩展 NodePool 对象时是否创建了两台机器：

```
oc -n <hosted_control_plane_namespace> get machine.cluster.x-k8s.io
```

请参见以下输出：

```
NAME                          CLUSTER  NODENAME PROVIDERID     PHASE     AGE
VERSION
hosted-forwarder-79558597ff-5tbqp   hosted-forwarder-crqq5   worker-zvm-
0.hostedn.example.com   agent://50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   Running
41h   4.15.0
hosted-forwarder-79558597ff-lfjfk   hosted-forwarder-crqq5   worker-zvm-
1.hostedn.example.com   agent://5e498cd3-542c-e54f-0c58-ed43e28b568a   Running
41h   4.15.0
```

7. 运行以下命令来检查集群版本：

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion,co
```

请参见以下输出：

```
NAME                                         VERSION      AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.15.0-ec.2   True        False         40h
Cluster version is 4.15.0-ec.2
```

8. 运行以下命令来检查集群 Operator 状态：

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusteroperators
```

对于集群中的每个组件，输出显示以下集群 operator 状态： NAME,VERSION, AVAILABLE

,AVAILABLE,PROGRESSING,DEGRADED,SINCE, 和 MESSAGE。

如需输出示例，请参阅 OpenShift Container Platform 文档中的 Initial Operator 配置部分。

1.7.10.7.1. 其他资源

- *要将数据平面缩减为零，请参阅将数据平面缩减为零。*

1.7.11. 在 OpenShift Virtualization 中管理托管的 control plane 集群

使用托管的 control plane 和 Red Hat OpenShift Virtualization，您可以创建带有由 KubeVirt 虚拟机托管的 worker 节点的 OpenShift Container Platform 集群。在 OpenShift Virtualization 上托管的 control plane 提供了几个优点：

- *通过在相同的底层裸机基础架构中打包托管的 control plane 和托管集群来提高资源使用量*
- *分隔托管的 control plane 和托管集群，以提供强大的隔离*
- *通过消除裸机节点 bootstrap 过程来减少集群置备时间*
- *管理同一基本 OpenShift Container Platform 集群下的多个发行版本*

托管的 control plane 功能默认启用。

您可以使用托管的 control plane 命令行界面 hcp 创建 OpenShift Container Platform 托管的集群。托管的集群自动导入为受管集群。如果要禁用此自动导入功能，请参阅禁用将托管集群的自动导入到 multicluster engine operator。

重要：

- *在托管 control plane 的同一平台上运行 hub 集群和 worker。*
-

每个托管集群都必须具有集群范围的唯一名称。托管的集群名称不能与任何现有受管集群相同，以便 multicluster engine operator 管理它。

- 不要使用 集群 作为托管的集群名称。
- 无法在 multicluster engine operator 受管集群的命名空间中创建托管集群。
- 当您为托管 control plane 配置存储时，请考虑推荐的 etcd 实践。要确保您满足延迟要求，请将快速存储设备专用于每个 control-plane 节点上运行的所有托管的 control plane etcd 实例。您可以使用 LVM 存储为托管 etcd pod 配置本地存储类。如需更多信息，请参阅 OpenShift Container Platform 文档中的使用逻辑卷管理器存储 的建议 etcd 实践 和持久性存储。

1.7.11.1. 先决条件

您必须满足以下先决条件才能在 OpenShift Virtualization 上创建 OpenShift Container Platform 集群：

- 您需要管理员访问由 KUBECONFIG 环境变量指定的 OpenShift Container Platform 集群，版本 4.14 或更高版本。
- OpenShift Container Platform 托管集群必须启用通配符 DNS 路由，如下所示：

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

- OpenShift Container Platform 托管集群必须安装有 OpenShift Virtualization 版本 4.14 或更高版本。如需更多信息，请参阅使用 Web 控制台安装 OpenShift Virtualization。
- OpenShift Container Platform 托管集群必须使用 OVNKubernetes 配置为默认 pod 网络 CNI。
- OpenShift Container Platform 托管集群必须具有默认存储类。如需更多信息，请参阅 安装后存储配置。以下示例演示了如何设置默认存储类：

```
oc patch storageclass ocs-storagecluster-ceph-rbd -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- 您需要 quay.io/openshift-release-dev 存储库的有效的 pull secret 文件。如需更多信息，请参阅使用用户置备的基础架构在任何 x86_64 平台上安装 OpenShift。

- 您需要安装托管的 control plane 命令行界面。

- 在置备集群前，您需要配置负载均衡器。如需更多信息，请参阅 可选：配置 MetalLB。

- 为获得最佳网络性能，请在托管 KubeVirt 虚拟机的 OpenShift Container Platform 集群上使用网络最大传输单元(MTU) 9000 或更高。如果您使用较低的 MTU 设置，则影响托管 pod 的网络延迟和吞吐量。仅在 MTU 为 9000 或更高版本时，在节点池中启用多队列。

- multicluster engine operator 必须至少有一个受管 OpenShift Container Platform 集群。local-cluster 会自动导入。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
```

1.7.11.2. 防火墙和端口要求

确保满足防火墙和端口要求，以便端口可以在管理集群、control plane 和托管的集群间进行通信：

- kube-apiserver 服务默认在端口 6443 上运行，需要 ingress 访问 control plane 组件之间的通信。
  - 如果使用 NodePort 发布策略，请确保公开给 kube-apiserver 服务的节点端口。
  - 如果使用 MetalLB 负载均衡，允许对用于负载均衡器 IP 地址的 IP 范围进行入口访问。

- 如果使用 NodePort 发布策略，请为 ignition-server 和 Oauth-server 设置使用防火墙规则。

- konnectivity 代理建立反向隧道，以允许托管集群上的双向通信，需要在端口 6443 上出口访问集群 API 服务器地址。通过该出口访问，代理可以访问 kube-apiserver 服务。

- 如果集群 API 服务器地址是一个内部 IP 地址，允许从工作负载子网访问端口 6443 上的 IP 地址。
- 如果地址是一个外部 IP 地址，允许将端口 6443 上的出口从节点传输到该外部 IP 地址。

- 如果您更改了 6443 的默认端口，请调整规则以反映该更改。
- 确保打开在集群中运行的工作负载所需的任何端口。
- 使用防火墙规则、安全组或其他访问控制来限制对所需源的访问。除非必要，否则请避免公开端口。
- 对于生产环境部署，请使用负载均衡器通过单个 IP 地址简化访问。

有关 Red Hat OpenShift Virtualization 上托管 control plane 的其他资源，请参阅以下文档：

- 要了解 etcd 和 LVM 存储建议，请参阅使用逻辑卷管理器存储 的建议 etcd 实践 和持久性存储。
- 要在断开连接的环境中在 Red Hat OpenShift Virtualization 上配置托管的 control plane，请参阅 在断开连接的环境中配置托管的 control plane。
- 要禁用托管的 control plane 功能，或者已经禁用了它并希望手动启用它，请参阅 启用或禁用托管的 control plane 功能。
- 要通过运行 Red Hat Ansible Automation Platform 作业来管理托管集群，请参阅配置 Ansible Automation Platform 作业以便在托管集群中运行。

1.7.11.3. 使用 KubeVirt 平台创建托管集群

在 OpenShift Container Platform 4.14 及更新的版本中，您可以创建一个具有 KubeVirt 的集群，使其包含使用外部基础架构创建。了解更多有关使用 KubeVirt 创建的进程的信息：

- *创建托管集群*
- *使用外部基础架构创建托管集群*

1.7.11.3.1. 创建托管集群

1. *要创建托管集群，请使用托管的 control plane 命令行界面 hcp ：*

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--etcd-storage-class=<etcd-storage-class> 6
```

1. *指定托管集群的名称，例如。*
2. *指定 worker 数量，如 2。*
3. *指定 pull secret 的路径，例如 /user/name/pullsecret。*
4. *为 memory 指定一个值，例如 6Gi。*
5. *为 CPU 指定一个值，例如 2。*
6. *指定 etcd 存储类名称，例如 lvm-storageclass。*

注：您可以使用 --release-image 标志使用特定的 OpenShift Container Platform 发行版本设置托管集群。

根据 --node-pool-replicas 标志，为集群创建一个默认节点池，它有两个虚拟机 worker 副本。

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 clusters-<hosted-cluster-name> get pods
```

请参见以下示例输出：

```
NAME                                   READY   STATUS    RESTARTS   AGE
capi-provider-5cc7b74f47-n5gkr         1/1     Running   0          3m
catalog-operator-5f799567b7-fd6jw      2/2     Running   0          69s
certified-operators-catalog-784b9899f9-mrp6p   1/1     Running   0          66s
cluster-api-6bbc867966-l4dwl           1/1     Running   0          66s
.
.
.
redhat-operators-catalog-9d5fd4d44-z8qqk       1/1     Running   0          66s
```

具有 KubeVirt 虚拟机支持的 worker 节点的托管集群通常需要 10-15 分钟才能被完全置备。

3. *要检查托管集群的状态，请输入以下命令来查看对应的 HostedCluster 资源：*

```
oc get --namespace clusters hostedclusters
```

请参阅以下示例输出，它演示了一个完全置备的 HostedCluster 对象：

```
NAMESPACE   NAME      VERSION   KUBECONFIG                PROGRESS    AVAILABLE   PROGRESSING   MESSAGE
clusters    example   4.x.0     example-admin-kubeconfig   Completed   True        False         The hosted control plane is available
```

将 4.x.0 替换为您要使用的 OpenShift Container Platform 版本。

4. 按照访问托管集群中的说明 访问托管集群。

1.7.11.3.2. 使用外部基础架构创建托管集群

默认情况下，HyperShift Operator 同时托管托管集群的 control plane pod 和同一集群中的 KubeVirt worker 虚拟机。使用外部基础架构功能，您可以将 worker 节点虚拟机放在与 control plane pod 独立的集群中。

- 管理集群 是运行 HyperShift Operator 的 OpenShift Container Platform 集群，并为托管集群托管 control plane pod。
- 基础架构集群是 为托管集群运行 KubeVirt worker 虚拟机的 OpenShift Container Platform 集群。
- 默认情况下，管理集群也充当托管虚拟机的基础架构集群。但是，对于外部基础架构，管理和基础架构集群会有所不同。

1.7.11.3.2.1. 外部基础架构的先决条件

- 您必须在外部基础架构集群中有一个命名空间，才能托管 KubeVirt 节点。
- 您必须具有外部基础架构集群的 kubeconfig 文件。

1.7.11.3.2.2. 使用 hcp 命令行界面创建托管集群

您可以使用 hcp 命令行界面创建托管集群。

1. 要将 KubeVirt worker 虚拟机放在基础架构集群中，请使用 --infra-kubeconfig-file 和 --infra-namespace 参数，如下例所示：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
```

```
--cores <value-for-cpu> \ 5
--infra-namespace=<hosted-cluster-namespace>-<hosted-cluster-name> \ 6
--infra-kubeconfig-file=<path-to-external-infra-kubeconfig> 7
```

1 指定托管集群的名称，例如。

2 指定 worker 数量，如 2。

3 指定 pull secret 的路径，例如 /user/name/pullsecret。

4 为 memory 指定一个值，例如 6Gi。

5 为 CPU 指定一个值，例如 2。

6 指定基础架构命名空间，如 cluster -example。

7 为基础架构集群指定 kubeconfig 文件的路径，如 /user/name/external-infra-kubeconfig。

输入该命令后，control plane pod 托管在 HyperShift Operator 上运行的管理集群上，并且 KubeVirt 虚拟机托管在一个单独的基础架构集群中。

2. 按照访问托管集群中的说明 访问托管集群。

1.7.11.3.3. 使用控制台创建托管集群

要使用控制台创建带有 KubeVirt 平台的托管集群，请完成以下步骤：

1. 打开 OpenShift Container Platform Web 控制台，并通过输入管理员凭证登录。有关打开控制台的说明，请参阅 OpenShift Container Platform 文档中的 访问 Web 控制台。

2. 在控制台标头中，确保选择了 All Clusters。

3. 点 Infrastructure > Clusters。

4. 点 Create cluster > Red Hat OpenShift Virtualization > Hosted。

5. 在 Create cluster 页面中，按照提示输入集群和节点池的详情。

   备注：

   - 如果要使用预定义的值来自动填充控制台中的字段，您可以创建 Red Hat OpenShift Virtualization 凭证。如需更多信息，请参阅 为内部环境创建凭证。

   - 在 Cluster details 页面中，pull secret 是用于访问 OpenShift Container Platform 资源的 OpenShift Container Platform pull secret。如果您选择了 Red Hat OpenShift Virtualization 凭证，pull secret 会自动填充。

6. 检查您的条目并点 Create。

   此时会显示 Hosted 集群 视图。

7. 在 Hosted 集群 视图中监控托管集群的部署。如果没有看到有关托管集群的信息，请确保选择了 All Clusters，然后点集群名称。

8. 等待 control plane 组件就绪。这个过程可能需要几分钟时间。

9. 要查看节点池状态，可滚动到 NodePool 部分。安装节点的过程大约需要 10 分钟。您还可以单击 Nodes 来确认节点是否加入托管集群。

1.7.11.3.4. 其他资源

- 要创建在使用控制台创建托管集群时可以重复使用的凭证，请参阅为内部环境创建凭证。
- 要访问托管集群，请参阅访问托管集群。

1.7.11.4. 默认入口和 DNS 行为

每个 OpenShift Container Platform 集群都包括一个默认的应用程序 Ingress Controller，它必须关联有通配符 DNS 记录。默认情况下，使用 HyperShift KubeVirt 供应商创建的托管集群会自动成为 KubeVirt 虚拟机上运行的 OpenShift Container Platform 集群的子域。

例如，OpenShift Container Platform 集群可能有以下默认入口 DNS 条目：

```
*.apps.mgmt-cluster.example.com
```

因此，名为 guest 的 KubeVirt 托管集群，并在该底层 OpenShift Container Platform 集群上运行有以下默认入口：

```
*.apps.guest.apps.mgmt-cluster.example.com
```

要使默认入口 DNS 正常工作，托管 KubeVirt 虚拟机的集群必须允许通配符 DNS 路由。您可以输入以下命令来配置此行为：

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

注：当您使用默认托管集群入口时，连接会被限制为通过端口 443 进行 HTTPS 流量。通过端口 80 的普通 HTTP 流量将被拒绝。这个限制只适用于默认的入口行为。

1.7.11.4.1. 自定义入口和 DNS 行为

如果您不想使用默认入口和 DNS 行为，您可以在创建时配置带有唯一基域的 KubeVirt 托管集群。这个选项需要创建过程中手动配置步骤，并涉及三个主要步骤：集群创建、负载均衡器创建和通配符 DNS 配置。

1.7.11.4.1.1. 部署指定基域的托管集群

1. 要创建指定基域的托管集群，请输入以下命令：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--base-domain <basedomain> 6
```

1 指定托管集群的名称，例如。

2 指定 worker 数量，如 2。

3 指定 pull secret 的路径，例如 /user/name/pullsecret。

4 为 memory 指定一个值，例如 6Gi。

5 为 CPU 指定一个值，例如 2。

6 指定基域，如 hypershift.lab。

因此，托管集群有一个入口通配符，它为集群名称和基域配置，如 .apps.example.hypershift.lab。托管的集群处于 Partial 状态。因为在使用唯一基域创建托管集群后，您必须配置所需的 DNS 记录和负载均衡器。

1. 输入以下命令来查看托管集群的状态：

```
oc get --namespace clusters hostedclusters
```

请参见以下示例输出：

```
NAME          VERSION   KUBECONFIG                  PROGRESS   AVAILABLE
PROGRESSING   MESSAGE
example       example-admin-kubeconfig    Partial    True       False        The
hosted control plane is available
```

2. 输入以下命令访问集群：

```
hcp create kubeconfig --name <hosted-cluster-name> > <hosted-cluster-name>-kubeconfig
```

```
oc --kubeconfig <hosted-cluster-name>-kubeconfig get co
```

请参见以下示例输出：

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
console                              4.x.0     False       False         False      30m
RouteHealthAvailable: failed to GET route (https://console-openshift-
console.apps.example.hypershift.lab): Get "https://console-openshift-
console.apps.example.hypershift.lab": dial tcp: lookup console-openshift-
console.apps.example.hypershift.lab on 172.31.0.10:53: no such host
ingress                              4.x.0     True        False         True       28m     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)
```

将 4.x.0 替换为您要使用的 OpenShift Container Platform 版本。

后续步骤修复了输出中的错误。

注：如果托管集群位于裸机上，您可能需要 MetalLB 来设置负载均衡器服务。如需更多信息，请参阅 可选：配置 MetalLB。

1.7.11.4.1.2. 设置负载均衡器

设置将入口流量路由到 KubeVirt 虚拟机的负载均衡器服务，并为负载均衡器 IP 地址分配通配符 DNS 条目。

1. 公开托管集群入口的 NodePort 服务已存在。您可以导出节点端口并创建以这些端口为目标的负载均衡器服务。

   a. 输入以下命令来获取 HTTP 节点端口：

   ```
   oc --kubeconfig <hosted-cluster-name>-kubeconfig get services -n openshift-ingress router-nodeport-default -o jsonpath='{.spec.ports[?(@.name=="http")].nodePort}'
   ```

   注意下一步中使用的 HTTP 节点端口值。

   b. 输入以下命令来获取 HTTPS 节点端口：

   ```
   oc --kubeconfig <hosted-cluster-name>-kubeconfig get services -n openshift-ingress router-nodeport-default -o jsonpath='{.spec.ports[?(@.name=="https")].nodePort}'
   ```

   请注意下一步中使用的 HTTPS 节点端口值。

2. 运行以下命令来创建负载均衡器服务：

```
oc apply -f -
apiVersion: v1
kind: Service
metadata:
  labels:
    app: <hosted-cluster-name>
  name: <hosted-cluster-name>-apps
  namespace: clusters-<hosted-cluster-name>
spec:
  ports:
  - name: https-443
    port: 443
    protocol: TCP
    targetPort: <https-node-port> 1
  - name: http-80
    port: 80
    protocol: TCP
    targetPort: <http-node-port> 2
  selector:
    kubevirt.io: virt-launcher
  type: LoadBalancer
```

1

指定您在上一步中记录的 HTTPS 节点端口值。

2

指定您在上一步中记录的 HTTP 节点端口值。

1.7.11.4.1.3. 设置通配符 DNS

设置引用负载均衡器服务外部 IP 的通配符 DNS 记录或 CNAME。

1. 输入以下命令来获取外部 IP 地址：

```
oc -n clusters-<hosted-cluster-name> get service <hosted-cluster-name>-apps -o
jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

请参见以下示例输出：

```
192.168.20.30
```

2. 配置引用外部 IP 地址的通配符 DNS 条目。查看以下 DNS 条目示例：

```
*.apps.<hosted-cluster-name\>.<base-domain\>.
```

DNS 条目必须能够在集群内部和外部路由。请参阅以下 DNS 解析示例：

```
dig +short test.apps.example.hypershift.lab

192.168.20.30
```

3. 输入以下命令检查托管的集群状态已从 Partial 移到 Completed ：

```
oc get --namespace clusters hostedclusters
```

请参见以下示例输出：

```
NAME          VERSION   KUBECONFIG                PROGRESS    AVAILABLE
```

```
PROGRESSING   MESSAGE
example       4.x.0    example-admin-kubeconfig       Completed   True        False       The hosted control plane is available
```

将 4.x.0 替换为您要使用的 OpenShift Container Platform 版本。

1.7.11.4.1.4. 其他资源

- 在 OpenShift Virtualization 中管理托管的 control plane 集群
- 可选：配置 MetalLB
- 返回到此主题的开头，默认入口和 DNS 行为。

1.7.11.5. 可选：配置 MetalLB

您必须先安装 MetalLB Operator，然后才能配置 MetalLB。如需更多信息，请参阅 OpenShift Container Platform 文档中的安装 MetalLB Operator。

执行以下步骤在客户机集群中配置 MetalLB：

1. 通过在 configure-metallb.yaml 文件中保存以下示例 YAML 内容来创建 MetalLB 资源：

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb-system
```

2. 输入以下命令应用 YAML 内容：

```
oc apply -f configure-metallb.yaml
```

请参见以下示例输出：

```
metallb.metallb.io/metallb created
```

3. 通过在 create-ip-address-pool.yaml 文件中保存以下示例 YAML 内容来创建 IPAddressPool 资源：

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: metallb
  namespace: metallb-system
spec:
  addresses:
  - 192.168.216.32-192.168.216.122 1
```

1 创建在节点网络中可用 IP 地址范围的地址池。将 IP 地址范围替换为网络中的可用 IP 地址池。

4. 输入以下命令应用 YAML 内容：

```
oc apply -f create-ip-address-pool.yaml
```

请参见以下示例输出：

```
ipaddresspool.metallb.io/metallb created
```

5. 通过在 l2advertisement.yaml 文件中保存以下示例 YAML 内容来创建 L2Advertisement 资源：

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
   - metallb
```

6. 输入以下命令应用 YAML 内容：

```
oc apply -f l2advertisement.yaml
```

请参见以下示例输出：

```
l2advertisement.metallb.io/metallb created
```

1.7.11.5.1. 其他资源

- 如需有关 MetalLB 的更多信息，请参阅安装 MetalLB Operator。

1.7.11.6. 为节点池配置额外网络、保证 CPU 和虚拟机调度

如果您需要为节点池配置额外网络，请对虚拟机(VM)请求保证 CPU 访问或管理 KubeVirt 虚拟机的调度，请参阅以下步骤。

1.7.11.6.1. 将多个网络添加到节点池

默认情况下，节点池生成的节点附加到 pod 网络。您可以使用 Multus 和 NetworkAttachmentDefinition 将额外网络附加到节点。

要将多个网络添加到节点，请运行以下命令来使用 --additional-network 参数：

```
hcp create cluster kubevirt \
--name <hosted_cluster_name> \ 1
--node-pool-replicas <worker_node_count> \ 2
--pull-secret <path_to_pull_secret> \ 3
--memory <memory> \ 4
--cores <cpu> \ 5
--additional-network name:<namespace/name> \ 6
—-additional-network name:<namespace/name>
```

1. 指定托管集群的名称，例如。
2. 指定 worker 节点数，例如 2。
3. 指定 pull secret 的路径，例如 /user/name/pullsecret。

4

指定内存值，例如 8Gi。

5

指定 CPU 值，例如 2。

6

将 -additional-network 参数的值设置为 name:<namespace/name>。将 <namespace/name > 替换为 NetworkAttachmentDefinition 的命名空间和名称。

1.7.11.6.2. 请求保证 CPU 资源

默认情况下，KubeVirt 虚拟机可能会与节点上的其他工作负载共享其 CPU。这会影响虚拟机的性能。为了避免性能影响，您可以请求虚拟机的 CPU 访问。

要请求保证的 CPU 资源，请运行以下命令将 --qos-class 参数设置为 Guaranteed ：

```
hcp create cluster kubevirt \
--name <hosted_cluster_name> \ 1
--node-pool-replicas <worker_node_count> \ 2
--pull-secret <path_to_pull_secret> \ 3
--memory <memory> \ 4
--cores <cpu> \ 5
--qos-class Guaranteed 6
```

1

指定托管集群的名称，例如。

2

指定 worker 节点数，例如 2。

3

指定 pull secret 的路径，例如 /user/name/pullsecret。

4

指定内存值，例如 8Gi。

5

指定 CPU 值，例如 2。

6

--qos-class Guaranteed 参数保证指定数量的 CPU 资源被分配给虚拟机。

1.7.11.6.3. 在一组节点上调度 KubeVirt VM

默认情况下，节点池创建的 KubeVirt 虚拟机被调度到任何可用的节点。您可以将 KubeVirt VM 调度到具有足够容量以运行虚拟机的特定节点。

要在特定节点池中调度 KubeVirt 虚拟机，请运行以下命令来使用 --vm-node-selector 参数：

```
hcp create cluster kubevirt \
--name <hosted_cluster_name> \ 1
--node-pool-replicas <worker_node_count> \ 2
--pull-secret <path_to_pull_secret> \ 3
--memory <memory> \ 4
--cores <cpu> \ 5
--vm-node-selector <label_key>=<label_value>,<label_key>=<label_value> 6
```

1

指定托管集群的名称，例如。

2

指定 worker 节点数，例如 2。

3

指定 pull secret 的路径，例如 /user/name/pullsecret。

4

指定内存值，例如 8Gi。

5

指定 CPU 值，例如 2。

6

1.7.11.7. 扩展节点池

1. 您可以使用 oc scale 命令手动扩展节点池：

```
NODEPOOL_NAME=${CLUSTER_NAME}-work
NODEPOOL_REPLICAS=5

oc scale nodepool/$NODEPOOL_NAME --namespace clusters --
replicas=$NODEPOOL_REPLICAS
```

2. 片刻后，输入以下命令查看节点池的状态：

```
oc --kubeconfig $CLUSTER_NAME-kubeconfig get nodes
```

请参见以下示例输出：

```
NAME              STATUS   ROLES    AGE     VERSION
example-9jvnf     Ready    worker   97s     v1.27.4+18eadca
example-n6prw     Ready    worker   116m    v1.27.4+18eadca
example-nc6g4     Ready    worker   117m    v1.27.4+18eadca
example-thp29     Ready    worker   4m17s   v1.27.4+18eadca
example-twxns     Ready    worker   88s     v1.27.4+18eadca
```

1.7.11.7.1. 添加节点池

您可以通过指定名称、副本数和任何其他信息（如内存和 CPU 要求）来为托管集群创建节点池。

1. 要创建节点池，请输入以下信息：在本例中，节点池分配有更多 CPU:

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"
export MEM="6Gi"
export CPU="4"
export DISK="16"

hcp create nodepool kubevirt \
  --cluster-name $CLUSTER_NAME \
  --name $NODEPOOL_NAME \
  --node-count $WORKER_COUNT \
```

```
  --memory $MEM \
  --cores $CPU
  --root-volume-size $DISK
```

2. 通过列出 cluster 命名空间中的 nodepool 资源来检查节点池的状态：

```
oc get nodepools --namespace clusters
```

请参见以下示例输出：

```
NAME                CLUSTER         DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION   UPDATINGVERSION   UPDATINGCONFIG
MESSAGE
example             example         5               5               False         False        4.x.0
example-extra-cpu   example         2                               False         False                True
True          Minimum availability requires 2 replicas, current 0 available
```

将 4.x.0 替换为您要使用的 OpenShift Container Platform 版本。

3. 一段时间后，您可以输入以下命令检查节点池的状态：

```
oc --kubeconfig $CLUSTER_NAME-kubeconfig get nodes
```

请参见以下示例输出：

```
NAME                  STATUS   ROLES    AGE     VERSION
example-9jvnf         Ready    worker   97s     v1.27.4+18eadca
example-n6prw         Ready    worker   116m    v1.27.4+18eadca
example-nc6g4         Ready    worker   117m    v1.27.4+18eadca
example-thp29         Ready    worker   4m17s   v1.27.4+18eadca
example-twxns         Ready    worker   88s     v1.27.4+18eadca
example-extra-cpu-zh9l5   Ready    worker   2m6s    v1.27.4+18eadca
example-extra-cpu-zr8mj   Ready    worker   102s    v1.27.4+18eadca
```

4. 输入以下命令验证节点池是否处于您所期望的状态：

```
oc get nodepools --namespace clusters
```

请参见以下示例输出：

```
NAME                CLUSTER        DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION   UPDATINGVERSION   UPDATINGCONFIG
MESSAGE
example             example        5               5               False         False        4.x.0
example-extra-cpu   example        2               2               False         False        4.x.0
```

将 4.x.0 替换为您要使用的 OpenShift Container Platform 版本。

1.7.11.7.1.1. 其他资源

- 在 OpenShift Virtualization 中管理托管的 control plane 集群
- 返回到此主题的开头，扩展节点池。
- 要将数据平面缩减为零，请参阅将数据平面缩减为零。

1.7.11.8. 在 OpenShift Virtualization 上验证托管集群创建

要验证托管集群是否已成功创建，请完成以下步骤。

1. 输入以下命令验证 HostedCluster 资源是否过渡到 completed 状态：

```
oc get --namespace clusters hostedclusters ${CLUSTER_NAME}
```

请参见以下示例输出：

```
NAMESPACE   NAME      VERSION   KUBECONFIG              PROGRESS    AVAILABLE
PROGRESSING   MESSAGE
clusters    example   4.12.2    example-admin-kubeconfig   Completed   True        False
The hosted control plane is available
```

2. 输入以下命令验证托管的集群中的所有集群 Operator 是否在线：

```
hcp create kubeconfig --name $CLUSTER_NAME > $CLUSTER_NAME-kubeconfig
```

```
oc get co --kubeconfig=$CLUSTER_NAME-kubeconfig
```

请参见以下示例输出：

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
console                                    4.12.2    True        False         False      2m38s
csi-snapshot-controller                    4.12.2    True        False         False      4m3s
dns                                        4.12.2    True        False         False      2m52s
image-registry                             4.12.2    True        False         False      2m8s
ingress                                    4.12.2    True        False         False      22m
kube-apiserver                             4.12.2    True        False         False      23m
kube-controller-manager                    4.12.2    True        False         False      23m
kube-scheduler                             4.12.2    True        False         False      23m
kube-storage-version-migrator              4.12.2    True        False         False      4m52s
monitoring                                 4.12.2    True        False         False      69s
network                                    4.12.2    True        False         False      4m3s
node-tuning                                4.12.2    True        False         False      2m22s
openshift-apiserver                        4.12.2    True        False         False      23m
openshift-controller-manager               4.12.2    True        False         False      23m
openshift-samples                          4.12.2    True        False         False      2m15s
operator-lifecycle-manager                 4.12.2    True        False         False      22m
operator-lifecycle-manager-catalog         4.12.2    True        False         False      23m
operator-lifecycle-manager-packageserver   4.12.2    True        False         False      23m
service-ca                                 4.12.2    True        False         False      4m41s
storage                                    4.12.2    True        False         False      4m43s
```

1.7.11.9. 为 OpenShift Virtualization 上托管的 control plane 配置存储

如果没有提供高级配置，则默认存储类用于 KubeVirt 虚拟机(VM)镜像、KubeVirt CSI 映射和 etcd 卷。

1.7.11.9.1. 映射 KubeVirt CSI 存储类

kubevirt CSI 允许任何具有 ReadWriteMany 访问模式的基础架构存储类公开给托管集群。您可以使用 --infra-storage-class-mapping 参数在集群创建过程中将基础架构集群存储类的这个映射配置为托管集群的存储类。

要将基础架构存储类映射到托管存储类，请参阅以下示例：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--infra-storage-class-mapping=<storage-class>/<hosted-storage-class> \ 6
```

1 指定托管集群的名称，例如。

2 指定 worker 数量，如 2。

3 指定 pull secret 的路径，例如 /user/name/pullsecret。

4 为 memory 指定一个值，例如 6Gi。

5 为 CPU 指定一个值，例如 2。

6 将 <storage-class > 替换为基础架构存储类名称，将 < hosted-storage-class > 替换为托管的集群存储类名称。您可以在 create 命令中多次使用 --infra-storage-class-mapping 参数。

创建托管集群后，基础架构存储类会在托管集群中可见。当您在使用其中一个存储类的托管集群中创建 PVC 时，KubeVirt CSI 会使用您在集群创建过程中配置的基础架构存储类映射来置备卷。

注： KubeVirt CSI 仅支持映射能够 ReadWriteMany (RWX)访问的基础架构存储类。

1.7.11.9.2. 配置 KubeVirt VM root 卷

在集群创建时，您可以使用 --root-volume-storage-class 参数配置用于托管 KubeVirt VM root 卷的存储类。

要为 KubeVirt 虚拟机设置自定义存储类和卷大小，请参阅以下示例：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
```

```
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--root-volume-storage-class <root-volume-storage-class> \ 6
--root-volume-size <volume-size> 7
```

1 指定托管集群的名称，例如。

2 指定 worker 数量，如 2。

3 指定 pull secret 的路径，例如 /user/name/pullsecret。

4 为 memory 指定一个值，例如 6Gi。

5 为 CPU 指定一个值，例如 2。

6 指定用于托管 KubeVirt VM root 卷的存储类的名称，如 ocs-storagecluster-ceph-rbd。

7 指定卷大小，例如 64。

结果是一个托管的、由 ocs-storagecluster-ceph-rdb 存储类托管的 PVC 上的虚拟机。

1.7.11.9.3. 启用 KubeVirt 虚拟机镜像缓存

kubevirt 镜像缓存是一个高级功能，可用于优化集群启动时间和存储利用率。此功能需要使用能够智能克隆和 ReadWriteMany 访问模式的存储类。有关智能克隆的更多信息，请参阅使用智能克隆克隆数据卷。

镜像缓存按如下方式工作：

1. 虚拟机镜像导入到与托管集群关联的 PVC 中。

2. 为添加到集群的每个 KubeVirt 虚拟机创建一个 PVC 的唯一克隆。

镜像缓存只需要单个镜像导入来减少虚拟机启动时间。当存储类支持写时复制克隆时，它可以进一步减少集群存储使用。

要启用镜像缓存，在集群创建过程中使用 **--root-volume-cache-strategy=PVC** 参数，如下例所示：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--root-volume-cache-strategy=PVC 6
```

1. 指定托管集群的名称，例如。

2. 指定 worker 数量，如 2。

3. 指定 pull secret 的路径，例如 /user/name/pullsecret。

4. 为 memory 指定一个值，例如 6Gi。

5. 为 CPU 指定一个值，例如 2。

6. 为镜像缓存指定一个策略，如 PVC。

1.7.11.9.4. 配置 etcd 存储

在集群创建时，您可以使用 --etcd-storage-class 参数配置用于托管 etcd 数据的存储类。

要为 etcd 配置存储类，请参阅以下示例：

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--etcd-storage-class=<etcd-storage-class-name> 6
```

1. 指定托管集群的名称，例如。
2. 指定 worker 数量，如 2。
3. 指定 pull secret 的路径，例如 /user/name/pullsecret。
4. 为 memory 指定一个值，例如 6Gi。
5. 为 CPU 指定一个值，例如 2。
6. 指定 etcd 存储类名称，例如 lvm-storageclass。如果没有提供 --etcd-storage-class 参数，则会使用默认存储类。

1.7.11.9.4.1. 其他资源

- 使用 smart-cloning（智能克隆） 克隆数据卷

1.7.11.10. 在 OpenShift Virtualization 上销毁托管集群

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，删除 multicluster engine operator 上的受管集群资源：

```
oc delete managedcluster <managed_cluster_name>
```

其中 <managed_cluster_name > 是受管集群的名称。

2. 运行以下命令来删除托管集群及其后端资源：

```
hcp destroy cluster kubevirt --name <hosted_cluster_name>
```

将 <hosted_cluster_name > 替换为您的托管的集群名称。

1.7.12. 在断开连接的环境中配置托管的 control plane

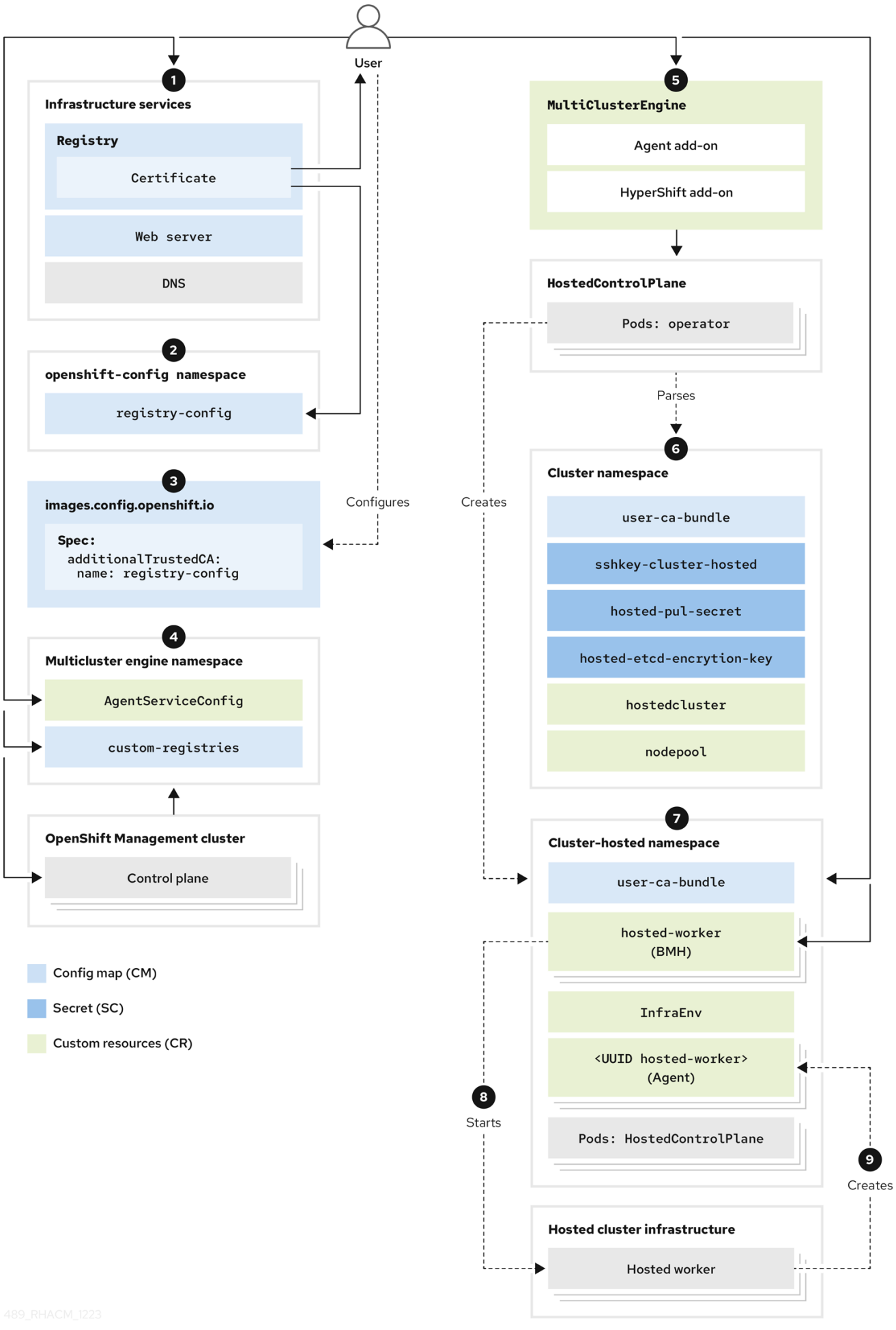
断开连接的环境是没有连接到互联网的 OpenShift Container Platform 集群，它使用托管的 control plane 作为基础。

技术预览：您可以使用 IPv4 或 IPv6 网络在裸机平台上在断开连接的环境中部署托管的 control plane。另外，在断开连接的环境中托管的 control plane 作为一个技术预览功能在双栈网络中提供。如果您使用 Red Hat OpenShift Virtualization 平台，则在断开连接的环境中托管的 control plane 仅作为技术预览功能提供。

1.7.12.1. 断开连接的环境架构

您可以使用 Agent 平台在裸机上置备托管的 control plane。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。有关中央基础架构管理服务简介，请参阅 启用中央基础架构管理服务。

请参阅以下断开连接的环境架构图：


User
1
Infrastructure services
Registry
Certificate
Web server
DNS
2
openshift-config namespace
registry-config
3
images.config.openshift.io
Spec:
additionalTrustedCA:
name: registry-config
4
Multicluster engine namespace
AgentServiceConfig
custom-registries
OpenShift Management cluster
Control plane
Configures
Creates
5
MultiClusterEngine
Agent add-on
HyperShift add-on
HostedControlPlane
Pods: operator
Parses
6
Cluster namespace
user-ca-bundle
sshkey-cluster-hosted
hosted-pul-secret
hosted-etcd-encrytion-key
hostedcluster
nodepool
7
Cluster-hosted namespace
user-ca-bundle
hosted-worker
(BMH)
InfraEnv
<UUID hosted-worker>
(Agent)
Pods: HostedControlPlane
8
Starts
9
Creates
Hosted cluster infrastructure
Hosted worker
Config map (CM)
Secret (SC)
Custom resources (CR)
489_RHACM_1223

1. 配置基础架构服务，包括带有 TLS 支持、Web 服务器和 DNS 的 registry 证书部署，以确保断开连接的部署正常工作。

2. 在 openshift-config 命名空间中创建配置映射。配置映射的内容是 Registry CA 证书。配置映射的 data 字段必须包含以下键和值：

   - Key: & lt;registry_dns_domain_name>..<port >，例如 registry.hypershiftdomain.lab..5000:。在指定端口时，请确保将 .. 放在 registry DNS 域名后。
   - Value ：证书内容

   有关创建配置映射的更多信息，请参阅为 IPv4 网络配置 TLS 证书。

3. 在 images.config.openshift.io 自定义资源(CR)中添加 additionalTrustedCA 字段，值设为 name: registry-config。

4. 在 multicluster-engine 命名空间中创建配置映射。请参见以下示例配置：

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: | 1
    -----BEGIN CERTIFICATE-----
    ...
    ...
    ...
    -----END CERTIFICATE-----
  registries.conf: | 2
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true
```

```
    [[registry.mirror]]
      location = "registry.ocp-edge-cluster-0.qe.lab.redhat.com:5000/openshift4"

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
...
...
```

1

包含 registry 的 CA 证书。

2

包含 RAW 格式的 registry.conf 文件内容。

1. 在 multicluster engine operator 命名空间中，您可以创建 multiclusterengine CR，它将同时启用 Agent 和 hypershift-addon 附加组件。multicluster engine operator 命名空间必须包含配置映射，才能修改断开连接的部署中的行为。命名空间还包含 multicluster-engine、assisted-service 和 hypershift-addon-manager pod。

2. 为以下组件创建对象来部署托管集群：

   - secret： Secret 包含 pull secret、SSH 密钥和 etcd 加密密钥。
   - 配置映射 ：配置映射包含私有 registry 的 CA 证书。
   - HostedCluster ： HostedCluster 资源定义托管集群的配置。
   - NodePool: NodePool 资源标识引用用于数据平面的机器的节点池。

3. 创建托管集群对象后，HyperShift Operator 在 HostedControlPlane 命名空间中创建 control plane pod。HostedControlPlane 命名空间还托管代理、裸机主机和 InfraEnv 资源等组件。

4. 创建 InfraEnv 资源。生成 ISO 镜像后，创建裸机主机及其包含基板管理控制器 (BMC)凭证的 secret。

5. *openshift-machine-api 命名空间中的 Metal3 Operator 会检查新的裸机主机。*

6. *Metal3 Operator 会尝试使用 LiveISO 和 RootFS 值连接并启动 BMC。您可以通过 multicluster engine operator 命名空间中的 AgentServiceConfig CR 指定 LiveISO 和 RootFS 值。*

7. *启动 HostedCluster 资源的 worker 节点后，会启动 Agent 容器。*

8. *将 NodePool 资源扩展到 HostedCluster 资源的 worker 节点数量。*

9. *等待部署过程完成。*

1.7.12.2. 先决条件

要在断开连接的环境中配置托管的 control plane，您必须满足以下条件：

- *CPU：提供的 CPU 数量决定了可以同时运行多少个托管集群。通常，每个节点为 3 个节点使用 16 个 CPU。对于最小开发，您可以对 3 个节点使用 12 个 CPU。*
- *memory： RAM 量会影响您可以托管的托管集群的数量。每个节点使用 48 GB RAM。对于最小开发，18 GB RAM 可能已经足够。*
- *Storage ：将 SSD 存储用于多集群引擎 operator。*
  - *管理集群： 250 GB.*
  - *registry：所需的 registry 存储取决于托管的发行版本、Operator 和镜像的数量。您可能需要 500 GB，最好与托管集群的磁盘分开。*
  - *Web 服务器：所需的 Web 服务器存储取决于托管的 ISO 和镜像的数量。您可能需要 500 GB。*

- *生产环境：对于生产环境，将管理集群、registry 和 Web 服务器分隔在不同的磁盘上。请参阅以下用于生产环境的示例配置：*
  - *Registry: 2 TB*
  - *管理集群：500 GB*
  - *Web 服务器：2 TB*

1.7.12.3. 提取 OpenShift Container Platform 发行镜像摘要

您可以使用标记的镜像提取 OpenShift Container Platform 发行镜像摘要。完成以下步骤：

1. *运行以下命令来获取镜像摘要：*

```
oc adm release info <tagged_openshift_release_image> | grep "Pull From"
```

将 <tagged_openshift_release_image > 替换为支持的 OpenShift Container Platform 版本标记的镜像，例如 quay.io/openshift-release-dev/ocp-release:4.14.0-x8_64。

请参见以下示例输出：

```
Pull From: quay.io/openshift-release-dev/ocp-release@sha256:69d1292f64a2b67227c5592c1a7d499c7d00376e498634ff8e1946bc9ccdddfe
```

要了解更多有关镜像标签和摘要的信息，请参阅 OpenShift Container Platform 文档中的镜像流中的 镜像。

1.7.12.3.1. 其他资源

- *为 IPv4 网络配置 TLS 证书*

- 引用镜像流中的镜像

1.7.12.4. 在断开连接的环境中监控用户工作负载

hypershift-addon 受管集群附加组件在 HyperShift Operator 中启用 --enable-uwm-telemetry-remote-write 选项。通过启用该选项，您可以确保启用了用户工作负载监控，并确保可以从 control plane 远程编写遥测指标。

如果您在没有连接到互联网的 OpenShift Container Platform 集群中安装 multicluster engine operator，当尝试通过输入以下命令来运行 HyperShift Operator 的用户工作负载监控功能时，该功能会失败并显示错误：

```
oc get events -n hypershift
```

这个错误可能类似以下示例：

```
LAST SEEN   TYPE      REASON          OBJECT               MESSAGE
4m46s       Warning   ReconcileError  deployment/operator  Failed to ensure UWM telemetry remote write: cannot get telemeter client secret: Secret "telemeter-client" not found
```

要避免这个错误，您必须在 local-cluster 命名空间中创建配置映射来禁用用户工作负载监控选项。您可以在启用附加组件之前或之后创建配置映射。附加组件代理重新配置 HyperShift Operator。

创建以下配置映射：

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: hypershift-operator-install-flags
  namespace: local-cluster
data:
  installFlagsToAdd: ""
  installFlagsToRemove: "--enable-uwm-telemetry-remote-write"
```

1.7.12.4.1. 验证托管 control plane 功能的状态

托管的 control plane 功能默认启用。

1. 如果禁用了这个功能，且您要启用它，请输入以下命令。将 multiclusterengine 替换为 multicluster engine operator 实例的名称：

```
oc patch mce <multiclusterengine> --type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift","enabled": true}]}}}'
```

当您启用这个功能时，**hypershift-addon 受管集群附加组件安装在 local-cluster 受管集群中，附加组件代理会在 multicluster engine operator hub 集群上安装 HyperShift Operator。**

2. 输入以下命令确认已安装 **hypershift-addon 受管集群附加组件：**

```
oc get managedclusteraddons -n local-cluster hypershift-addon
```

3. 查看生成的输出：

```
NAME              AVAILABLE   DEGRADED   PROGRESSING
hypershift-addon   True        False
```

4. 要避免这个过程出现超时，请输入以下命令：

```
oc wait --for=condition=Degraded=True managedclusteraddons/hypershift-addon -n local-
cluster --timeout=5m
```

```
oc wait --for=condition=Available=True managedclusteraddons/hypershift-addon -n local-
cluster --timeout=5m
```

过程完成后，**hypershift-addon 受管集群附加组件和 HyperShift Operator 会被安装，local-cluster 受管集群可用于托管和管理托管集群。**

1.7.12.4.2. 配置 hypershift-addon 受管集群附加组件以便在基础架构节点上运行

默认情况下，未为 **hypershift-addon 受管集群附加组件指定节点放置首选项。考虑在基础架构节点上运行附加组件，因为这样做可以防止因订阅计数和单独的维护和管理任务造成计费成本。**

1. 登录到 **hub 集群。**

2. 输入以下命令打开 **hypershift-addon-deploy-config 附加组件部署配置规格以进行编辑：**

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

3. 在规格中添加 nodePlacement 字段，如下例所示：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  nodePlacement:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      operator: Exists
```

4. 保存更改。hypershift-addon 受管集群附加组件部署在用于新和现有受管集群的基础架构节点上。

1.7.12.5. 使用 IPv4 在断开连接的环境中配置托管的 control plane

您可以使用 IPv4 网络在断开连接的环境中配置托管的 control plane。IPv4 范围需要比 IPv6 或双栈设置更少的外部组件。

查看以下步骤在 IPv4 网络上配置托管的 control plane：

1. 为 IPv4 网络配置虚拟机监控程序
2. 为 IPv4 网络配置 DNS
3. 为 IPv4 网络部署 registry
4. 为 IPv4 网络设置管理集群
5. 为 IPv4 网络配置 Web 服务器

6. 为 IPv4 网络配置镜像镜像

7. 为 IPv4 网络部署多集群引擎 operator

8. 为 IPv4 网络配置 TLS 证书

9. 为 IPv4 网络部署托管集群

10. 完成 IPv4 网络的部署

1.7.12.5.1. 为 IPv4 网络配置 hypervisor

以下信息仅适用于虚拟机环境。

1.7.12.5.1.1. 访问和部署虚拟 OpenShift Container Platform 集群的软件包

1. 要部署虚拟 OpenShift Container Platform 管理集群，请输入以下命令来访问所需的软件包：

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 输入以下命令启用并启动 Podman 服务：

```
systemctl enable --now podman
```

3. 要使用 kcli 部署 OpenShift Container Platform 管理集群和其他虚拟组件，请输入以下命令安装和配置 hypervisor：

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd
```

-

```
sudo dnf -y copr enable karmab/kcli

sudo dnf -y install kcli

sudo kcli create pool -p /var/lib/libvirt/images default

kcli create host kvm -H 127.0.0.1 local

sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images

kcli create network  -c 192.168.122.0/24 default
```

1.7.12.5.1.2. 启用网络管理器分配程序

1. 启用网络管理器分配程序，以确保虚拟机可以解析所需的域、路由和 registry。要启用网络管理器分配程序，在 /etc/NetworkManager/dispatcher.d/ 目录中，创建一个名为 forcedns 的脚本，其中包含以下内容，根据需要替换与您的环境匹配的值：

```
#!/bin/bash

export IP="192.168.126.1" (1)
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE (2)
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

(1) 修改 IP 变量，以指向托管 OpenShift Container Platform 管理集群的虚拟机监控程序接口的 IP 地址。

(2) 将 dns.base.domain.name 替换为 DNS 基域名称。

2. 创建该文件后，输入以下命令添加权限：

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. 运行脚本，并验证输出是否返回 ok。

1.7.12.5.1.3. 配置 BMC 访问

1. 配置 ksushy 以模拟虚拟机的基板管理控制器(BMC)。输入以下命令：

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
```

```
kcli create sushy-service --ssl --port 9000
```

```
sudo systemctl daemon-reload
```

```
systemctl enable --now ksushy
```

2. 输入以下命令测试该服务是否正常工作：

```
systemctl status ksushy
```

1.7.12.5.1.4. 配置 hypervisor 系统以允许连接

如果您在开发环境中工作，请将 hypervisor 系统配置为允许通过环境中不同虚拟网络进行各种连接。

注：如果您在生产环境中工作，您必须为 firewalld 服务建立正确的规则，并配置 SELinux 策略以维护安全环境。

- 对于 SELinux，输入以下命令：

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- 对于 firewalld，输入以下命令：

```
systemctl disable --now firewalld
```

- 对于 libvirtd，输入以下命令：

```
systemctl restart libvirtd
```

```
systemctl enable --now libvirtd
```

接下来，为您的环境配置 DNS。

1.7.12.5.1.5. 其他资源

- 有关 kcli 的更多信息，请参阅官方 kcli 文档。

1.7.12.5.2. 为 IPv4 网络配置 DNS

对于虚拟和裸机环境中断开连接和连接的环境，此步骤是必需的。虚拟和裸机环境之间的主要区别在于配置资源的位置。在裸机环境中，使用 Bind 等解决方案，而不是 dnsmasq 等轻量级解决方案。

- 要在虚拟环境中为 IPv4 网络配置 DNS，请参阅 默认入口和 DNS 行为。
- 要在裸机上为 IPv4 网络配置 DNS，请参阅 在裸机上配置 DNS。

接下来，部署 registry。

1.7.12.5.3. 为 IPv4 网络部署 registry

对于开发环境，使用 Podman 容器部署小型自托管注册表。对于生产环境，请使用企业托管的 registry，如 Red Hat Quay、Nexus 或 Artifactory。

要使用 Podman 部署小 registry，请完成以下步骤：

1. 以特权用户身份，访问 ${HOME} 目录并创建以下脚本：

```
#!/usr/bin/env bash

set -euo pipefail
```

```
PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" (1)

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi

dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -
x509 -days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER
$REGISTRY_PASSWORD
podman create --name registry --net host --security-opt label=disable --replace -v
```

```
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e
"REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -
e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1 将 PULL_SECRET 的位置替换为您的设置的适当位置。

2. 命名脚本文件 registry.sh 并保存它。运行脚本时，它会拉取以下信息：

   - registry 名称，基于虚拟机监控程序主机名
   - 所需的凭证和用户访问详情

3. 通过添加执行标记来调整权限，如下所示：

```
chmod u+x ${HOME}/registry.sh
```

4. 要在没有任何参数的情况下运行脚本，请输入以下命令：

```
${HOME}/registry.sh
```

   该脚本启动服务器。

5. 该脚本使用 systemd 服务来管理目的。如果需要管理脚本，您可以使用以下命令：

```
systemctl status
```

```
systemctl start
```

```
systemctl stop
```

registry 的根文件夹位于 /opt/registry 目录中，包含以下子目录：

- *certs 包含 TLS 证书。*
- *auth 包含凭据。*
- *数据 包含 registry 镜像。*
- *conf 包含 registry 配置。*

1.7.12.5.4. 为 IPv4 网络设置管理集群

要设置 OpenShift Container Platform 管理集群，您可以使用 dev-scripts，或者基于虚拟机，您可以使用 kcli 工具。以下说明特定于 kcli 工具。

1. *确保正确的网络已准备好在 hypervisor 中使用。网络将托管管理和托管集群。输入以下 kcli 命令：*

```
kcli create network -c 192.168.125.0/24 -P dhcp=false -P dns=false --domain
dns.base.domain.name ipv4
```

其中：

- *-c 指定网络的 CIDR。*
- *-p dhcp=false 将网络配置为禁用 DHCP，该 DHCP 由您配置的 dnsmasq 处理。*
- *-p dns=false 将网络配置为禁用 DNS，这也由您配置的 dnsmasq 处理。*
- *--domain 设置要搜索的域。*

- *dns.base.domain.name 是 DNS 基础域名。*

- *ipv4 是您要创建的网络的名称。*

2. *创建网络后，查看以下输出：*

```
[root@hypershiftbm ~]# kcli list network
Listing Networks...
+---------+--------+--------------------+-------+------------------+------+
| Network |  Type  |        Cidr        |  Dhcp |      Domain      | Mode |
+---------+--------+--------------------+-------+------------------+------+
| default | routed |   192.168.122.0/24 |  True |     default      | nat  |
| ipv4    | routed |   192.168.125.0/24 | False | dns.base.domain.name | nat  |
| ipv6    | routed | 2620:52:0:1306::/64 | False | dns.base.domain.name | nat  |
+---------+--------+--------------------+-------+------------------+------+
```

```
[root@hypershiftbm ~]# kcli info network ipv4
Providing information about network ipv4...
cidr: 192.168.125.0/24
dhcp: false
domain: dns.base.domain.name
mode: nat
plan: kvirt
type: routed
```

3. *确保 pull secret 和 kcli 计划文件已就位，以便您可以部署 OpenShift Container Platform 管理集群：*

a. *确认 pull secret 与 kcli 计划位于同一个文件夹中，并且 pull secret 文件名为 openshift_pull.json。*

b. *在 mgmt-compact-hub-ipv4.yaml 文件中添加 kcli 计划，其中包含 OpenShift Container Platform 定义。确保更新文件内容以匹配您的环境：*

```
plan: hub-ipv4
force: true
version: nightly
tag: "4.x.y-x86_64" (1)
cluster: "hub-ipv4"
domain: dns.base.domain.name
api_ip: 192.168.125.10
ingress_ip: 192.168.125.11
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
```

```
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: false
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: ipv4
users_dev: developer
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: ipv4-virtual-network
metallb_ranges:
- 192.168.125.150-192.168.125.190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:10
- hub-ctlplane-0:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:01
- hub-ctlplane-1:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:02
- hub-ctlplane-2:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:03
```

1

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

4. 要置备管理集群，请输入以下命令：

```
kcli create cluster openshift --pf mgmt-compact-hub-ipv4.yaml
```

1.7.12.5.4.1. 其他资源

- 有关 kcli 计划文件中的参数的更多信息，请参阅官方 kcli 文档中的 创建一个 parameters.yml。

1.7.12.5.5. 为 IPv4 网络配置 Web 服务器

您需要配置一个额外的 Web 服务器来托管与您要部署为托管集群的 OpenShift Container Platform 发行版本关联的 Red Hat Enterprise Linux CoreOS (RHCOS)镜像。

要配置 web 服务器，请完成以下步骤：

1. 输入以下命令从您要使用的 OpenShift Container Platform 发行版本中提取 openshift-install 二进制文件：

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTUR
E}"
```

2. 运行以下脚本：该脚本在 /opt/srv 目录中创建文件夹。文件夹包含用于置备 worker 节点的 RHCOS 镜像。

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*

## Run Webserver
podman ps --noheading | grep -q websrv-ai
```

```
if [[ $? == 0 ]];then
   echo "Launching Registry pod..."
   /usr/bin/podman run --name websrv-ai --net host -v
/opt/srv:/usr/local/apache2/htdocs:z quay.io/alosadag/httpd:p8080
fi
```

1

您可以在 OpenShift CI Release 页面上找到 ROOTFS_IMG_URL 值。

2

您可以在 OpenShift CI Release 页面上找到 LIVE_ISO_URL 值。

下载完成后，容器将运行，以托管 Web 服务器上的镜像。容器使用官方 HTTPd 镜像的一种变体，这使得它能够使用 IPv6 网络。

1.7.12.5.6. 为 IPv4 网络配置镜像镜像

镜像镜像是从外部 registry 获取镜像的过程，如 registry.redhat.com 或 quay.io，并将它们存储在私有 registry 中。

1.7.12.5.6.1. 完成镜像过程

注： 在 registry 服务器运行后启动镜像过程。

在以下步骤中，使用 oc-mirror 工具，它是一个使用 ImageSetConfiguration 对象的二进制文件。在文件中，您可以指定以下信息：

- 要镜像的 OpenShift Container Platform 版本。版本位于 quay.io 中。
- 要镜像的额外 Operator。单独选择软件包。
- 要添加到存储库的额外镜像。

要配置镜像镜像，请完成以下步骤：

1. 确保 ${HOME}/.docker/config.json 文件已更新为您要从中镜像 registry 以及您要将镜像推送到的私有 registry。

2. 通过使用以下示例，创建一个 ImageSetConfiguration 对象以用于镜像。根据需要替换与您的环境匹配的值：

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest (1)
mirror:
  platform:
    channels:
    - name: candidate-4.14
      minVersion: 4.x.y-x86_64 (2)
      maxVersion: 4.x.y-x86_64
      type: ocp
    graph: true
  additionalImages:
  - name: quay.io/karmab/origin-keepalived-ipfailover:latest
  - name: quay.io/karmab/kubectl:latest
  - name: quay.io/karmab/haproxy:latest
  - name: quay.io/karmab/mdns-publisher:latest
  - name: quay.io/karmab/origin-coredns:latest
  - name: quay.io/karmab/curl:latest
  - name: quay.io/karmab/kcli:latest
  - name: quay.io/user-name/trbsht:latest
  - name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
  - name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
    packages:
    - name: lvms-operator
    - name: local-storage-operator
    - name: odf-csi-addons-operator
    - name: odf-operator
    - name: mcg-operator
    - name: ocs-operator
    - name: metallb-operator
    - name: kubevirt-hyperconverged
```

(1) 将 dns.base.domain.name 替换为 DNS 基域名称。

(2) 将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

3. 输入以下命令启动镜像过程：

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

镜像过程完成后，您有一个名为 oc-mirror-workspace/results-XXXXXX/ 的新文件夹，其中包含 ICSP 和要应用到托管的集群的目录源。

4. 使用 oc adm release mirror 命令镜像 OpenShift Container Platform 的 nightly 或 CI 版本。输入以下命令：

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
  --from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
  --to=${REGISTRY}/openshift/release \
  --to-release-image=${REGISTRY}/openshift/release-images:4.x.y-x86_64
```

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

5. 按照在 *断开连接的网络上安装* 中的步骤镜像最新的多集群引擎 operator 镜像。

1.7.12.5.6.2. 在管理集群中应用对象

镜像过程完成后，您需要在管理集群中应用两个对象：

- 镜像内容源策略(ICSP)或镜像 Digest 镜像集(IDMS)
- 目录源

当使用 oc-mirror 工具时，输出工件位于名为 oc-mirror-workspace/results-XXXXXX/ 的目录中。

ICSP 或 IDMS 会启动一个 MachineConfig 更改，它不会重启节点，而是在每个节点上重启 kubelet。节点标记为 READY 后，您需要应用新生成的目录源。

目录源在 openshift-marketplace Operator 中启动操作，如下载目录镜像并处理它来检索该镜像中

包含的所有 PackageManifests。

1. 要检查新源，请使用新的 CatalogSource 作为源运行以下命令：

```
oc get packagemanifest
```

2. 要应用工件，请完成以下步骤：

   a. 输入以下命令来创建 ImageContentSourcePolicy (ICSP)或 IDMS 工件：

   ```
   oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
   ```

   b. 等待节点就绪，然后输入以下命令：

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

3. 镜像 OLM 目录并配置 hosed 集群以指向镜像。

   当您使用 management （默认）OLMCatalogPlacement 模式时，用于 OLM 目录的镜像流不会自动通过管理集群上的 ICSP 的信息覆盖信息。

   a. 如果 OLM 目录使用原始名称和标签正确镜像到内部 registry，请将 hypershift.openshift.io/olm-catalogs-is-registry-overrides 注解添加到 HostedCluster 资源。格式为 "sr1=dr1,sr2=dr2"，其中源 registry 字符串是一个键，目标 registry 是一个值。

   b. 要绕过 OLM 目录镜像流机制，请在 HostedCluster 资源上使用以下四个注解来直接指定用于 OLM operator 目录的四个镜像的地址：

      - hypershift.openshift.io/certified-operators-catalog-image
      - hypershift.openshift.io/community-operators-catalog-image

- *hypershift.openshift.io/redhat-marketplace-catalog-image*

- *hypershift.openshift.io/redhat-operators-catalog-image*

在这种情况下，镜像流不会被创建，当刷新内部镜像以拉取 Operator 更新时，您必须更新注解的值。

注：如果需要覆盖机制，则需要四个默认目录源的所有四个值。

1.7.12.5.6.3. 其他资源

- *如果您在虚拟环境中工作，请在配置镜像后确保满足 OpenShift Virtualization 上托管 control plane 的先决条件。*

- *如需有关 OpenShift Container Platform 每天镜像或 CI 版本的更多信息，请参阅使用 oc-mirror 插件为断开连接的安装 mirror 镜像。*

1.7.12.5.7. 为 IPv4 网络部署多集群引擎 operator

multicluster engine operator 在供应商间部署集群时扮演了关键角色。如果已安装 Red Hat Advanced Cluster Management，则不需要安装 multicluster engine operator，因为它会被自动安装。

如果您没有安装 multicluster engine operator，请参阅以下文档以了解安装它的先决条件和步骤：

- *关于多集群引擎 operator 的集群生命周期*

- *安装和升级多集群引擎 Operator*

1.7.12.5.7.1. 部署 AgentServiceConfig 资源

AgentServiceConfig 自定义资源是 Assisted Service add-on 的基本组件，它是 multicluster engine operator 的一部分。它负责裸机集群部署。启用附加组件后，您要部署 AgentServiceConfig 资源来配置附加组件。

除了配置 AgentServiceConfig 资源外，还需要包含额外的配置映射，以确保多集群引擎 Operator 在断开连接的环境中正常工作。

1. 通过添加以下配置映射来配置自定义 registry，其中包含用于自定义部署的断开连接的详情：

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" 1

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

1 将 dns.base.domain.name 替换为 DNS 基域名称。

对象包含两个字段：

- 自定义 CA：此字段包含加载到部署各种进程的证书颁发机构(CA)。
- Registry: Registries.conf 字段包含有关需要从镜像 registry 而不是原始源 registry

中消耗的镜像和命名空间的信息。

2. *通过添加 AssistedServiceConfig 对象来配置 Assisted Service，如下例所示：*

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  annotations:
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-
service-config (1)
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries (2)
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
  osImages: (3)
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img (4)
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-
0-live.x86_64.iso
    version: 414.92.202308281054-0
```

(1) *metadata.annotations["unsupported.agent-install.openshift.io/assisted-service-configmap"] 注解引用 Operator 用来自定义行为的配置映射名称。*

(2) *spec.mirrorRegistryRef.name 注解指向包含 Assisted Service Operator 使用断开连接的 registry 信息的配置映射。此配置映射在部署过程中添加这些资源。*

(3)

4

在 rootFSUrl 和 url 字段中，将 dns.base.domain.name 替换为 DNS 基域名称。

3. 通过将所有对象串联到一个文件中，并将它们应用到管理集群，以此部署所有对象。要做到这一点，请输入以下命令：

```
oc apply -f agentServiceConfig.yaml
```

该命令会触发两个 pod，如本示例输出所示：

```
assisted-image-service-0                1/1     Running   2          11d (1)
assisted-service-668b49548-9m7xw        2/2     Running   5          11d (2)
```

1

assisted-image-service pod 负责创建 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像模板，该模板针对您部署的每个集群自定义。

2

assisted-service 指的是 Operator。

1.7.12.5.8. 为 IPv4 网络配置 TLS 证书

涉及几个 TLS 证书，以便在断开连接的环境中配置托管的 control plane。要将证书颁发机构(CA)添加到管理集群中，您需要修改 OpenShift Container Platform control plane 和 worker 节点上的以下文件的内容：

- /etc/pki/ca-trust/extracted/pem/
- /etc/pki/ca-trust/source/anchors
- /etc/pki/tls/certs/

要在管理集群中添加 CA，请完成以下步骤：

1. 完成官方 OpenShift Container Platform 文档中的 更新 CA 捆绑包 中的步骤。该方法涉及使用 image-registry-operator，它将 CA 部署到 OpenShift Container Platform 节点。

2. 如果该方法不适用于您的情况，请检查管理集群中的 openshift-config 命名空间是否包含名为 user-ca-bundle 的配置映射。

   - 如果命名空间包含该配置映射，请输入以下命令：

   ```
   ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
   export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

   oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
   bundle.crt=${REGISTRY_CERT_PATH}
   ```

   - 如果命名空间不包含该配置映射，请输入以下命令：

   ```
   ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
   export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
   export TMP_FILE=$(mktemp)

   oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' >
   ${TMP_FILE}
   echo >> ${TMP_FILE}
   echo \#registry.$(hostname --long) >> ${TMP_FILE}
   cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
   oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
   bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
   ```

1.7.12.5.9. 为 IPv4 网络部署托管集群

托管的集群是一个 OpenShift Container Platform 集群，它带有托管在管理集群上的 control plane 和 API 端点。托管的集群包括控制平面和它的对应的数据平面。

虽然您可以使用 Red Hat Advanced Cluster Management 中的控制台创建托管集群，但以下步骤使用清单，这提供了修改相关工件的灵活性。

1.7.12.5.9.1. 部署托管集群对象

对于此过程，使用以下值：

- *HostedCluster name: hosted-ipv4*
- *HostedCluster 命名空间： 集群*
- *disconnected: true*
- *网络堆栈： IPv4*

通常，HyperShift Operator 会创建 HostedControlPlane 命名空间。但是，在这种情况下，您想要在 HyperShift Operator 开始协调 HostedCluster 对象前包含所有对象。然后，当 Operator 启动协调过程时，它可以找到所有对象。

1. *使用有关命名空间的以下信息创建 YAML 文件：*

```
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters-hosted-ipv4
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. *创建一个 YAML 文件，其中包含有关配置映射和 secret 的信息，以便在 HostedCluster 部署中包括：*

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
```

```
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
data:
  .dockerconfigjson: xxxxxxxxx
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-ipv4-pull-secret
  namespace: clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: sshkey-cluster-hosted-ipv4
  namespace: clusters
stringData:
  id_rsa.pub: ssh-rsa xxxxxxxxx
---
apiVersion: v1
data:
  key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-ipv4-etcd-encryption-key
  namespace: clusters
type: Opaque
```

3. 创建一个包含 RBAC 角色的 YAML 文件，以便辅助服务代理可以与托管 control plane 位于同一个 HostedControlPlane 命名空间中，仍由集群 API 管理：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-ipv4
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4.

4. 使用 HostedCluster 对象的信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-ipv4
  namespace: clusters
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: 1
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
  platform:
    agent:
      agentNamespace: clusters-hosted-ipv4
    type: Agent
  pullSecret:
    name: hosted-ipv4-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
  secretEncryption:
    aescbc:
      activeKey:
        name: hosted-ipv4-etcd-encryption-key
    type: aescbc
  services:
```

```
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-ipv4
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

其中 dns.base.domain.name 是 DNS 基础域名，而 4.x.y 是您要使用的 OpenShift Container Platform 版本。

1 *imageContentSources 部分包含托管集群中用户工作负载的镜像引用。*

5. *在 HostedCluster 对象中添加指向 OpenShift Container Platform 发行版本中 HyperShift Operator 发行版本的注解：*

   a. *输入以下命令来获取镜像有效负载：*

```
oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
```

其中 dns.base.domain.name 是 DNS 基础域名，而 4.x.y 是您要使用的 OpenShift Container Platform 版本。

b. 请参见以下输出：

```
hypershift
sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

c. 使用 OpenShift Container Platform Images 命名空间，输入以下命令检查摘要：

```
podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-
dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

其中 dns.base.domain.name 是 DNS 基础域名。

d. 请参见以下输出：

```
podman pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8
Trying to pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
Getting image source signatures
Copying blob d8190195889e skipped: already exists
Copying blob c71d2589fba7 skipped: already exists
Copying blob d4dc6e74b6ce skipped: already exists
Copying blob 97da74cc6d8f skipped: already exists
Copying blob b70007a560c9 done
Copying config 3a62961e6e done
Writing manifest to image destination
Storing signatures
3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
```

注： HostedCluster 对象中设置的发行镜像必须使用摘要而不是标签；例如，quay.io/openshift-release-dev/ocp-release-dev/ocp-release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0。

6. 通过将 YAML 文件中定义的所有对象串联成文件并根据管理集群应用，以创建您在 YAML 文件中定义的所有对象。要做到这一点，请输入以下命令：

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

7. *查看托管 **control plane** 的输出：*

```
NAME                                              READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc                    1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv                  2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq                       1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d                 1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht  1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8          1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf     1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf        1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts         1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7          1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4            1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr          1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp  1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j             1/1     Running   0          77s
dns-operator-6874b577f-9tc6b                      1/1     Running   0          94s
etcd-0                                            3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh    1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk                  1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7            1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg                 1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782               1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp                   4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk          1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck                   1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr                  1/1     Running   0          92s
oauth-openshift-79c7bf44bf-j975g                  2/2     Running   0          62s
olm-operator-767f9584c-4lcl2                      2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj              3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7     1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574         2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc  1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh                    2/2     Running   0          93s
```

8. *查看托管集群的输出：*

```
NAMESPACE   NAME          VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
clusters    hosted-ipv4             hosted-admin-kubeconfig   Partial    True        False         The hosted control plane is available
```

接下来，创建 NodePool 对象。

1.7.12.5.9.2. 为托管集群创建 NodePool 对象

NodePool 是与托管集群关联的一组可扩展的 worker 节点。NodePool 机器架构在特定池中保持一致，独立于 control plane 的机器架构。

1. 使用有关 NodePool 对象的以下信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-ipv4
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-ipv4
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
(3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) autoRepair 字段被设置为 false，因为如果删除了该节点，则不会重新创建该节点。

(2) upgradeType 设置为 InPlace，这表示升级过程中会重复使用相同的裸机节点。

(3) 此 NodePool 中包含的所有节点都基于以下 OpenShift Container Platform 版本：4.x.y-x86_64。将 dns.base.domain.name 替换为 DNS 基础域名，将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

(4) replicas 值设为 0，以便在需要时可以扩展它们。务必要将 NodePool 副本保持为 0，直到所有步骤都完成为止。

2. *运行以下命令来创建 NodePool 对象：*

```
oc apply -f 02-nodepool.yaml
```

3. *查看输出：*

```
NAMESPACE   NAME         CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-ipv4   hosted    0                        False        False        4.x.y-x86_64
```

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

接下来，创建一个 InfraEnv 资源。

1.7.12.5.9.3. 为托管集群创建 InfraEnv 资源

InfraEnv 资源是一个 Assisted Service 对象，其中包含基本详情，如 pullSecretRef 和 sshAuthorizedKey。这些详情用于创建为托管集群自定义的 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像。

1. *使用以下有关 InfraEnv 资源的信息创建 YAML 文件，根据需要替换值：*

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-ipv4
  namespace: clusters-hosted-ipv4
spec:
  pullSecretRef: (1)
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52
afsNkw0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZF
Bfq2XRUxo1wrRdor2z0Bbh93ytR+ZsDbbLIGngXaMa0Vbt+z74FqIcajbHTZ6zBmTpBVq
5RHtDPgKITdpE1fongp7+ZXQNBIkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmn
KLMW1VtCE6nJzEgWCufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmy
hw5Sj6YboA8WICs4z+TrFP89LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1
AmEy8QYNwpJQJNpSxzoKBjI73XxvPYYC/ljPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4
KeMu7eS2tC+Kn4deJ7KwwUycx8n6RHMeD8Qg9fITHCv3gmab8JKZJqN3hW1D378Juv
mIX4V0= (2)
```

1

pullSecretRef 引用与 InfraEnv 相同的命名空间中的配置映射引用，其中使用 pull secret。

2

sshAuthorizedKey 代表放在引导镜像中的 SSH 公钥。SSH 密钥允许以 core 用户身份访问 worker 节点。

2. *运行以下命令来创建 InfraEnv 资源：*

```
oc apply -f 03-infraenv.yaml
```

3. *请参见以下输出：*

```
NAMESPACE              NAME     ISO CREATED AT
clusters-hosted-ipv4   hosted   2023-09-11T15:14:10Z
```

接下来，创建 worker 节点。

1.7.12.5.9.4. 为托管集群创建 worker 节点

如果您在裸机平台上工作，创建 worker 节点对于确保正确配置了 BareMetalHost 的详情至关重要。

如果使用虚拟机，您可以完成以下步骤来创建 Metal3 Operator 消耗的空 worker 节点。为此，您可以使用 kcli。

1. *如果这不是您首次尝试创建 worker 节点，您必须首先删除之前的设置。要做到这一点，请输入以下命令删除计划：*

```
kcli delete plan hosted-ipv4
```

a. *当系统提示您确认是否要删除计划时，键入 y。*

'

b. 确认您看到一条消息，表示计划已被删除。

2. 输入以下命令来创建虚拟机：

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:11\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211 -P name=hosted-ipv4-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:12\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212 -P name=hosted-ipv4-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:13\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213 -P name=hosted-ipv4-worker2
```

```
systemctl restart ksushy
```

其中：

- start=False 表示虚拟机(VM)不会在创建时自动启动。
- uefi_legacy=true 表示您将使用 UEFI 传统引导来确保与之前的 UEFI 实现兼容。
- plan=hosted-dual 表示计划名称，将一组机器标识为集群。
- memory=8192 和 numcpus=16 是指定虚拟机资源的参数，包括 RAM 和 CPU。
- disks=[200,200] 表示您要在虚拟机中创建两个精简置备磁盘。
- nets=[{"name": "dual", "mac": "aa:aa:aa:02:13"}] 是网络详情，包括要连接到的网络详情以及主接口的 MAC 地址。
- 重启 ksushy 重启 ksushy 工具，以确保工具检测到您添加的虚拟机。

3. 查看生成的输出：

```
+--------------------+--------+------------------+------------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                        |    Plan    | Profile
|
+--------------------+--------+------------------+------------------------------------------------+-----
--------+---------+
|   hosted-worker0   |  down  |                  |                                                | hosted-ipv4 |
kvirt  |
|   hosted-worker1   |  down  |                  |                                                | hosted-ipv4 |
kvirt  |
|   hosted-worker2   |  down  |                  |                                                | hosted-ipv4 |
kvirt  |
+--------------------+--------+------------------+------------------------------------------------+-----
--------+---------+
```

接下来，为托管集群创建裸机主机。

1.7.12.5.9.5. 为托管集群创建裸机主机

裸机主机 是一个 openshift-machine-api 对象，其中包含物理和虚拟详情，以便可以通过 Metal3 Operator 识别它。这些详细信息与其他辅助服务对象关联，称为 代理。

重要： 在创建裸机主机和目标节点前，您必须创建虚拟机。

要创建裸机主机，请完成以下步骤：

1. 使用以下信息创建 YAML 文件：

   注： 因为至少有一个 secret 包含裸机主机凭证，所以您需要为每个 worker 节点至少创建两个对象。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-ipv4-worker0-bmc-secret
  namespace: clusters-hosted-ipv4
data:
  password: YWRtaW4=
  username: YWRtaW4=
```

```
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-ipv4-worker0
  namespace: clusters-hosted-ipv4
  labels:
    infraenvs.agent-install.openshift.io: hosted-ipv4 (1)
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-ipv4-worker0 (2)
spec:
  automatedCleaningMode: disabled (3)
  bmc:
    disableCertificateVerification: true (4)
    address: redfish-
virtualmedia://[192.168.125.1]:9000/redfish/v1/Systems/local/hosted-ipv4-worker0 (5)
    credentialsName: hosted-ipv4-worker0-bmc-secret (6)
  bootMACAddress: aa:aa:aa:aa:02:11 (7)
  online: true (8)
```

(1) infraenvs.agent-install.openshift.io 作为 Assisted Installer 和 BareMetalHost 对象之间的链接。

(2) bmac.agent-install.openshift.io/hostname 代表部署期间采用的节点名称。

(3) automatedCleaningMode 可防止节点被 Metal3 Operator 擦除。

(4) disableCertificateVerification 设置为 true，以从客户端绕过证书验证。

(5) address 表示 worker 节点的基板管理控制器(BMC)地址。

(6) credentialsName 指向存储了用户和密码凭证的 secret。

(7)

bootMACAddress 表示节点从其启动的接口 MACAddress。

8 *Online 定义 BareMetalHost 对象创建后节点的状态。*

2. *输入以下命令部署 BareMetalHost 对象：*

```
oc apply -f 04-bmh.yaml
```

在此过程中，您可以查看以下输出：

- *这个输出表示进程正在尝试访问节点：*

```
NAMESPACE        NAME           STATE         CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 registering               true     2s
clusters-hosted  hosted-worker1 registering               true     2s
clusters-hosted  hosted-worker2 registering               true     2s
```

- *此输出显示节点已启动：*

```
NAMESPACE        NAME           STATE         CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioning              true     16s
clusters-hosted  hosted-worker1 provisioning              true     16s
clusters-hosted  hosted-worker2 provisioning              true     16s
```

- *此输出显示节点成功启动：*

```
NAMESPACE        NAME           STATE         CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioned              true     67s
clusters-hosted  hosted-worker1 provisioned              true     67s
clusters-hosted  hosted-worker2 provisioned              true     67s
```

3. *节点启动后，请注意命名空间中的代理，如下例所示：*

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411             true       auto-assign
```

```
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412          true      auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413          true      auto-assign
```

代理代表可用于安装的节点。要将节点分配到托管集群，请扩展节点池。

1.7.12.5.9.6. 扩展节点池

创建裸机主机后，其状态将从 **Registering** 变为 **Provisioning** 。节点以代理的 **LiveISO** 开头，以及名为 **agent** 的默认 **pod**。该代理负责从 **Assisted Service Operator** 接收安装 **OpenShift Container Platform** 有效负载的说明。

1. 要扩展节点池，请输入以下命令：

```
oc -n clusters scale nodepool hosted-ipv4 --replicas 3
```

2. 扩展过程完成后，请注意代理被分配给一个托管的集群：

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413   hosted    true       auto-assign
```

3. 另请注意设置了节点池副本：

```
NAMESPACE   NAME     CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted   hosted    3                       False         False         4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

将 **4.x.y** 替换为您要使用的 **OpenShift Container Platform** 版本。

4. 等待节点加入集群。在此过程中，代理在其阶段和状态上提供更新。

接下来，监控托管集群的部署。

1.7.12.5.10. 为 IPv4 网络完成托管集群部署

您可以从两个视角监控托管集群的部署：control plane 和 data plane。

1.7.12.5.10.1. 监控 control plane

在托管集群部署时，您可以输入以下命令来监控 control plane：

```
export KUBECONFIG=/root/.kcli/clusters/hub-ipv4/auth/kubeconfig
```

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n clusters-hosted-ipv4;echo;echo;oc get bmh -A;echo;echo;oc get agent -A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -A;echo;echo;oc get nodepool -A;echo;echo;"
```

这些命令提供有关以下工件的信息：

- HyperShift Operator
- HostedControlPlane pod
- 裸机主机
- 代理
- InfraEnv 资源
- HostedCluster 和 NodePool 资源

1.7.12.5.10.2. 监控数据平面

要监控 Operator 在部署过程中如何进行，请输入以下命令：

```
oc get secret -n clusters-hosted-ipv4 admin-kubeconfig -o jsonpath='{.data.kubeconfig}' |base64 -d > /root/hc_admin_kubeconfig.yaml
```

```
export KUBECONFIG=/root/hc_admin_kubeconfig.yaml
```

```
watch "oc get clusterversion,nodes,co"
```

这些命令提供有关以下工件的信息：

- 集群版本
- 节点，特别是有关节点是否加入集群
- 集群 Operator

1.7.12.6. 在 IPv6 网络中配置托管的 control plane

IPv6 网络配置目前被指定为 disconnected。此设计的主要原因是远程 registry 不适用于 IPv6。

查看以下步骤在 IPv6 网络上配置托管的 control plane：

1. 为 IPv6 网络配置虚拟机监控程序
2. 为 IPv6 网络配置 DNS
3. 为 IPv6 网络部署 registry
4. 为 IPv6 网络设置管理集群
5. 为 IPv6 网络配置 Web 服务器
6. 为 IPv6 网络配置镜像
7. 为 IPv6 网络部署多集群引擎 operator

8. 为 IPv6 网络配置 TLS 证书

9. 为 IPv6 网络部署托管集群

10. 完成 IPv6 网络的部署

1.7.12.6.1. 为 IPv6 网络配置 hypervisor

以下信息仅适用于虚拟机环境。

1.7.12.6.1.1. 访问和部署虚拟 OpenShift Container Platform 集群的软件包

1. 要部署虚拟 OpenShift Container Platform 管理集群，请输入以下命令来访问所需的软件包：

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 输入以下命令启用并启动 Podman 服务：

```
systemctl enable --now podman
```

3. 要使用 kcli 部署 OpenShift Container Platform 管理集群和其他虚拟组件，请输入以下命令安装和配置 hypervisor：

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd
```

```
sudo dnf -y copr enable karmab/kcli
```

```
sudo dnf -y install kcli
```

```
sudo kcli create pool -p /var/lib/libvirt/images default
```

```
kcli create host kvm -H 127.0.0.1 local
```

```
sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images
```

```
kcli create network  -c 192.168.122.0/24 default
```

1.7.12.6.1.2. 启用网络管理器分配程序

1. 启用网络管理器分配程序，以确保虚拟机可以解析所需的域、路由和 registry。要启用网络管理器分配程序，在 /etc/NetworkManager/dispatcher.d/ 目录中，创建一个名为 forcedns 的脚本，其中包含以下内容，根据需要替换与您的环境匹配的值：

```
#!/bin/bash

export IP="2620:52:0:1306::1" 1
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE 2
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

1 修改 IP 变量，以指向托管 OpenShift Container Platform 管理集群的虚拟机监控程序接口的 IP 地址。

2 将 dns.base.domain.name 替换为 DNS 基域名称。

2. 创建该文件后，输入以下命令添加权限：

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. 运行脚本，并验证输出是否返回 ok。

1.7.12.6.1.3. 配置 BMC 访问

1. 配置 ksushy 以模拟虚拟机的基板管理控制器(BMC)。输入以下命令：

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
kcli create sushy-service --ssl --ipv6 --port 9000
sudo systemctl daemon-reload
systemctl enable --now ksushy
```

2. 输入以下命令测试该服务是否正常工作：

```
systemctl status ksushy
```

1.7.12.6.1.4. 配置 hypervisor 系统以允许连接

如果您在开发环境中工作，请将 hypervisor 系统配置为允许通过环境中不同虚拟网络进行各种连接。

注： 如果您在生产环境中工作，您必须为 firewalld 服务建立正确的规则，并配置 SELinux 策略以维护安全环境。

- 对于 SELinux，输入以下命令：

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- 对于 firewalld，输入以下命令：

```
systemctl disable --now firewalld
```

- 对于 libvirtd，输入以下命令：

```
systemctl restart libvirtd
systemctl enable --now libvirtd
```

接下来，为您的环境配置 DNS。

1.7.12.6.1.5. 其他资源

- 有关 kcli 的更多信息，请参阅官方 kcli 文档。

1.7.12.6.2. 为 IPv6 网络配置 DNS

对于虚拟和裸机环境中断开连接和连接的环境，此步骤是必需的。虚拟和裸机环境之间的主要区别在于配置资源的位置。在裸机环境中，使用 Bind 等解决方案，而不是 dnsmasq 等轻量级解决方案。

- 要在虚拟环境中为 IPv6 网络配置 DNS，请参阅默认入口和 DNS 行为。
- 要在裸机上为 IPv6 网络配置 DNS，请参阅在裸机上配置 DNS。

接下来，部署 registry。

1.7.12.6.3. 为 IPv6 网络部署 registry

对于开发环境，使用 Podman 容器部署小型自托管注册表。对于生产环境，请使用企业托管的 registry，如 Red Hat Quay、Nexus 或 Artifactory。

要使用 Podman 部署小 registry，请完成以下步骤：

1. 以特权用户身份，访问 ${HOME} 目录并创建以下脚本：

```
#!/usr/bin/env bash

set -euo pipefail

PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" (1)

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi
```

```
dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -
x509 -days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER
$REGISTRY_PASSWORD
podman create --name registry --net host --security-opt label=disable --replace -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e
"REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -
e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1

将 PULL_SECRET 的位置替换为您的设置的适当位置。

2. 命名脚本文件 registry.sh 并保存它。运行脚本时，它会拉取以下信息：

   - registry 名称，基于虚拟机监控程序主机名
   - 所需的凭证和用户访问详情

3. 通过添加执行标记来调整权限，如下所示：

```
chmod u+x ${HOME}/registry.sh
```

4. 要在没有任何参数的情况下运行脚本，请输入以下命令：

```
${HOME}/registry.sh
```

   该脚本启动服务器。

5. 该脚本使用 systemd 服务来管理目的。如果需要管理脚本，您可以使用以下命令：

```
systemctl status
systemctl start
systemctl stop
```

registry 的根文件夹位于 /opt/registry 目录中，包含以下子目录：

- certs 包含 TLS 证书。
- auth 包含凭据。

- 数据 包含 registry 镜像。
- conf 包含 registry 配置。

1.7.12.6.4. 为 IPv6 网络设置管理集群

要设置 OpenShift Container Platform 管理集群，您可以使用 dev-scripts，或者基于虚拟机，您可以使用 kcli 工具。以下说明特定于 kcli 工具。

1. 确保正确的网络已准备好在 hypervisor 中使用。网络将托管管理和托管集群。输入以下 kcli 命令：

```
kcli create network -c 2620:52:0:1305::0/64 -P dhcp=false -P dns=false --domain
dns.base.domain.name --nodhcp ipv6
```

其中：

- -c 指定网络的 CIDR。
- -p dhcp=false 将网络配置为禁用 DHCP，该 DHCP 由您配置的 dnsmasq 处理。
- -p dns=false 将网络配置为禁用 DNS，这也由您配置的 dnsmasq 处理。
- --domain 设置要搜索的域。
- dns.base.domain.name 是 DNS 基础域名。
- ipv6 是您要创建的网络的名称。

2. 创建网络后，查看以下输出：

```
[root@hypershiftbm ~]# kcli list network
```

```
Listing Networks...
+---------+--------+--------------------+-------+-----------------+------+
| Network |  Type  |        Cidr        |  Dhcp |      Domain     | Mode |
+---------+--------+--------------------+-------+-----------------+------+
| default | routed |   192.168.122.0/24 |  True |     default     | nat  |
| ipv4    | routed |   192.168.125.0/24 | False | dns.base.domain.name | nat  |
| ipv4    | routed | 2620:52:0:1305::/64 | False | dns.base.domain.name | nat  |
+---------+--------+--------------------+-------+-----------------+------+
```

```
[root@hypershiftbm ~]# kcli info network ipv6
Providing information about network ipv6...
cidr: 2620:52:0:1305::/64
dhcp: false
domain: dns.base.domain.name
mode: nat
plan: kvirt
type: routed
```

3. **确保 pull secret 和 kcli 计划文件已就位，以便您可以部署 OpenShift Container Platform 管理集群：**

   a. **确认 pull secret 与 kcli 计划位于同一个文件夹中，并且 pull secret 文件名为 openshift_pull.json。**

   b. **在 mgmt-compact-hub-ipv6.yaml 文件中添加 kcli 计划，其中包含 OpenShift Container Platform 定义。确保更新文件内容以匹配您的环境：**

```
plan: hub-ipv6
force: true
version: nightly
tag: "4.x.y-x86_64"
cluster: "hub-ipv6"
ipv6: true
domain: dns.base.domain.name
api_ip: 2620:52:0:1305::2
ingress_ip: 2620:52:0:1305::3
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
```

```
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: false
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: ipv6
users_dev: developer
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: ipv6-virtual-network
metallb_ranges:
- 2620:52:0:1305::150-2620:52:0:1305::190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:10
- hub-ctlplane-0:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:01
- hub-ctlplane-1:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:02
- hub-ctlplane-2:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:03
```

4. *要置备管理集群，请输入以下命令：*

```
kcli create cluster openshift --pf mgmt-compact-hub-ipv6.yaml
```

1.7.12.6.4.1. 其他资源

- *有关 kcli 计划文件中的参数的更多信息，请参阅官方 kcli 文档中的 创建一个 parameters.yml。*

1.7.12.6.5. 为 IPv6 网络配置 Web 服务器

您需要配置一个额外的 Web 服务器来托管与您要部署为托管集群的 OpenShift Container Platform 发行版本关联的 Red Hat Enterprise Linux CoreOS (RHCOS)镜像。

要配置 web 服务器，请完成以下步骤：

1. *输入以下命令从您要使用的 OpenShift Container Platform 发行版本中提取 openshift-install 二进制文件：*

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTUR
E}"
```

2. *运行以下脚本：该脚本在 /opt/srv 目录中创建文件夹。文件夹包含用于置备 worker 节点的 RHCOS 镜像。*

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*

## Run Webserver
podman ps --noheading | grep -q websrv-ai
if [[ $? == 0 ]];then
    echo "Launching Registry pod..."
    /usr/bin/podman run --name websrv-ai --net host -v
/opt/srv:/usr/local/apache2/htdocs:z quay.io/alosadag/httpd:p8080
fi
```

1 *您可以在 OpenShift CI Release 页面上找到 ROOTFS_IMG_URL 值。*

2 *您可以在 OpenShift CI Release 页面上找到 LIVE_ISO_URL 值。*

下载完成后，容器将运行，以托管 Web 服务器上的镜像。容器使用官方 HTTPd 镜像的一种变体，这使得它能够使用 IPv6 网络。

1.7.12.6.6. 为 IPv6 网络配置镜像

镜像镜像是从外部 registry 获取镜像的过程，如 registry.redhat.com 或 quay.io，并将它们存储在私有 registry 中。

1.7.12.6.6.1. 完成镜像过程

注： 在 registry 服务器运行后启动镜像过程。

在以下步骤中，使用 oc-mirror 工具，它是一个使用 ImageSetConfiguration 对象的二进制文件。在文件中，您可以指定以下信息：

- 要镜像的 OpenShift Container Platform 版本。版本位于 quay.io 中。
- 要镜像的额外 Operator。单独选择软件包。
- 要添加到存储库的额外镜像。

要配置镜像镜像，请完成以下步骤：

1. 确保 ${HOME}/.docker/config.json 文件已更新为您要从中镜像 registry 以及您要将镜像推送到的私有 registry。
2. 通过使用以下示例，创建一个 ImageSetConfiguration 对象以用于镜像。根据需要替换与您的环境匹配的值：

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest (1)
mirror:
  platform:
```

```
  channels:
  - name: candidate-4.14
    minVersion: 4.x.y-x86_64
    maxVersion: 4.x.y-x86_64
    type: ocp
  graph: true
additionalImages:
- name: quay.io/karmab/origin-keepalived-ipfailover:latest
- name: quay.io/karmab/kubectl:latest
- name: quay.io/karmab/haproxy:latest
- name: quay.io/karmab/mdns-publisher:latest
- name: quay.io/karmab/origin-coredns:latest
- name: quay.io/karmab/curl:latest
- name: quay.io/karmab/kcli:latest
- name: quay.io/user-name/trbsht:latest
- name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
operators:
- catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
  packages:
  - name: lvms-operator
  - name: local-storage-operator
  - name: odf-csi-addons-operator
  - name: odf-operator
  - name: mcg-operator
  - name: ocs-operator
  - name: metallb-operator
```

1

将 dns.base.domain.name 替换为 DNS 基础域名，将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

3. **输入以下命令启动镜像过程：**

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

镜像过程完成后，您有一个名为 oc-mirror-workspace/results-XXXXXX/ 的新文件夹，其中包含 ICSP 和要应用到托管的集群的目录源。

4. **使用 oc adm release mirror 命令镜像 OpenShift Container Platform 的 nightly 或 CI 版本。输入以下命令：**

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
```

```
--from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
--to=${REGISTRY}/openshift/release \
--to-release-image=${REGISTRY}/openshift/release-images:4.x.y-x86_64
```

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

5. 按照在 断开连接的网络 上安装 中的步骤镜像最新的多集群引擎 operator 镜像。

1.7.12.6.6.2. 在管理集群中应用对象

镜像过程完成后，您需要在管理集群中应用两个对象：

- 镜像内容源策略(ICSP)或镜像 Digest 镜像集(IDMS)
- 目录源

当使用 oc-mirror 工具时，输出工件位于名为 oc-mirror-workspace/results-XXXXXX/ 的目录中。

ICSP 或 IDMS 会启动一个 MachineConfig 更改，它不会重启节点，而是在每个节点上重启 kubelet。节点标记为 READY 后，您需要应用新生成的目录源。

目录源在 openshift-marketplace Operator 中启动操作，如下载目录镜像并处理它来检索该镜像中包含的所有 PackageManifests。

1. 要检查新源，请使用新的 CatalogSource 作为源运行以下命令：

```
oc get packagemanifest
```

2. 要应用工件，请完成以下步骤：

   a. 输入以下命令来创建 ICSP 或 IDMS 工件：

```
oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
```

b. 等待节点就绪，然后输入以下命令：

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

1.7.12.6.6.3. 其他资源

- 如果您在虚拟环境中工作，请在配置镜像后确保满足 OpenShift Virtualization 上托管 control plane 的先决条件。
- 如需有关 OpenShift Container Platform 每天镜像或 CI 版本的更多信息，请参阅使用 oc-mirror 插件为断开连接的安装 mirror 镜像。

1.7.12.6.7. 为 IPv6 网络部署多集群引擎 operator

multicluster engine operator 在供应商间部署集群时扮演了关键角色。如果已安装 Red Hat Advanced Cluster Management，则不需要安装 multicluster engine operator，因为它会被自动安装。

如果您没有安装 multicluster engine operator，请参阅以下文档以了解安装它的先决条件和步骤：

- 关于多集群引擎 operator 的集群生命周期
- 安装和升级多集群引擎 Operator

1.7.12.6.7.1. 部署 AgentServiceConfig 资源

AgentServiceConfig 自定义资源是 Assisted Service add-on 的基本组件，它是 multicluster engine operator 的一部分。它负责裸机集群部署。启用附加组件后，您要部署 AgentServiceConfig 资源来配置附加组件。

除了配置 AgentServiceConfig 资源外，还需要包含额外的配置映射，以确保多集群引擎 Operator 在断开连接的环境中正常工作。

1. 通过添加以下配置映射来配置自定义 registry，其中包含用于自定义部署的断开连接的详情：

-

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" 1

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

1 将 dns.base.domain.name 替换为 DNS 基域名称。

对象包含两个字段：

- 自定义 CA：此字段包含加载到部署各种进程的证书颁发机构(CA)。
- Registry: Registries.conf 字段包含有关需要从镜像 registry 而不是原始源 registry 中消耗的镜像和命名空间的信息。

2. 通过添加 AssistedServiceConfig 对象来配置 Assisted Service，如下例所示：

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
```

```
metadata:
  annotations:
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-
service-config (1)
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries (2)
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
  osImages: (3)
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img (4)
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-
0-live.x86_64.iso
    version: 414.92.202308281054-0
```

(1) metadata.annotations["unsupported.agent-install.openshift.io/assisted-service-configmap"] 注解引用 Operator 用来自定义行为的配置映射名称。

(2) spec.mirrorRegistryRef.name 注解指向包含 Assisted Service Operator 使用断开连接的 registry 信息的配置映射。此配置映射在部署过程中添加这些资源。

(3) spec.osImages 字段包含可供此 Operator 部署的不同版本。这个字段是必须的。本例假定您已下载了 RootFS 和 LiveISO 文件。

(4) 在 rootFSUrl 和 url 字段中，将 dns.base.domain.name 替换为 DNS 基域名称。

3. 通过将所有对象串联到一个文件中，并将它们应用到管理集群，以此部署所有对象。要做到这一点，请输入以下命令：

```
oc apply -f agentServiceConfig.yaml
```

该命令会触发两个 pod，如本示例输出所示：

```
assisted-image-service-0                1/1     Running   2          11d (1)
assisted-service-668b49548-9m7xw        2/2     Running   5          11d (2)
```

(1) assisted-image-service pod 负责创建 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像模板，该模板针对您部署的每个集群自定义。

(2) assisted-service 指的是 Operator。

1.7.12.6.8. 为 IPv6 网络配置 TLS 证书

涉及几个 TLS 证书，以便在断开连接的环境中配置托管的 control plane。要将证书颁发机构(CA)添加到管理集群中，您需要修改 OpenShift Container Platform control plane 和 worker 节点上的以下文件的内容：

- /etc/pki/ca-trust/extracted/pem/
- /etc/pki/ca-trust/source/anchors
- /etc/pki/tls/certs/

要在管理集群中添加 CA，请完成以下步骤：

1. 完成官方 OpenShift Container Platform 文档中的 *更新 CA 捆绑包* 中的步骤。该方法涉及使用 image-registry-operator，它将 CA 部署到 OpenShift Container Platform 节点。

2. 如果该方法不适用于您的情况，请检查管理集群中的 **openshift-config** 命名空间是否包含名为 **user-ca-bundle** 的配置映射。

- 如果命名空间包含该配置映射，请输入以下命令：

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${REGISTRY_CERT_PATH}
```

- 如果命名空间不包含该配置映射，请输入以下命令：

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
export TMP_FILE=$(mktemp)

oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' >
${TMP_FILE}
echo >> ${TMP_FILE}
echo \#registry.$(hostname --long) >> ${TMP_FILE}
cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
```

1.7.12.6.9. 为 IPv6 网络部署托管集群

托管的集群是一个 OpenShift Container Platform 集群，它带有托管在管理集群上的 control plane 和 API 端点。托管的集群包括控制平面和它的对应的数据平面。

虽然您可以使用 Red Hat Advanced Cluster Management 中的控制台创建托管集群，但以下步骤使用清单，这提供了修改相关工件的灵活性。

1.7.12.6.9.1. 部署托管集群对象

对于此过程，使用以下值：

- HostedCluster name: hosted-ipv6

- *HostedCluster 命名空间： 集群*
- *disconnected: true*
- *网络堆栈： IPv6*

通常，HyperShift Operator 会创建 HostedControlPlane 命名空间。但是，在这种情况下，您想要在 HyperShift Operator 开始协调 HostedCluster 对象前包含所有对象。然后，当 Operator 启动协调过程时，它可以找到所有对象。

1. *使用有关命名空间的以下信息创建 YAML 文件：*

```
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters-hosted-ipv6
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. *创建一个 YAML 文件，其中包含有关配置映射和 secret 的信息，以便在 HostedCluster 部署中包括：*

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
```

```
  data:
    .dockerconfigjson: xxxxxxxxx
  kind: Secret
  metadata:
    creationTimestamp: null
    name: hosted-ipv6-pull-secret
    namespace: clusters
  ---
  apiVersion: v1
  kind: Secret
  metadata:
    name: sshkey-cluster-hosted-ipv6
    namespace: clusters
  stringData:
    id_rsa.pub: ssh-rsa xxxxxxxxx
  ---
  apiVersion: v1
  data:
    key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
  kind: Secret
  metadata:
    creationTimestamp: null
    name: hosted-ipv6-etcd-encryption-key
    namespace: clusters
  type: Opaque
```

3. 创建一个包含 RBAC 角色的 YAML 文件，以便辅助服务代理可以与托管 control plane 位于同一个 HostedControlPlane 命名空间中，仍由集群 API 管理：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-ipv6
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4. 使用 HostedCluster 对象的信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-ipv6
  namespace: clusters
  annotations:
```

```
    hypershift.openshift.io/control-plane-operator-image: registry.ocp-edge-cluster-
0.qe.lab.redhat.com:5005/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d8997
5cf5f7a73801b2c06c639bf664876611 7f8
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: 1
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
  platform:
    agent:
      agentNamespace: clusters-hosted-ipv6
    type: Agent
  pullSecret:
    name: hosted-ipv6-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
  secretEncryption:
    aescbc:
      activeKey:
        name: hosted-ipv6-etcd-encryption-key
    type: aescbc
  services:
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
```

```
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-ipv6
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

其中 dns.base.domain.name 是 DNS 基础域名，而 4.x.y 是您要使用的 OpenShift Container Platform 版本。

1 *imageContentSources 部分包含托管集群中用户工作负载的镜像引用。*

5. *在 HostedCluster 对象中添加指向 OpenShift Container Platform 发行版本中 HyperShift Operator 发行版本的注解：*

   a. *输入以下命令来获取镜像有效负载：*

```
oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
```

其中 dns.base.domain.name 是 DNS 基础域名，而 4.x.y 是您要使用的 OpenShift Container Platform 版本。

.

b. 请参见以下输出：

```
hypershift
sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

c. 使用 OpenShift Container Platform Images 命名空间，输入以下命令检查摘要：

```
podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-
dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

其中 dns.base.domain.name 是 DNS 基础域名。

d. 请参见以下输出：

```
podman pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8
Trying to pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
Getting image source signatures
Copying blob d8190195889e skipped: already exists
Copying blob c71d2589fba7 skipped: already exists
Copying blob d4dc6e74b6ce skipped: already exists
Copying blob 97da74cc6d8f skipped: already exists
Copying blob b70007a560c9 done
Copying config 3a62961e6e done
Writing manifest to image destination
Storing signatures
3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
```

注： HostedCluster 对象中设置的发行镜像必须使用摘要而不是标签；例如，quay.io/openshift-release-dev/ocp-release-dev/ocp-release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0。

6. 通过将 YAML 文件中定义的所有对象串联成文件并根据管理集群应用，以创建您在 YAML 文件中定义的所有对象。要做到这一点，请输入以下命令：

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

–

7.

查看托管 control plane 的输出：

```
NAME                                     READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc           1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv         2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq              1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d        1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht   1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8 1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf   1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf   1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts   1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7   1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4   1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr   1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp   1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j    1/1     Running   0          77s
dns-operator-6874b577f-9tc6b             1/1     Running   0          94s
etcd-0                                   3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh   1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk         1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7   1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg        1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782      1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp          4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk   1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck          1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr         1/1     Running   0          92s
oauth-openshift-79c7bf44bf-j975g         2/2     Running   0          62s
olm-operator-767f9584c-4lcl2             2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj     3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7   1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574   2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc   1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh           2/2     Running   0          93s
```

8.

查看托管集群的输出：

```
NAMESPACE   NAME         VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
clusters    hosted-ipv6              hosted-admin-kubeconfig   Partial    True        False         The hosted control plane is available
```

接下来，创建 NodePool 对象。

1.7.12.6.9.2. 为托管集群创建 NodePool 对象

NodePool 是与托管集群关联的一组可扩展的 worker 节点。NodePool 机器架构在特定池中保持一

致，独立于 control plane 的机器架构。

1. 使用有关 NodePool 对象的以下信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-ipv6
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-ipv6
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
(3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) autoRepair 字段被设置为 false，因为如果删除了该节点，则不会重新创建该节点。

(2) upgradeType 设置为 InPlace，这表示升级过程中会重复使用相同的裸机节点。

(3) 此 NodePool 中包含的所有节点都基于以下 OpenShift Container Platform 版本：4.x.y-x86_64。将 dns.base.domain.name 替换为 DNS 基础域名，将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

(4) replicas 值设为 0，以便在需要时可以扩展它们。务必要将 NodePool 副本保持为 0，直到所有步骤都完成为止。

2. 运行以下命令来创建 NodePool 对象：

```
oc apply -f 02-nodepool.yaml
```

3. 查看输出：

```
NAMESPACE   NAME         CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-ipv6   hosted    0                          False        False        4.x.y-x86_64
```

接下来，创建一个 InfraEnv 资源。

1.7.12.6.9.3. 为托管集群创建 InfraEnv 资源

InfraEnv 资源是一个 Assisted Service 对象，其中包含基本详情，如 pullSecretRef 和 sshAuthorizedKey。这些详情用于创建为托管集群自定义的 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像。

1. 使用以下有关 InfraEnv 资源的信息创建 YAML 文件，根据需要替换值：

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-ipv6
  namespace: clusters-hosted-ipv6
spec:
  pullSecretRef: 1
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52
afsNkw0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZF
Bfq2XRUxo1wrRdor2z0Bbh93ytR+ZsDbbLIGngXaMa0Vbt+z74FqIcajbHTZ6zBmTpBVq
5RHtDPgKITdpE1fongp7+ZXQNBIkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmn
KLMW1VtCE6nJzEgWCufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmy
hw5Sj6YboA8WICs4z+TrFP89LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1
AmEy8QYNwpJQJNpSxzoKBjI73XxvPYYC/ljPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4
KeMu7eS2tC+Kn4deJ7KwwUycx8n6RHMeD8Qg9fITHCv3gmab8JKZJqN3hW1D378Juv
mIX4V0= 2
```

1. pullSecretRef 引用与 InfraEnv 相同的命名空间中的配置映射引用，其中使用 pull secret。

2.

2. 运行以下命令来创建 **InfraEnv** 资源：

```
oc apply -f 03-infraenv.yaml
```

3. 请参见以下输出：

```
NAMESPACE              NAME     ISO CREATED AT
clusters-hosted-ipv6   hosted   2023-09-11T15:14:10Z
```

接下来，创建 **worker** 节点。

1.7.12.6.9.4. 为托管集群创建 worker 节点

如果您在裸机平台上工作，创建 **worker** 节点对于确保正确配置了 **BareMetalHost** 的详情至关重要。

如果使用虚拟机，您可以完成以下步骤来创建 **Metal3 Operator** 消耗的空 **worker** 节点。为此，您可以使用 **kcli** 工具。

1. 如果这不是您首次尝试创建 **worker** 节点，您必须首先删除之前的设置。要做到这一点，请输入以下命令删除计划：

```
kcli delete plan hosted-ipv6
```

   a. 当系统提示您确认是否要删除计划时，键入 **y**。

   b. 确认您看到一条消息，表示计划已被删除。

2. 输入以下命令来创建虚拟机：

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:11\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211 -P name=hosted-ipv6-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:12\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212 -P name=hosted-ipv6-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:13\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213 -P name=hosted-ipv6-worker2
```

```
systemctl restart ksushy
```

其中：

- start=False 表示虚拟机(VM)不会在创建时自动启动。
- uefi_legacy=true 表示您将使用 UEFI 传统引导来确保与之前的 UEFI 实现兼容。
- plan=hosted-dual 表示计划名称，将一组机器标识为集群。
- memory=8192 和 numcpus=16 是指定虚拟机资源的参数，包括 RAM 和 CPU。
- disks=[200,200] 表示您要在虚拟机中创建两个精简置备磁盘。
- nets=[{"name": "dual", "mac": "aa:aa:aa:02:13"}] 是网络详情，包括要连接到的网络详情以及主接口的 MAC 地址。
- 重启 ksushy 重启 ksushy 工具，以确保工具检测到您添加的虚拟机。

3. 查看生成的输出：

```
+--------------------+--------+------------------+-----------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                       |    Plan    | Profile
|
+--------------------+--------+------------------+-----------------------------------------------+-----
--------+---------+
|   hosted-worker0   |  down  |                  |                                               | hosted-ipv6 |
kvirt  |
```

```
|   hosted-worker1   | down |              |                                                  | hosted-ipv6 |
kvirt |
|   hosted-worker2   | down |              |                                                  | hosted-ipv6 |
kvirt |
+--------------------+------+--------------+--------------------------------------------------+-----
-------+---------+
```

接下来，为托管集群创建裸机主机。

1.7.12.6.9.5. 为托管集群创建裸机主机

裸机主机 是一个 openshift-machine-api 对象，其中包含物理和虚拟详情，以便可以通过 Metal3 Operator 识别它。这些详细信息与其他辅助服务对象关联，称为 代理。

重要：在创建裸机主机和目标节点前，您必须创建虚拟机。

要创建裸机主机，请完成以下步骤：

1. *使用以下信息创建 YAML 文件：*

   注：因为至少有一个 secret 包含裸机主机凭证，所以您需要为每个 worker 节点至少创建两个对象。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-ipv6-worker0-bmc-secret
  namespace: clusters-hosted-ipv6
data:
  password: YWRtaW4=
  username: YWRtaW4=
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-ipv6-worker0
  namespace: clusters-hosted-ipv6
  labels:
    infraenvs.agent-install.openshift.io: hosted-ipv6 (1)
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-ipv6-worker0 (2)
```

```
spec:
  automatedCleaningMode: disabled 3
  bmc:
    disableCertificateVerification: true 4
    address: redfish-
virtualmedia://[192.168.125.1]:9000/redfish/v1/Systems/local/hosted-ipv6-worker0 5
    credentialsName: hosted-ipv6-worker0-bmc-secret 6
  bootMACAddress: aa:aa:aa:aa:03:11 7
  online: true 8
```

1. *infraenvs.agent-install.openshift.io 作为 Assisted Installer 和 BareMetalHost 对象之间的链接。*

2. *bmac.agent-install.openshift.io/hostname 代表部署期间采用的节点名称。*

3. *automatedCleaningMode 可防止节点被 Metal3 Operator 擦除。*

4. *disableCertificateVerification 设置为 true，以从客户端绕过证书验证。*

5. *address 表示 worker 节点的基板管理控制器(BMC)地址。*

6. *credentialsName 指向存储了用户和密码凭证的 secret。*

7. *bootMACAddress 表示节点从其启动的接口 MAC 地址。*

8. *Online 定义 BareMetalHost 对象创建后节点的状态。*

2. *输入以下命令部署 BareMetalHost 对象：*

```
oc apply -f 04-bmh.yaml
```

在此过程中，您可以查看以下输出：

- 这个输出表示进程正在尝试访问节点：

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   registering             true     2s
clusters-hosted  hosted-worker1   registering             true     2s
clusters-hosted  hosted-worker2   registering             true     2s
```

- 此输出显示节点已启动：

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   provisioning            true     16s
clusters-hosted  hosted-worker1   provisioning            true     16s
clusters-hosted  hosted-worker2   provisioning            true     16s
```

- 此输出显示节点成功启动：

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   provisioned             true     67s
clusters-hosted  hosted-worker1   provisioned             true     67s
clusters-hosted  hosted-worker2   provisioned             true     67s
```

3. 节点启动后，请注意命名空间中的代理，如下例所示：

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411             true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412             true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413             true       auto-assign
```

代理代表可用于安装的节点。要将节点分配到托管集群，请扩展节点池。

1.7.12.6.9.6. 扩展节点池

创建裸机主机后，其状态将从 **Registering** 变为 **Provisioning**。节点以代理的 **LiveISO** 开头，以及名为 **agent** 的默认 **pod**。该代理负责从 **Assisted Service Operator** 接收安装 **OpenShift Container Platform** 有效负载的说明。

1. 要扩展节点池，请输入以下命令：

```
oc -n clusters scale nodepool hosted-ipv6 --replicas 3
```

2. 扩展过程完成后，请注意代理被分配给一个托管的集群：

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211   hosted    true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212   hosted    true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213   hosted    true       auto-assign
```

3. 另请注意设置了节点池副本：

```
NAMESPACE   NAME     CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION          UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted   hosted    3                False         False        4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

4. 等待节点加入集群。在此过程中，代理在其阶段和状态上提供更新。

接下来，监控托管集群的部署。

1.7.12.6.10. 为 IPv6 网络完成托管集群部署

您可以从两个视角监控托管集群的部署：control plane 和 data plane。

1.7.12.6.10.1. 监控 control plane

在托管集群部署时，您可以输入以下命令来监控 control plane：

```
export KUBECONFIG=/root/.kcli/clusters/hub-ipv4/auth/kubeconfig
```

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n clusters-hosted-ipv4;echo;echo;oc get bmh -
A;echo;echo;oc get agent -A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -
A;echo;echo;oc get nodepool -A;echo;echo;"
```

这些命令提供有关以下工件的信息：

- *HyperShift Operator*
- *HostedControlPlane pod*
- *裸机主机*
- *代理*
- *InfraEnv 资源*
- *HostedCluster 和 NodePool 资源*

1.7.12.6.10.2. 监控数据平面

要监控 Operator 在部署过程中如何进行，请输入以下命令：

```
oc get secret -n clusters-hosted-ipv4 admin-kubeconfig -o jsonpath='{.data.kubeconfig}' |base64 -d >
/root/hc_admin_kubeconfig.yaml
```

```
export KUBECONFIG=/root/hc_admin_kubeconfig.yaml
```

```
watch "oc get clusterversion,nodes,co"
```

这些命令提供有关以下工件的信息：

- *集群版本*
- *节点，特别是有关节点是否加入集群*

- *集群 Operator*

1.7.12.7. 在双栈网络上配置托管的 control plane （技术预览）

在托管的 control plane 中，断开连接的环境是一个没有连接到互联网的 OpenShift Container Platform 集群，它使用托管的 control plane 作为基础。您可以在断开连接的环境中的双栈网络上配置托管的 control plane，因为远程 registry 无法使用 IPv6。

查看以下步骤在双栈网络中配置托管的 control plane：

1. *为双栈网络配置 hypervisor*
2. *为双栈网络配置 DNS*
3. *为双栈网络部署 registry*
4. *为双栈网络设置管理集群*
5. *为双栈网络配置 Web 服务器*
6. *为双栈网络配置镜像镜像*
7. *为双栈网络部署多集群引擎 operator*
8. *为双栈网络配置 TLS 证书*
9. *为双栈网络部署托管集群*
10. *完成双栈网络的部署*

1.7.12.7.1. 为双栈网络配置 hypervisor

以下信息仅适用于虚拟机环境。

1.7.12.7.1.1. 访问和部署虚拟 OpenShift Container Platform 集群的软件包

1. 要部署虚拟 OpenShift Container Platform 管理集群，请输入以下命令来访问所需的软件包：

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 输入以下命令启用并启动 Podman 服务：

```
systemctl enable --now podman
```

3. 要使用 kcli 部署 OpenShift Container Platform 管理集群和其他虚拟组件，请输入以下命令安装和配置 hypervisor：

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd
```

```
sudo dnf -y copr enable karmab/kcli
```

```
sudo dnf -y install kcli
```

```
sudo kcli create pool -p /var/lib/libvirt/images default
```

```
kcli create host kvm -H 127.0.0.1 local
```

```
sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images
```

```
kcli create network  -c 192.168.122.0/24 default
```

1.7.12.7.1.2. 启用网络管理器分配程序

1. 启用网络管理器分配程序，以确保虚拟机可以解析所需的域、路由和 registry。要启用网络管理器分配程序，在 /etc/NetworkManager/dispatcher.d/ 目录中，创建一个名为 forcedns 的

脚本，其中包含以下内容，根据需要替换与您的环境匹配的值：

```
#!/bin/bash

export IP="192.168.126.1" (1)
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE (2)
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

(1) 修改 IP 变量，以指向托管 OpenShift Container Platform 管理集群的虚拟机监控程序接口的 IP 地址。

(2) 将 dns.base.domain.name 替换为 DNS 基域名称。

2. 创建该文件后，输入以下命令添加权限：

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. 运行脚本，并验证输出是否返回 ok。

1.7.12.7.1.3. 配置 BMC 访问

1. 配置 ksushy 以模拟虚拟机的基板管理控制器(BMC)。输入以下命令：

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
```

```
kcli create sushy-service --ssl --ipv6 --port 9000
```

```
sudo systemctl daemon-reload
```

```
systemctl enable --now ksushy
```

2. 输入以下命令测试该服务是否正常工作：

```
systemctl status ksushy
```

1.7.12.7.1.4. 配置 hypervisor 系统以允许连接

如果您在开发环境中工作，请将 hypervisor 系统配置为允许通过环境中不同虚拟网络进行各种连接。

注： 如果您在生产环境中工作，您必须为 firewalld 服务建立正确的规则，并配置 SELinux 策略以维护安全环境。

- 对于 SELinux，输入以下命令：

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- 对于 firewalld，输入以下命令：

```
systemctl disable --now firewalld
```

- 对于 libvirtd，输入以下命令：

```
systemctl restart libvirtd
```

```
systemctl enable --now libvirtd
```

接下来，为您的环境配置 DNS。

1.7.12.7.1.5. 其他资源

- 有关 kcli 的更多信息，请参阅官方 kcli 文档。

1.7.12.7.2. 为双栈网络配置 DNS

对于虚拟和裸机环境中的断开连接和连接的环境，配置 DNS 是强制的。虚拟和裸机环境之间的主要区别在于配置资源的位置。在非虚拟环境中，使用 Bind 等解决方案，而不是 dnsmasq 等轻量级解决方

案。

- *要在虚拟环境中为双栈网络配置 DNS，请参阅 默认入口和 DNS 行为。*
- *要在裸机上为双栈网络配置 DNS，请参阅 在裸机上配置 DNS。*

接下来，部署 registry。

1.7.12.7.3. 为双栈网络部署 registry

对于开发环境，使用 Podman 容器部署小型自托管注册表。对于生产环境，部署企业级托管的 registry，如 Red Hat Quay、Nexus 或 Artifactory。

要使用 Podman 部署小 registry，请完成以下步骤：

1. *以特权用户身份，访问 ${HOME} 目录并创建以下脚本：*

```
#!/usr/bin/env bash

set -euo pipefail

PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" (1)

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi

dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
```

```
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -
x509 -days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER
$REGISTRY_PASSWORD
podman create --name registry --net host --security-opt label=disable --replace -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e
"REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -
e "REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1 将 PULL_SECRET 的位置替换为您的设置的适当位置。

2. 命名脚本文件 registry.sh 并保存它。运行脚本时，它会拉取以下信息：

- registry 名称，基于虚拟机监控程序主机名

- 所需的凭证和用户访问详情

3. 通过添加执行标记来调整权限，如下所示：

```
chmod u+x ${HOME}/registry.sh
```

4. 要在没有任何参数的情况下运行脚本，请输入以下命令：

```
${HOME}/registry.sh
```

该脚本启动服务器。

5. 该脚本使用 systemd 服务来管理目的。如果需要管理脚本，您可以使用以下命令：

```
systemctl status
systemctl start
systemctl stop
```

registry 的根文件夹位于 /opt/registry 目录中，包含以下子目录：

- certs 包含 TLS 证书。
- auth 包含凭据。
- 数据 包含 registry 镜像。
- conf 包含 registry 配置。

1.7.12.7.4. 为双栈网络设置管理集群

要设置 OpenShift Container Platform 管理集群，您可以使用 dev-scripts，或者基于虚拟机，您可

以使用 kcli 工具。以下说明特定于 kcli 工具。

1. *确保正确的网络已准备好在 hypervisor 中使用。网络将托管管理和托管集群。输入以下 kcli 命令：*

```
kcli create network -c 192.168.126.0/24 -P dhcp=false -P dns=false -d 2620:52:0:1306::0/64
--domain dns.base.domain.name --nodhcp dual
```

其中：

- *-c 指定网络的 CIDR。*
- *-p dhcp=false 将网络配置为禁用 DHCP，该 DHCP 由您配置的 dnsmasq 处理。*
- *-p dns=false 将网络配置为禁用 DNS，这也由您配置的 dnsmasq 处理。*
- *--domain 设置要搜索的域。*
- *dns.base.domain.name 是 DNS 基础域名。*
- *dual 是您要创建的网络的名称。*

2. *创建网络后，查看以下输出：*

```
[root@hypershiftbm ~]# kcli list network
Listing Networks...
+---------+--------+--------------------+-------+------------------+------+
| Network |  Type  |        Cidr        |  Dhcp |      Domain      | Mode |
+---------+--------+--------------------+-------+------------------+------+
| default | routed |   192.168.122.0/24 |  True |     default      | nat  |
| ipv4    | routed | 2620:52:0:1306::/64 | False | dns.base.domain.name | nat  |
| ipv4    | routed |   192.168.125.0/24  | False | dns.base.domain.name | nat  |
| ipv6    | routed | 2620:52:0:1305::/64 | False | dns.base.domain.name | nat  |
+---------+--------+--------------------+-------+------------------+------+
```

```
[root@hypershiftbm ~]# kcli info network ipv6
Providing information about network ipv6...
```

```
cidr: 2620:52:0:1306::/64
dhcp: false
domain: dns.base.domain.name
mode: nat
plan: kvirt
type: routed
```

3. **确保 pull secret 和 kcli 计划文件已就位，以便您可以部署 OpenShift Container Platform 管理集群：**

   a. **确认 pull secret 与 kcli 计划位于同一个文件夹中，并且 pull secret 文件名为 openshift_pull.json。**

   b. **在 mgmt-compact-hub-dual.yaml 文件中添加 kcli 计划，其中包含 OpenShift Container Platform 定义。确保更新文件内容以匹配您的环境：**

```
plan: hub-dual
force: true
version: stable
tag: "4.x.y-x86_64" (1)
cluster: "hub-dual"
dualstack: true
domain: dns.base.domain.name
api_ip: 192.168.126.10
ingress_ip: 192.168.126.11
service_networks:
- 172.30.0.0/16
- fd02::/112
cluster_networks:
- 10.132.0.0/14
- fd01::/48
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: true
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
```

```
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: dual
users_dev: developer
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: dual-virtual-network
metallb_ranges:
- 192.168.126.150-192.168.126.190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:07
- hub-ctlplane-0:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:01
- hub-ctlplane-1:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:02
- hub-ctlplane-2:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:03
```

1

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

4. 要置备管理集群，请输入以下命令：

```
kcli create cluster openshift --pf mgmt-compact-hub-dual.yaml
```

接下来，配置 Web 服务器。

1.7.12.7.4.1. 其他资源

- 有关 kcli 计划文件中的参数的更多信息，请参阅官方 kcli 文档中的 创建一个

parameters.yml。

1.7.12.7.5. 为双栈网络配置 Web 服务器

您需要配置一个额外的 Web 服务器来托管与您要部署为托管集群的 OpenShift Container Platform 发行版本关联的 Red Hat Enterprise Linux CoreOS (RHCOS)镜像。

要配置 web 服务器，请完成以下步骤：

1. *输入以下命令从您要使用的 OpenShift Container Platform 发行版本中提取 openshift-install 二进制文件：*

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTUR
E}"
```

2. *运行以下脚本：该脚本在 /opt/srv 目录中创建文件夹。文件夹包含用于置备 worker 节点的 RHCOS 镜像。*

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*

## Run Webserver
podman ps --noheading | grep -q websrv-ai
if [[ $? == 0 ]];then
    echo "Launching Registry pod..."
    /usr/bin/podman run --name websrv-ai --net host -v
/opt/srv:/usr/local/apache2/htdocs:z quay.io/alosadag/httpd:p8080
fi
```

1 *您可以在 OpenShift CI Release 页面上找到 ROOTFS_IMG_URL 值。*

2

您可以在 OpenShift CI Release 页面上找到 LIVE_ISO_URL 值。

下载完成后，容器将运行，以托管 Web 服务器上的镜像。容器使用官方 HTTPd 镜像的一种变体，这使得它能够使用 IPv6 网络。

1.7.12.7.6. 为双栈网络配置镜像镜像

镜像镜像是从外部 registry 获取镜像的过程，如 registry.redhat.com 或 quay.io，并将它们存储在私有 registry 中。

1.7.12.7.6.1. 完成镜像过程

注： 在 registry 服务器运行后启动镜像过程。

在以下步骤中，使用 oc-mirror 工具，它是一个使用 ImageSetConfiguration 对象的二进制文件。在文件中，您可以指定以下信息：

- 要镜像的 OpenShift Container Platform 版本。版本位于 quay.io 中。
- 要镜像的额外 Operator。单独选择软件包。
- 要添加到存储库的额外镜像。

要配置镜像镜像，请完成以下步骤：

1. 确保 ${HOME}/.docker/config.json 文件已更新为您要从中镜像 registry 以及您要将镜像推送到的私有 registry。
2. 通过使用以下示例，创建一个 ImageSetConfiguration 对象以用于镜像。根据需要替换与您的环境匹配的值：

```
apiVersion: mirror.openshift.io/v1alpha2
```

```
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest
mirror:
  platform:
    channels:
    - name: candidate-4.14
      minVersion: 4.x.y-x86_64 (1)
      maxVersion: 4.x.y-x86_64
      type: ocp
    graph: true
  additionalImages:
  - name: quay.io/karmab/origin-keepalived-ipfailover:latest
  - name: quay.io/karmab/kubectl:latest
  - name: quay.io/karmab/haproxy:latest
  - name: quay.io/karmab/mdns-publisher:latest
  - name: quay.io/karmab/origin-coredns:latest
  - name: quay.io/karmab/curl:latest
  - name: quay.io/karmab/kcli:latest
  - name: quay.io/user-name/trbsht:latest
  - name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
  - name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
    packages:
    - name: lvms-operator
    - name: local-storage-operator
    - name: odf-csi-addons-operator
    - name: odf-operator
    - name: mcg-operator
    - name: ocs-operator
    - name: metallb-operator
```

(1) ***将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。***

3. ***输入以下命令启动镜像过程：***

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

镜像过程完成后，您有一个名为 oc-mirror-workspace/results-XXXXXX/ 的新文件夹，其中包含 ICSP 和要应用到托管的集群的目录源。

4. ***使用 oc adm release mirror 命令镜像 OpenShift Container Platform 的 nightly 或 CI 版本。输入以下命令：***

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
  --from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
  --to=${REGISTRY}/openshift/release \
  --to-release-image=${REGISTRY}/openshift/release-
images:registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
```

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

5. 按照在 断开连接的网络 上安装 中的步骤镜像最新的多集群引擎 operator 镜像。

1.7.12.7.6.2. 在管理集群中应用对象

镜像过程完成后，您需要在管理集群中应用两个对象：

- 镜像内容源策略(ICSP)或镜像 Digest 镜像集(IDMS)
- 目录源

当使用 oc-mirror 工具时，输出工件位于名为 oc-mirror-workspace/results-XXXXXX/ 的目录中。

ICSP 或 IDMS 会启动一个 MachineConfig 更改，它不会重启节点，而是在每个节点上重启 kubelet。节点标记为 READY 后，您需要应用新生成的目录源。

目录源在 openshift-marketplace Operator 中启动操作，如下载目录镜像并处理它来检索该镜像中包含的所有 PackageManifests。

1. 要检查新源，请使用新的 CatalogSource 作为源运行以下命令：

```
oc get packagemanifest
```

2. 要应用工件，请完成以下步骤：

a. 输入以下命令来创建 ICSP 或 IDMS 工件：

```
oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
```

b. 等待节点就绪，然后输入以下命令：

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

接下来，部署多集群引擎 operator。

1.7.12.7.6.3. 其他资源

- 如果您在虚拟环境中工作，请在配置镜像后确保满足 OpenShift Virtualization 上托管 control plane 的先决条件。
- 如需有关 OpenShift Container Platform 每天镜像或 CI 版本的更多信息，请参阅使用 oc-mirror 插件为断开连接的安装 mirror 镜像。

1.7.12.7.7. 为双栈网络部署多集群引擎 operator

multicluster engine operator 在供应商间部署集群时扮演了关键角色。如果已安装 Red Hat Advanced Cluster Management，则不需要安装 multicluster engine operator，因为它会被自动安装。

如果您没有安装 multicluster engine operator，请参阅以下文档以了解安装它的先决条件和步骤：

- 关于多集群引擎 operator 的集群生命周期
- 安装和升级多集群引擎 Operator

1.7.12.7.7.1. 部署 AgentServiceConfig 资源

AgentServiceConfig 自定义资源是 Assisted Service add-on 的基本组件，它是 multicluster engine operator 的一部分。它负责裸机集群部署。启用附加组件后，您要部署 AgentServiceConfig 资源来配置附加组件。

除了配置 AgentServiceConfig 资源外，还需要包含额外的配置映射，以确保多集群引擎 Operator 在断开连接的环境中正常工作。

1. 通过添加以下配置映射来配置自定义 registry，其中包含用于自定义部署的断开连接的详情：

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" 1

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

1 将 dns.base.domain.name 替换为 DNS 基域名称。

对象包含两个字段：

- 自定义 CA：此字段包含加载到部署各种进程的证书颁发机构(CA)。
- Registry: Registries.conf 字段包含有关需要从镜像 registry 而不是原始源 registry

中消耗的镜像和命名空间的信息。

2. 通过添加 AssistedServiceConfig 对象来配置 Assisted Service，如下例所示：

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  annotations:
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-
service-config (1)
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries (2)
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
  osImages: (3)
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img (4)
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-
0-live.x86_64.iso
    version: 414.92.202308281054-0
```

(1) metadata.annotations["unsupported.agent-install.openshift.io/assisted-service-configmap"] 注解引用 Operator 用来自定义行为的配置映射名称。

(2) spec.mirrorRegistryRef.name 注解指向包含 Assisted Service Operator 使用断开连接的 registry 信息的配置映射。此配置映射在部署过程中添加这些资源。

(3)

4

在 rootFSUrl 和 url 字段中，将 dns.base.domain.name 替换为 DNS 基域名称。

3. 通过将所有对象串联到一个文件中，并将它们应用到管理集群，以此部署所有对象。要做到这一点，请输入以下命令：

```
oc apply -f agentServiceConfig.yaml
```

该命令会触发两个 pod，如本示例输出所示：

```
assisted-image-service-0                1/1     Running   2          11d (1)
assisted-service-668b49548-9m7xw        2/2     Running   5          11d (2)
```

1

assisted-image-service pod 负责创建 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像模板，该模板针对您部署的每个集群自定义。

2

assisted-service 指的是 Operator。

1.7.12.7.8. 为双栈网络配置 TLS 证书

涉及几个 TLS 证书，以便在断开连接的环境中配置托管的 control plane。要将证书颁发机构(CA)添加到管理集群中，您需要修改 OpenShift Container Platform control plane 和 worker 节点上的以下文件的内容：

- /etc/pki/ca-trust/extracted/pem/
- /etc/pki/ca-trust/source/anchors
- /etc/pki/tls/certs/

要在管理集群中添加 CA，请完成以下步骤：

1. *完成官方 OpenShift Container Platform 文档中的更新 CA 捆绑包 中的步骤。该方法涉及使用 image-registry-operator，它将 CA 部署到 OpenShift Container Platform 节点。*

2. *如果该方法不适用于您的情况，请检查管理集群中的 openshift-config 命名空间是否包含名为 user-ca-bundle 的配置映射。*

   - *如果命名空间包含该配置映射，请输入以下命令：*

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${REGISTRY_CERT_PATH}
```

   - *如果命名空间不包含该配置映射，请输入以下命令：*

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
export TMP_FILE=$(mktemp)

oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' >
${TMP_FILE}
echo >> ${TMP_FILE}
echo \#registry.$(hostname --long) >> ${TMP_FILE}
cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
```

1.7.12.7.9. 为双栈网络部署托管集群

托管的集群是一个 OpenShift Container Platform 集群，它带有托管在管理集群上的 control plane 和 API 端点。托管的集群包括控制平面和它的对应的数据平面。

虽然您可以使用 Red Hat Advanced Cluster Management 中的控制台创建托管集群，但以下步骤使用清单，这提供了修改相关工件的灵活性。

1.7.12.7.9.1. 部署托管集群对象

对于此过程，使用以下值：

- HostedCluster 名称： hosted-dual
- HostedCluster 命名空间： 集群
- disconnected: true
- 网络堆栈： Dual

通常，HyperShift Operator 会创建 HostedControlPlane 命名空间。但是，在这种情况下，您想要在 HyperShift Operator 开始协调 HostedCluster 对象前包含所有对象。然后，当 Operator 启动协调过程时，它可以找到所有对象。

1. 使用有关命名空间的以下信息创建 YAML 文件：

```
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters-hosted-dual
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. 创建一个 YAML 文件，其中包含有关配置映射和 secret 的信息，以便在 HostedCluster 部署中包括：

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
```

```
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
data:
  .dockerconfigjson: xxxxxxxxx
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-dual-pull-secret
  namespace: clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: sshkey-cluster-hosted-dual
  namespace: clusters
stringData:
  id_rsa.pub: ssh-rsa xxxxxxxxx
---
apiVersion: v1
data:
  key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-dual-etcd-encryption-key
  namespace: clusters
type: Opaque
```

3. 创建一个包含 RBAC 角色的 YAML 文件，以便辅助服务代理可以与托管 control plane 位于同一个 HostedControlPlane 命名空间中，仍由集群 API 管理：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-dual
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4. 使用 HostedCluster 对象的信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-dual
  namespace: clusters
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: 1
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release 2
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
    - cidr: fd01::/48
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
    - cidr: fd02::/112
  platform:
    agent:
      agentNamespace: clusters-hosted-dual
    type: Agent
  pullSecret:
    name: hosted-dual-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
3
  secretEncryption:
    aescbc:
      activeKey:
```

```
      name: hosted-dual-etcd-encryption-key
    type: aescbc
  services:
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-dual
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

1

imageContentSources 部分包含托管集群中用户工作负载的镜像引用。

2

在 YAML 文件中，将 dns.base.domain.name 替换为 DNS 基域名称。

3

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

5. *在 HostedCluster 对象中添加指向 OpenShift Container Platform 发行版本中 HyperShift Operator 发行版本的注解：*

a.

输入以下命令来获取镜像有效负载：

```
oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
```

其中 dns.base.domain.name 是 DNS 基础域名，而 4.x.y 是您要使用的 OpenShift Container Platform 版本。

b.

请参见以下输出：

```
hypershift
sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

c.

使用 OpenShift Container Platform Images 命名空间，输入以下命令检查摘要：

```
podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
```

其中 dns.base.domain.name 是 DNS 基础域名。

d.

请参见以下输出：

```
podman pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
Trying to pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
Getting image source signatures
Copying blob d8190195889e skipped: already exists
Copying blob c71d2589fba7 skipped: already exists
Copying blob d4dc6e74b6ce skipped: already exists
Copying blob 97da74cc6d8f skipped: already exists
Copying blob b70007a560c9 done
Copying config 3a62961e6e done
Writing manifest to image destination
Storing signatures
3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
```

注： HostedCluster 对象中设置的发行镜像必须使用摘要而不是标签；例如，quay.io/openshift-release-dev/ocp-release-dev/ocp-

release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0。

6. ***通过将 YAML 文件中定义的所有对象串联成文件并根据管理集群应用，以创建您在 YAML 文件中定义的所有对象。要做到这一点，请输入以下命令：***

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

7. ***查看托管 control plane 的输出：***

```
NAME                                     READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc           1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv         2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq              1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d        1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht   1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8 1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf   1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf 1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts 1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7 1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4   1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr 1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp   1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j    1/1     Running   0          77s
dns-operator-6874b577f-9tc6b             1/1     Running   0          94s
etcd-0                                   3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh   1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk         1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7   1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg        1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782      1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp          4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk 1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck          1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr         1/1     Running   0          92s
oauth-openshift-79c7bf44bf-j975g         2/2     Running   0          62s
olm-operator-767f9584c-4lcl2             2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj     3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7   1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574   2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc   1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh           2/2     Running   0          93s
```

8. ***查看托管集群的输出：***

```
NAMESPACE   NAME        VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
```

```
clusters   hosted-dual   hosted-admin-kubeconfig   Partial   True   False   The hosted control plane is available
```

接下来，创建 NodePool 对象。

1.7.12.7.9.2. 为托管集群创建 NodePool 对象

NodePool 是与托管集群关联的一组可扩展的 worker 节点。NodePool 机器架构在特定池中保持一致，独立于 control plane 的机器架构。

1. 使用有关 NodePool 对象的以下信息创建 YAML 文件，根据需要替换值：

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-dual
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-dual
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
(3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) autoRepair 字段被设置为 false，因为如果删除了该节点，则不会重新创建该节点。

(2) upgradeType 设置为 InPlace，这表示升级过程中会重复使用相同的裸机节点。

(3)

(4)

2. 运行以下命令来创建 NodePool 对象：

```
oc apply -f 02-nodepool.yaml
```

3. 查看输出：

```
NAMESPACE   NAME          CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-dual   hosted    0                False         False        4.x.y-x86_64
```

接下来，创建一个 InfraEnv 资源。

1.7.12.7.9.3. 为托管集群创建 InfraEnv 资源

InfraEnv 资源是一个 Assisted Service 对象，其中包含基本详情，如 pullSecretRef 和 sshAuthorizedKey。这些详情用于创建为托管集群自定义的 Red Hat Enterprise Linux CoreOS (RHCOS)引导镜像。

1. 使用以下有关 InfraEnv 资源的信息创建 YAML 文件，根据需要替换值：

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-dual
  namespace: clusters-hosted-dual
spec:
  pullSecretRef: 1
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52
afsNkw0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZF
Bfq2XRUxo1wrRdor2z0Bbh93ytR+ZsDbbLIGngXaMa0Vbt+z74FqIcajbHTZ6zBmTpBVq
5RHtDPgKITdpE1fongp7+ZXQNBIkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmn
KLMW1VtCE6nJzEgWCufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmy
hw5Sj6YboA8WICs4z+TrFP89LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1
AmEy8QYNwpJQJNpSxzoKBjI73XxvPYYC/IjPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4
KeMu7eS2tC+Kn4deJ7KwwUycx8n6RHMeD8Qg9fITHCv3gmab8JKZJqN3hW1D378Juv
mIX4V0= 2
```

1

2

sshAuthorizedKey 代表放在引导镜像中的 SSH 公钥。SSH 密钥允许以 core 用户身份访问 worker 节点。

2. 运行以下命令来创建 InfraEnv 资源：

```
oc apply -f 03-infraenv.yaml
```

3. 请参见以下输出：

```
NAMESPACE              NAME     ISO CREATED AT
clusters-hosted-dual   hosted   2023-09-11T15:14:10Z
```

接下来，创建 worker 节点。

1.7.12.7.9.4. 为托管集群创建 worker 节点

如果您在裸机平台上工作，创建 worker 节点对于确保正确配置了 BareMetalHost 的详情至关重要。

如果使用虚拟机，您可以完成以下步骤来为 Metal3 Operator 创建空的 worker 节点。为此，您可以使用 kcli 工具。

1. 如果这不是您首次尝试创建 worker 节点，您必须首先删除之前的设置。要做到这一点，请输入以下命令删除计划：

```
kcli delete plan hosted-dual
```

a. 当系统提示您确认是否要删除计划时，键入 y。

b. 确认您看到一条消息，表示计划已被删除。

2. 输入以下命令来创建虚拟机：

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\": \"aa:aa:aa:aa:11:01\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1101 -P name=hosted-dual-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\": \"aa:aa:aa:aa:11:02\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1102 -P name=hosted-dual-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\": \"aa:aa:aa:aa:11:03\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1103 -P name=hosted-dual-worker2
```

```
systemctl restart ksushy
```

其中：

- start=False 表示虚拟机(VM)不会在创建时自动启动。
- uefi_legacy=true 表示您将使用 UEFI 传统引导来确保与之前的 UEFI 实现兼容。
- plan=hosted-dual 表示计划名称，将一组机器标识为集群。
- memory=8192 和 numcpus=16 是指定虚拟机资源的参数，包括 RAM 和 CPU。
- disks=[200,200] 表示您要在虚拟机中创建两个精简置备磁盘。
- nets=[{"name": "dual", "mac": "aa:aa:aa:02:13"}] 是网络详情，包括要连接到的网络详情以及主接口的 MAC 地址。
- 重启 ksushy 重启 ksushy 工具，以确保工具检测到您添加的虚拟机。

3. 查看生成的输出：

```
+--------------------+--------+------------------+-------------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                         |    Plan    | Profile
|
+--------------------+--------+------------------+-------------------------------------------------+-----
--------+---------+
|   hosted-worker0   |  down  |                  |                                                 | hosted-dual |
kvirt  |
|   hosted-worker1   |  down  |                  |                                                 | hosted-dual |
kvirt  |
|   hosted-worker2   |  down  |                  |                                                 | hosted-dual |
kvirt  |
+--------------------+--------+------------------+-------------------------------------------------+-----
--------+---------+
```

接下来，为托管集群创建裸机主机。

1.7.12.7.9.5. 为托管集群创建裸机主机

裸机主机 是一个 openshift-machine-api 对象，其中包含物理和虚拟详情，以便可以通过 Metal3 Operator 识别它。这些详细信息与其他辅助服务对象关联，称为 代理。

重要： 在创建裸机主机和目标节点前，您必须创建虚拟机。

要创建裸机主机，请完成以下步骤：

1. 使用以下信息创建 YAML 文件：

   注： 因为至少有一个 secret 包含裸机主机凭证，所以您需要为每个 worker 节点至少创建两个对象。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-dual-worker0-bmc-secret
  namespace: clusters-hosted-dual
data:
  password: YWRtaW4=
  username: YWRtaW4=
```

```
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-dual-worker0
  namespace: clusters-hosted-dual
  labels:
    infraenvs.agent-install.openshift.io: hosted-dual (1)
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-dual-worker0 (2)
spec:
  automatedCleaningMode: disabled (3)
  bmc:
    disableCertificateVerification: true (4)
    address: redfish-
virtualmedia://[192.168.126.1]:9000/redfish/v1/Systems/local/hosted-dual-worker0 (5)
    credentialsName: hosted-dual-worker0-bmc-secret (6)
  bootMACAddress: aa:aa:aa:aa:02:11 (7)
  online: true (8)
```

(1) *infraenvs.agent-install.openshift.io 作为 Assisted Installer 和 BareMetalHost 对象之间的链接。*

(2) *bmac.agent-install.openshift.io/hostname 代表部署期间采用的节点名称。*

(3) *automatedCleaningMode 可防止节点被 Metal3 Operator 擦除。*

(4) *disableCertificateVerification 设置为 true，以从客户端绕过证书验证。*

(5)

(6) *credentialsName 指向存储了用户和密码凭证的 secret。*

(7) *bootMACAddress 表示节点从其启动的接口 MAC 地址。*

8

Online 定义 BareMetalHost 对象创建后节点的状态。

2. *输入以下命令部署 BareMetalHost 对象：*

```
oc apply -f 04-bmh.yaml
```

在此过程中，您可以查看以下输出：

- *这个输出表示进程正在尝试访问节点：*

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 registering             true         2s
clusters-hosted  hosted-worker1 registering             true         2s
clusters-hosted  hosted-worker2 registering             true         2s
```

- *此输出显示节点已启动：*

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioning            true         16s
clusters-hosted  hosted-worker1 provisioning            true         16s
clusters-hosted  hosted-worker2 provisioning            true         16s
```

- *此输出显示节点成功启动：*

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioned              true         67s
clusters-hosted  hosted-worker1 provisioned              true         67s
clusters-hosted  hosted-worker2 provisioned              true         67s
```

3. *节点启动后，请注意命名空间中的代理，如下例所示：*

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411                true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412                true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413                true       auto-assign
```

代理代表可用于安装的节点。要将节点分配到托管集群，请扩展节点池。

1.7.12.7.9.6. 扩展节点池

创建裸机主机后，其状态将从 Registering 变为 Provisioning 。节点以代理的 LiveISO 开头，以及名为 agent 的默认 pod。该代理负责从 Assisted Service Operator 接收安装 OpenShift Container Platform 有效负载的说明。

1. 要扩展节点池，请输入以下命令：

```
oc -n clusters scale nodepool hosted-dual --replicas 3
```

2. 扩展过程完成后，请注意代理被分配给一个托管的集群：

```
NAMESPACE        NAME                                 CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411   hosted    true      auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412   hosted    true      auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413   hosted    true      auto-assign
```

3. 另请注意设置了节点池副本：

```
NAMESPACE   NAME     CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                 UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted   hosted    3                       False        False        4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

将 4.x.y 替换为您要使用的 OpenShift Container Platform 版本。

4. 等待节点加入集群。在此过程中，代理在其阶段和状态上提供更新。

接下来，监控托管集群的部署。

1.7.12.7.10. 为双栈网络完成托管集群部署

您可以从两个视角监控托管集群的部署：control plane 和 data plane。

1.7.12.7.10.1. 使用 control plane 监控托管集群部署

您可以使用 control plane 监控托管集群部署。

1. 输入以下命令导出托管集群 kubeconfig 文件：

```
export KUBECONFIG=<path_to_hosted_cluster_kubeconfig>
```

2. 输入以下命令观察托管集群部署进度：

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n
<hosted_control_plane_namespace>;echo;echo;oc get bmh -A;echo;echo;oc get agent -
A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -A;echo;echo;oc get
nodepool -A;echo;echo;"
```

此命令提供有关以下工件的信息：

- HyperShift Operator
- HostedControlPlane pod
- 裸机主机
- 代理
- InfraEnv 资源
- HostedCluster 和 NodePool 资源

1.7.12.7.10.2. 使用数据平面监控托管集群部署

您可以使用数据平面来监控 Operator 在托管集群部署期间如何进行。

1. 输入以下命令为托管集群创建 kubeconfig 文件：

```
hcp create kubeconfig --name <hosted_cluster_name> --namespace
<hosted_cluster_namespace>
```

2. 输入以下命令导出托管集群 kubeconfig 文件：

```
export KUBECONFIG=<path_to_hosted_cluster_kubeconfig>
```

3. 输入以下命令观察集群版本、节点和集群 Operator 的状态：

```
watch "oc get clusterversion,nodes,co"
```

1.7.13. 手动导入托管的 control plane 集群

托管的集群会在托管的 control plane 可用后自动导入到多集群引擎 Operator。如果要手动导入托管集群，请完成以下步骤：

1. 在控制台中，点 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。

1.7.13.1. 在 AWS 上手动导入托管的 control plane 集群

您还可以通过完成以下步骤，使用命令行界面在 AWS 上导入托管的 control plane 集群：

1. 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: hypershift
  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。

2. 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。

3. 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。

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. 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>
```

1.7.13.2. 其他资源

- 有关禁用自动导入托管集群的说明，请参阅禁用将托管集群的自动导入到 multicluster engine operator。

1.7.13.3. 禁用将托管集群自动导入到多集群引擎 operator

在 control plane 可用时，托管的集群会自动导入到 multicluster engine operator，您可以禁用自动导入托管集群。

任何之前导入的托管集群都不会受到影响，即使您禁用了自动导入。当您升级到 multicluster engine operator 2.5 并启用自动导入时，如果 control plane 可用，所有没有导入的托管集群都会被自动导入。

注： 如果安装了 Red Hat Advanced Cluster Management，也会启用所有 Red Hat Advanced Cluster Management 附加组件。

当禁用了自动导入时，只有新创建的托管集群才会自动导入。已导入的托管集群不会受到影响。您仍然可以使用控制台或创建 ManagedCluster 和 KlusterletAddonConfig 自定义资源来手动导入集群。

要禁用托管集群的自动导入，请完成以下步骤：

1.

在 multicluster engine operator hub 集群中，打开 AddonDeploymentConfig 资源，以编辑 multicluster-engine 命名空间中的 hypershift-addon-deploy-config 规格。输入以下命令：

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

2. 在 spec.customizedVariables 部分中，添加 autoImportDisabled 变量，值设为 "true"，如下例所示：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  customizedVariables:
  - name: autoImportDisabled
    value: "true"
```

3. 要重新启用自动导入，请将 autoImportDisabled 变量的值设置为 "false"，或者从 AddonDeploymentConfig 资源中删除变量。

1.7.13.3.1. 其他资源

有关手动导入托管集群的步骤，请参阅 手动导入托管的 control plane 集群。

1.7.14. 启用或禁用托管的 control plane 功能

托管的 control plane 功能以及 hypershift-addon 受管集群附加组件会被默认启用。如果要禁用这个功能，或者禁用它并希望手动启用它，请参阅以下步骤：

- 手动启用托管的 control plane 功能
- 禁用托管的 control plane 功能

1.7.14.1. 手动启用托管的 control plane 功能

1. 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","enabled": true}]}}}' 1
```

1

默认 MultiClusterEngine 资源实例名称为 multiclusterengine，但您可以通过运行以下命令来从集群中获取 MultiClusterEngine 名称 ：$ oc get mce。

2. 运行以下命令，以验证 MultiClusterEngine 自定义资源中是否启用了 hypershift 和 hypershift-local-hosting 功能：

```
oc get mce multiclusterengine -o yaml 1
```

1

默认 MultiClusterEngine 资源实例名称为 multiclusterengine，但您可以通过运行以下命令来从集群中获取 MultiClusterEngine 名称 ：$ oc get mce。

输出类似以下示例：

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

1.7.14.1.1. 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>
```

使用您创建的文件的名称替换 filename。

3. 运行以下命令确认已安装 hypershift-addon：

```
oc get managedclusteraddons -n local-cluster hypershift-addon
```

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托管集群。

1.7.14.2. 禁用托管的 control plane 功能

您可以卸载 HyperShift Operator 并禁用托管的 control plane。当您禁用托管的 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。

1.7.14.2.1. 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}]}}}' 1
```

1

默认 MultiClusterEngine 资源实例名称为 multiclusterengine，但您可以通过运行以下命令来从集群中获取 MultiClusterEngine 名称 ：$ oc get mce。

注： 在禁用 hypershift-addon 后，您还可以从 multicluster engine operator 控制台为 local-cluster 禁用 hypershift-addon。

1.7.14.2.2. 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","enabled": false}]}}}' 1
```

1

默认 MultiClusterEngine 资源实例名称为 multiclusterengine，但您可以通过运行以下命令来从集群中获取 MultiClusterEngine 名称 ：$ oc get mce。

您可以运行以下命令来验证 MultiClusterEngine 自定义资源中禁用了 hypershift 和 hypershift-local-hosting 功能：

```
oc get mce multiclusterengine -o yaml 1
```

1

默认 MultiClusterEngine 资源实例名称为 multiclusterengine，但您可以通过运行以下命令来从集群中获取 MultiClusterEngine 名称 ：$ oc get mce。

请参阅以下示例，其中 hypershift 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
      enabled: false
    - name: hypershift-local-hosting
      enabled: false
```

1.7.14.3. 其他资源

- 在 AWS 上配置托管的 control plane 集群（技术预览）
- 在裸机上配置托管的 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。

注： 您还可以从集成控制台访问所有 API。在 local-cluster 视图中，进入到 Home > API Explorer 以探索 API 组。

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)*
- *ManagedServiceAccount API*

1.8.1. Clusters API

1.8.1.1. 概述

本文档介绍了与 Kubernetes operator 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 *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
Body	body *必需*	描述要创建集群的参数。	Cluster

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
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 operator 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

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

1.8.2.2.1.4. 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

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

1.8.2.2.2.4. 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** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	**clusterset_name** *必需*	要查询的集群集的名称。	字符串

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
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

类型	Name	描述	模式
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
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
apiVersion *必需*	字符串
kind *必需*	字符串
metadata *必需*	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

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

1.8.3.2.1.4. 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** *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
Body	**body** *必需*	描述要创建的 clustersetbinding 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
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
apiVersion *必需*	字符串
kind *必需*	字符串

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

spec

Name	模式
clusterSet *必需*	字符串

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
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** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
Body	**body** *可选*	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
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** *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

1.8.4.2.5.3. 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
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** *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

1.8.4.2.6.3. 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

HTTP 代码	描述	模式
503	服务不可用	无内容

1.8.5. ManagedServiceAccount API (v1alpha1) （已弃用）

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。

deprecated: v1alpha1 API 已被弃用。为获得最佳结果，请使用 v1beta1。

1.8.5.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.8.5.1.2. Tags

- *ManagedServiceAccounts .authentication.open-cluster-management.io': 创建和管理 ManagedServiceAccounts*

1.8.5.2. 路径

1.8.5.2.1. 创建 ManagedServiceAccount

> *POST /authentication.open-cluster-management.io/v1beta1/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

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
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

- ***managedserviceaccounts.authentication.open-cluster-management.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: v0.14.0
  name: managedserviceaccounts.authentication.open-cluster-management.io
spec:
  group: authentication.open-cluster-management.io
  names:
    kind: ManagedServiceAccount
    listKind: ManagedServiceAccountList
    plural: managedserviceaccounts
```

```
    singular: managedserviceaccount
  scope: Namespaced
  versions:
  - deprecated: true
    deprecationWarning: authentication.open-cluster-management.io/v1alpha1
ManagedServiceAccount
      is deprecated; use authentication.open-cluster-management.io/v1beta1
ManagedServiceAccount;
      version v1alpha1 will be removed in the next release
    name: v1alpha1
    schema:
      openAPIV3Schema:
        description: ManagedServiceAccount is the Schema for the managedserviceaccounts
          API
        properties:
          apiVersion:
            description: |-
              APIVersion defines the versioned schema of this representation of 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: 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
          ManagedServiceAccount
        properties:
          conditions:
            description: Conditions is the condition list.
            items:
              description: "Condition contains details for one aspect of the current
                state of this API Resource.\n---\nThis struct is intended for
                direct use as an array at the field path .status.conditions.  For
                example,\n\n\n\ttype FooStatus struct{\n\t    // Represents the
                observations of a foo's current state.\n\t    // Known .status.conditions.type
                are: \"Available\", \"Progressing\", and \"Degraded\"\n\t    //
                +patchMergeKey=type\n\t    // +patchStrategy=merge\n\t    // +listType=map\n\t
                \   // +listMapKey=type\n\t    Conditions []metav1.Condition
`json:\"conditions,omitempty\"
                patchStrategy:\"merge\" patchMergeKey:\"type\"
protobuf:\"bytes,1,rep,name=conditions\"`\n\n\n\t
                \   // other fields\n\t}"
              properties:
                lastTransitionTime:
                  description: |-
                    lastTransitionTime is the last time the condition transitioned from one status to
another.
                    This should be when the underlying condition changed.  If that is not known,
then using the time when the API field changed is acceptable.
                  format: date-time
                  type: string
                message:
                  description: |-
                    message is a human readable message indicating details about the transition.
                    This may be an empty string.
                  maxLength: 32768
                  type: string
                observedGeneration:
                  description: |-
                    observedGeneration represents the .metadata.generation that the condition was
set based upon.
                    For instance, if .metadata.generation is currently 12, but the
.status.conditions[x].observedGeneration is 9, the condition is out of date
```

```
                with respect to the current state of the instance.
              format: int64
              minimum: 0
              type: integer
            reason:
              description: |-
                reason contains a programmatic identifier indicating the reason for the
condition's last transition.
                Producers of specific condition types may define expected values and meanings
for this field,
                and whether the values are considered a guaranteed API.
                The value should be a CamelCase string.
                This 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.
                ---
                Many .condition.type values are consistent across resources like Available, but
because arbitrary conditions can be
                useful (see .node.status.conditions), the ability to deconflict is important.
                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 Secret,
which stores
          the CA certficate and token from the managed cluster.
        properties:
          lastRefreshTimestamp:
            description: |-
```

```
                LastRefreshTimestamp is the timestamp indicating when 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: false
    subresources:
      status: {}
  - name: v1beta1
    schema:
      openAPIV3Schema:
        description: ManagedServiceAccount is the Schema for the managedserviceaccounts
          API
        properties:
          apiVersion:
            description: |-
              APIVersion defines the versioned schema of this representation of 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: 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 before it
expires.
                      Deprecated: All ServiceAccount tokens will be rotated before they expire
regardless of this field.
```

```
              type: boolean
            validity:
              default: 8640h0m0s
              description: Validity is the duration of validity for requesting
                the signed ServiceAccount token.
              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
          ManagedServiceAccount
        properties:
          conditions:
            description: Conditions is the condition list.
            items:
              description: "Condition contains details for one aspect of the current
                state of this API Resource.\n---\nThis struct is intended for
                direct use as an array at the field path .status.conditions.  For
                example,\n\n\n\ttype FooStatus struct{\n\t    // Represents the
                observations of a foo's current state.\n\t    // Known .status.conditions.type
                are: \"Available\", \"Progressing\", and \"Degraded\"\n\t    //
                +patchMergeKey=type\n\t    // +patchStrategy=merge\n\t    // +listType=map\n\t
                \   // +listMapKey=type\n\t    Conditions []metav1.Condition
`json:\"conditions,omitempty\"
                patchStrategy:\"merge\" patchMergeKey:\"type\"
protobuf:\"bytes,1,rep,name=conditions\"`\n\n\n\t
                \   // other fields\n\t}"
              properties:
                lastTransitionTime:
                  description: |-
                    lastTransitionTime is the last time the condition transitioned from one status to
another.
                    This should be when the underlying condition changed.  If that is not known,
then using the time when the API field changed is acceptable.
                  format: date-time
                  type: string
                message:
                  description: |-
```

```
                message is a human readable message indicating details about the transition.
                This may be an empty string.
              maxLength: 32768
              type: string
            observedGeneration:
              description: |-
                observedGeneration represents the .metadata.generation that the condition was
set based upon.
                For instance, if .metadata.generation is currently 12, but the
.status.conditions[x].observedGeneration is 9, the condition is out of date
                with respect to the current state of the instance.
              format: int64
              minimum: 0
              type: integer
            reason:
              description: |-
                reason contains a programmatic identifier indicating the reason for the
condition's last transition.
                Producers of specific condition types may define expected values and meanings
for this field,
                and whether the values are considered a guaranteed API.
                The value should be a CamelCase string.
                This 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.
                ---
                Many .condition.type values are consistent across resources like Available, but
because arbitrary conditions can be
                useful (see .node.status.conditions), the ability to deconflict is important.
                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 Secret,
which stores
              the CA certficate and token from the managed cluster.
            properties:
              lastRefreshTimestamp:
                description: |-
                  LastRefreshTimestamp is the timestamp indicating when 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: {}
```

1.8.5.2.2. 查询单个 ManagedServiceAccount

```
GET /authentication.open-cluster-
management.io/v1beta1/namespaces/{namespace}/managedserviceaccounts/{managedservic
eaccount_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** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	managedserviceaccount_name *required*	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
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

- ***managedserviceaccounts.authentication.open-cluster-management.io***

1.8.5.2.3. 删除 ManagedServiceAccount

> ***DELETE /authentication.open-cluster-management.io/v1beta1/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** *必需*	身份验证：Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN 是用户访问令牌。	字符串
路径	managedserviceaccount_name *required*	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
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

- ***managedserviceaccounts.authentication.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 (v1alpha1)

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

- multiclusterengines.multicluster.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": "multiclusterengines.multicluster.openshift.io"
  },
  "spec": {
    "group": "multicluster.openshift.io",
    "names": {
      "kind": "MultiClusterEngine",
      "listKind": "MultiClusterEngineList",
      "plural": "multiclusterengines",
      "shortNames": [
        "mce"
      ],
      "singular": "multiclusterengine"
    },
    "scope": "Cluster",
    "versions": [
      {
        "additionalPrinterColumns": [
          {
            "description": "The overall state of the MultiClusterEngine",
            "jsonPath": ".status.phase",
            "name": "Status",
            "type": "string"
          },
          {
            "jsonPath": ".metadata.creationTimestamp",
            "name": "Age",
            "type": "date"
          }
        ],
        "name": "v1alpha1",
        "schema": {
          "openAPIV3Schema": {
            "description": "MultiClusterEngine is the Schema for the multiclusterengines\nAPI",
            "properties": {
```

```
          "apiVersion": {
            "description": "APIVersion defines the versioned schema of this representation\nof an object. Servers should convert recognized schemas to the latest\ninternal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources",
            "type": "string"
          },
          "kind": {
            "description": "Kind is a string value representing the REST resource this\nobject represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds",
            "type": "string"
          },
          "metadata": {
            "type": "object"
          },
          "spec": {
            "description": "MultiClusterEngineSpec defines the desired state of MultiClusterEngine",
            "properties": {
              "imagePullSecret": {
                "description": "Override pull secret for accessing MultiClusterEngine\noperand and endpoint images",
                "type": "string"
              },
              "nodeSelector": {
                "additionalProperties": {
                  "type": "string"
                },
                "description": "Set the nodeselectors",
                "type": "object"
              },
              "targetNamespace": {
                "description": "Location where MCE resources will be placed",
                "type": "string"
              },
              "tolerations": {
                "description": "Tolerations causes all components to tolerate any taints.",
                "items": {
                  "description": "The pod this Toleration is attached to tolerates any\ntaint that matches the triple <key,value,effect> using the matching\noperator <operator>.",
                  "properties": {
                    "effect": {
                      "description": "Effect indicates the taint effect to match. Empty\nmeans match all taint effects. When specified, allowed values\nare NoSchedule, PreferNoSchedule and NoExecute.",
                      "type": "string"
                    },
                    "key": {
                      "description": "Key is the taint key that the toleration applies\nto. Empty means match all taint keys. If the key is empty,\noperator must be Exists; this combination means to match all\nvalues and all keys.",
                      "type": "string"
                    },
                    "operator": {
```

```
                  "description": "Operator represents a key's relationship to the\nvalue. Valid
operators are Exists and Equal. Defaults to Equal.\nExists is equivalent to wildcard for value,
so that a pod\ncan tolerate all taints of a particular category.",
                  "type": "string"
                 },
                 "tolerationSeconds": {
                  "description": "TolerationSeconds represents the period of time\nthe toleration
(which must be of effect NoExecute, otherwise\nthis field is ignored) tolerates the taint. By
default, it\nis not set, which means tolerate the taint forever (do not\nevict). Zero and negative
values will be treated as 0 (evict\nimmediately) by the system.",
                  "format": "int64",
                  "type": "integer"
                 },
                 "value": {
                  "description": "Value is the taint value the toleration matches\nto. If the
operator is Exists, the value should be empty,\notherwise just a regular string.",
                  "type": "string"
                 }
                },
                "type": "object"
               },
               "type": "array"
              }
             },
             "type": "object"
            },
            "status": {
             "description": "MultiClusterEngineStatus defines the observed state of
MultiClusterEngine",
             "properties": {
              "components": {
               "items": {
                "description": "ComponentCondition contains condition information for\ntracked
components",
                "properties": {
                 "kind": {
                  "description": "The resource kind this condition represents",
                  "type": "string"
                 },
                 "lastTransitionTime": {
                  "description": "LastTransitionTime is the last time the condition\nchanged
from one status to another.",
                  "format": "date-time",
                  "type": "string"
                 },
                 "message": {
                  "description": "Message is a human-readable message indicating\ndetails
about the last status change.",
                  "type": "string"
                 },
                 "name": {
                  "description": "The component name",
                  "type": "string"
                 },
                 "reason": {
                  "description": "Reason is a (brief) reason for the condition's\nlast status
```

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

```
                "description": "Latest observed overall state",
                "type": "string"
              }
             },
             "type": "object"
            }
           },
           "type": "object"
          }
         },
         "served": true,
         "storage": true,
         "subresources": {
          "status": {}
         }
        }
       ]
      },
      "status": {
       "acceptedNames": {
        "kind": "",
        "plural": ""
       },
       "conditions": [],
       "storedVersions": []
      }
     }
```


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

1.8.6.3.2. specs 列表

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

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

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

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

- Minimal ISO 引导故障故障排除

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 命令以及使用它的前提条件，请参阅 OpenShift Container Platform 文档中的 收集集群数据。

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：

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel9:v2.5 --dest-dir=
<directory>
```

1. 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，该日志从其镜像 registry 中引用镜像。将 sha256 替换为当前镜像：

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-
rhel9@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8>

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

您可以在此处为产品团队创建一个 JIRA 错误。https://issues.redhat.com/projects/ACM/summary/

- 运行 must-gather 命令进行故障排除

1.9.2.4. 托管集群的 must-gather

如果托管 control plane 集群出现问题，您可以运行 must-gather 命令来收集信息，以帮助您进行故障排除。

1.9.2.4.1. 关于托管集群的 must-gather 命令

该命令为管理集群和托管集群生成输出。

- 多集群引擎 operator hub 集群中的数据：
  - 集群范围的资源：这些资源是管理集群的节点定义。
  - hypershift-dump 压缩文件：如果您需要与其他人员共享内容，该文件非常有用。
  - 命名空间资源 ：这些资源包括来自相关命名空间中的所有对象，如配置映射、服务、事件和日志。
  - 网络日志： 这些日志包括 OVN 北向和南向数据库和每个数据库的状态。
  - 托管的集群：此级别的输出涉及托管集群内的所有资源。
- 来自托管集群的数据：
  - 集群范围的资源：这些资源包含所有集群范围的对象，如节点和 CRD。
  - 命名空间资源 ：这些资源包括来自相关命名空间中的所有对象，如配置映射、服务、事件和日志。

虽然输出不包含集群中的任何 secret 对象，但它可以包含对 secret 的名称的引用。

1.9.2.4.2. 先决条件

要通过运行 must-gather 命令来收集信息，您必须满足以下条件：

- 您必须确保 kubeconfig 文件已被加载，并指向 multicluster engine operator hub 集群。
- 您必须具有对 multicluster engine operator hub 集群的 cluster-admin 访问权限。
- 您必须具有 HostedCluster 资源的 name 值，以及部署自定义资源的命名空间。

1.9.2.4.3. 为托管集群输入 must-gather 命令

1. 输入以下命令收集有关托管集群的信息。在命令中，hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE 参数是可选的；如果没有包括它，则命令会像在 default 命名空间（即 集群 ）中运行。

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.x
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME
```

2. 要将命令的结果保存到压缩文件中，请包含 --dest-dir= NAME 参数，将 NAME 替换为您要保存结果的目录的名称：

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.x
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME --dest-dir=NAME ; tar -cvzf NAME.tgz NAME
```

1.9.2.4.4. 在断开连接的环境中输入 must-gather 命令

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 /usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-name=HOSTEDCLUSTERNAME --dest-dir=./data
```

1.9.2.4.5. 其他资源

- 有关托管 control plane 故障排除的更多信息，请参阅 OpenShift Container Platform 文档中的对托管的 control plane 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. 在代理平台上删除托管的 control plane 集群失败的问题

当您在 Agent 平台上销毁托管的 control plane 集群时，所有后端资源通常会被删除。如果没有正确删除机器资源，集群删除会失败。在这种情况下，您必须手动删除剩余的机器资源。

1.9.4.1. 症状：销毁托管 control plane 集群时发生错误

试图销毁 Agent 平台上托管的 control plane 集群后，hcp destroy 命令会失败并显示以下错误：

+

```
2024-02-22T09:56:19-05:00    ERROR    HostedCluster deletion failed    {"namespace": "clusters", "name": "hosted-0", "error": "context deadline exceeded"}
2024-02-22T09:56:19-05:00    ERROR    Failed to destroy cluster    {"error": "context deadline exceeded"}
```

1.9.4.2. 解决问题： 手动删除剩余的机器资源

完成以下步骤，在 Agent 平台上成功销毁托管的 control plane 集群：

1. 运行以下命令，将 < hosted_cluster_namespace > 替换为托管集群命名空间的名称来查看剩余的机器资源列表：

```
oc get machine -n <hosted_cluster_namespace>
```

请参见以下示例输出：

```
NAMESPACE          NAME          CLUSTER          NODENAME    PROVIDERID    PHASE      AGE    VERSION
clusters-hosted-0    hosted-0-9gg8b    hosted-0-nhdbp                    Deleting    10h    4.15.0-rc.8
```

2. 运行以下命令以删除附加到机器资源的 machine.cluster.x-k8s.io finalizer：

```
oc edit machines -n <hosted_cluster_namespace>
```

3. 运行以下命令验证您是否在终端中收到 No resources found 信息：

```
oc get agentmachine -n <hosted_cluster_namespace>
```

4. 运行以下命令，在 Agent 平台上销毁托管的 control plane 集群：

```
hcp destroy cluster agent --name <cluster_name>
```

将 <cluster_name> 替换为集群的名称。

1.9.5. 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。

1.9.5.1. 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。

1.9.5.2. 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。

1.9.6. 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。

1.9.6.1. 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。

1.9.6.2. 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。

1.9.7. 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。

1.9.7.1. 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）。

1.9.7.2. 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。

1.9.8. 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。

1.9.8.1. 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。

1.9.8.2. 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。

1.9.9. 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。

1.9.9.1. 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。

1.9.9.2. 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

1.9.9.3. 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}"
```

1.9.10. 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。

1.9.10.1. 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。

1.9.10.2. 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**。

1.9.10.3. 解决问题： 集群处于待处理导入状态

1.

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。

1.9.11. 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。

1.9.11.1. 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。

1.9.11.2. 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：

请参阅以下 work-agent 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
```

请参阅以下 registration-agent 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
```

1.9.11.3. 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由 multicluster engine operator 创建的 local-cluster 或受管集群，则必须等待 10 分钟或更长时间恢复受管集群。

要立即恢复受管集群，您可以删除 hub 集群上的受管集群导入 secret，并使用 multicluster engine operator 恢复它。运行以下命令：

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

将 < cluster_name > 替换为您要恢复的受管集群的名称。

如果要恢复使用 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。

1.9.12. 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。

1.9.12.1. 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。

1.9.12.2. 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。

1.9.13. 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。

1.9.13.1. 受管集群创建失败并显示证书 IP SAN 错误

1.9.13.1.1. 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。

1.9.13.1.2. 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
```

1.9.13.1.3. 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。

1.9.13.2. 受管集群创建失败并显示未知证书颁发机构

1.9.13.2.1. 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。

1.9.13.2.2. 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"

1.9.13.2.3. 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。

1.9.13.3. 受管集群创建带有过期证书失败

1.9.13.3.1. 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。

1.9.13.3.2. 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

1.9.13.3.3. 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。

1.9.13.4. 受管集群创建失败且没有标记权限

1.9.13.4.1. 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。

1.9.13.4.2. 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\"
{"
```

1.9.13.4.3. 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。

1.9.13.5. 受管集群创建失败并显示无效的 dnsVIP

1.9.13.5.1. 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。

1.9.13.5.2. 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
```

1.9.13.5.3. 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。

1.9.13.6. 受管集群创建带有不正确的网络类型失败

1.9.13.6.1. 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。

1.9.13.6.2. 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"
```

1.9.13.6.3. 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。

1.9.13.7. 受管集群创建失败并显示磁盘更改错误

1.9.13.7.1. 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。

1.9.13.7.2. 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)
```

1.9.13.7.3. 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。

1.9.14. 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。

1.9.14.1. 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。

1.9.14.2. 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。*

1.9.14.3. 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。

有关创建集群的更多信息，请参阅 集群创建简介。

1.9.15. OpenShift Container Platform 版本 3.11 集群导入失败的故障排除

1.9.15.1. 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|...
```

1.9.15.2. 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。

1.9.15.3. 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 文档中的安装和设置 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
```

1.9.16. 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。

1.9.16.1. 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。

1.9.16.2. 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。

1.9.16.3. 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。

1.9.17. 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。

1.9.17.1. 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。

1.9.17.2. 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**。*

1.9.18. 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。

1.9.18.1. 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。

1.9.18.2. 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. 使用集群导入简介 中的流程再次导入集群。

1.9.19. 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 PlacementDescision，请按照以下步骤排除此问题。

1.9.19.1. 症状：创建放置后缺少 PlacementDecision

创建 放置 后，不会自动生成一个 PlacementDescision。

1.9.19.2. 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。

1.9.20. 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。

1.9.20.1. 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"}
]
```

1.9.20.2. 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。

1.9.21. Minimal ISO 引导故障故障排除

当您尝试引导最小 ISO 时可能会遇到问题。

1.9.21.1. 症状：最小 ISO 引导失败

引导屏幕显示主机无法下载根文件系统镜像。

1.9.21.2. 解决问题： Minimal ISO 引导失败

请参阅 OpenShift Container Platform 文档中的辅助安装程序 故障排除最小 ISO 引导失败，以了解如何排除此问题。

1.9.22. Red Hat OpenShift Virtualization 上托管集群的故障排除

当您对 Red Hat OpenShift Virtualization 上的托管集群进行故障排除时，请从顶层 HostedCluster 和 NodePool 资源开始，然后处理堆栈，直到您找到根本原因。以下步骤可帮助您发现常见问题的根本原因。

1.9.22.1. 症状： HostedCluster 资源处于部分状态

托管的 control plane 不会完全在线，因为 HostedCluster 资源待处理。

1.9.22.1.1. 鉴别问题： 检查先决条件、资源状况和节点和 operator 状态

- 确保您满足 Red Hat OpenShift Virtualization 上托管集群的所有先决条件
- 查看 HostedCluster 和 NodePool 资源中的条件，以了解防止进度的验证错误。

- 通过使用托管的集群的 kubeconfig 文件，检查托管集群的状态：
  - 查看 oc get clusteroperators 命令的输出以查看哪些集群操作器待处理。
  - 查看 oc get nodes 命令的输出，以确保 worker 节点就绪。

1.9.22.2. 症状：没有 worker 节点被注册

托管的 control plane 不会完全在线，因为托管的 control plane 没有注册 worker 节点。

1.9.22.2.1. 鉴别问题： 检查托管 control plane 的各种部分的状态

- 查看 HostedCluster 和 NodePool 条件，以了解问题可能是什么。
- 输入以下命令查看 NodePool 资源的 KubeVirt worker 节点虚拟机(VM)状态：

```
oc get vm -n <namespace>
```

- 如果虚拟机处于 provisioning 状态，请输入以下命令来查看 VM 命名空间中的 CDI 导入 pod，以了解 importer pod 尚未完成的原因：

```
oc get pods -n <namespace> | grep "import"
```

- 如果虚拟机处于启动状态，请输入以下命令查看 virt-launcher pod 的状态：

```
oc get pods -n <namespace> -l kubevirt.io=virt-launcher
```

  如果 virt-launcher pod 处于待处理状态，请调查 pod 没有调度的原因。例如，运行 virt-launcher pod 可能没有足够的资源。

- 如果虚拟机正在运行，但它们没有注册为 worker 节点，请使用 Web 控制台获得对受影响虚拟机的 VNC 访问。VNC 输出指示是否应用了 ignition 配置。如果虚拟机在启动时无法访问托管的 control plane ignition 服务器，则虚拟机无法正确置备。
-

如果应用了 ignition 配置，但虚拟机仍然没有注册为节点，请参阅 识别问题：访问 VM 控制台日志 以了解如何在启动时访问虚拟机控制台日志。

1.9.22.3. 症状： Worker 节点处于 NotReady 状态

在集群创建过程中，在部署网络堆栈时，节点会临时进入 NotReady 状态。该过程的这一部分是正常的。但是，如果这部分进程需要超过 15 分钟，则可能会出现问题。

1.9.22.3.1. 鉴别问题：检查节点对象和 pod

- 输入以下命令查看节点对象中的条件，并确定节点未就绪的原因：

```
oc get nodes -o yaml
```

- 输入以下命令查找集群中的 pod 失败：

```
oc get pods -A --field-selector=status.phase!=Running,status,phase!=Succeeded
```

1.9.22.4. 症状：入口和控制台集群操作器不会上线

托管的 control plane 不会完全在线，因为 Ingress 和控制台集群 Operator 没有在线。

1.9.22.4.1. 鉴别问题： 检查通配符 DNS 路由和负载均衡器

- 如果集群使用默认 Ingress 行为，请输入以下命令来确保在托管虚拟机(VM)的 OpenShift Container Platform 集群中启用了通配符 DNS 路由：

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

- 如果您将自定义基域用于托管的 control plane，请完成以下步骤：
  - 确保负载均衡器正确以 VM pod 为目标。
  - 确保通配符 DNS 条目以负载均衡器 IP 为目标。

1.9.22.5. 症状：托管集群的负载均衡器服务不可用

托管的 control plane 不会完全在线，因为负载均衡器服务不可用。

1.9.22.5.1. 鉴别问题： 检查事件、详情和 kccm pod

- 查找与托管集群中的负载均衡器服务关联的事件和详情。
- 默认情况下，托管集群的负载均衡器由托管的 control plane 命名空间中的 kubevirt-cloud-controller-manager 处理。确保 kccm pod 在线，并查看其日志中是否有错误或警告。要识别托管的 control plane 命名空间中的 kccm pod，请输入以下命令：

```
oc get pods -n <hosted-control-plane-namespace> -l app=cloud-controller-manager
```

1.9.22.6. 症状： 托管集群 PVC 不可用

托管的 control plane 不会完全在线，因为托管集群的持久性卷声明(PVC)不可用。

1.9.22.6.1. 鉴别问题： 检查 PVC 事件和详情，以及组件日志

- 查找与 PVC 关联的事件和详情，以了解发生哪些错误。
- 如果 PVC 无法附加到 pod，请查看托管集群中 kubevirt-csi-node daemonset 组件的日志，以进一步调查问题。要为每个节点识别 kubevirt-csi-node pod，请输入以下命令：

```
oc get pods -n openshift-cluster-csi-drivers -o wide -l app=kubevirt-csi-driver
```

- 如果 PVC 无法绑定到持久性卷(PV)，查看托管 control plane 命名空间中的 kubevirt-csi-controller 组件的日志。要识别托管 control plane 命名空间中的 kubevirt-csi-controller pod，请输入以下命令：

```
oc get pods -n <hcp namespace> -l app=kubevirt-csi-driver
```

1.9.22.7. 症状：虚拟机节点没有正确加入集群

托管的 control plane 不会完全在线，因为虚拟机节点没有正确加入集群。

1.9.22.7.1. 鉴别问题： 访问虚拟机控制台日志

要访问 VM 控制台日志，请完成 如何获取 OpenShift Virtualization Hosted Control Plane 集群部分虚拟机的串口控制台日志的步骤。