Red Hat Advanced Cluster Management for Kubernetes 2.5

附加组件

了解更多信息，了解如何为集群使用附加组件。

Last Updated: 2023-07-15

Red Hat Advanced Cluster Management for Kubernetes 2.5 附加组件

了解更多信息，了解如何为集群使用附加组件。

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

法律通告

摘要

了解更多信息，了解如何为集群使用附加组件。

目录

第 1 章 附加组件概述

Red Hat Advanced Cluster Management for Kubernetes 附加组件可以改进一些性能，并添加功能来简化您的应用程序。以下小节概述了 Red Hat Advanced Cluster Management 可用的附加组件：

- Submariner multicluster networking 和 service discovery
- VolSync 持久性卷复制服务
- 在来自 Kubernetes operator 的 multicluster engine 的集群上启用 klusterlet 附加组件

1.1. SUBMARINER MULTICLUSTER NETWORKING 和 SERVICE DISCOVERY

Submariner 服务是一个开源工具，可与 Red Hat Advanced Cluster Management for Kubernetes 一起使用，用于为您的环境中两个或多个 Kubernetes 集群（内部环境或云）提供 Pod 之间的直接联网。有关 Submariner 的更多信息，请参阅 Submariner 站点。

您可以在以下环境中托管的 OpenShift Container Platform 集群上启用 Submariner：

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- IBM Cloud
- Microsoft Azure
- Red Hat OpenShift Dedicated
- VMware vSphere
- 裸机
- Red Hat OpenStack Platform

Red Hat Advanced Cluster Management for Kubernetes 提供了一个 Submariner 组件，您可以使用 hub 集群在环境中部署。

- 先决条件
- 为 Submariner 准备所选主机
  - 为 Submariner 准备 Microsoft Azure
  - 为 Submariner 准备 IBM Cloud
  - 为 Submariner 准备 VMware vSphere
  - 为 Submariner 准备裸机
  - 为 Submariner 准备 Red Hat OpenStack Platform

1.1.1. 先决条件

在使用 Submariner 前，请确保已满足以下先决条件：

- 使用 **cluster-admin** 权限访问 hub 集群的凭证。
- 在网关节点之间配置了 IP 连接。连接两个集群时，网关节点必须可使用分配给网关节点的公共或私有 IP 地址访问至少一个集群。如需更多信息，请参阅 Submariner NAT Traversal。
- 各个受管集群中所有节点的防火墙配置必须同时允许 4800/UDP。
- 网关节点上的防火墙配置允许入口 8080/TCP，以便集群中的其他节点可以访问它。
- 为 4500/UDP 以及网关节点上 IPsec 流量的其他端口打开防火墙配置。
  注：当您的集群部署在 AWS 或 GCP 环境中时，这会自动配置，但必须为其他环境中的集群以及保护私有云的防火墙手动配置。
- **managedcluster** 名称必须遵循 RFC 1123 中定义的 DNS 标签标准。这意味着名称必须满足以下条件：
  - 最多 63 个字符
  - 仅包含小写字母字符或 '-'
  - 以字母数字字符开头
  - 以字母数字字符结尾

表 1.1. Submariner 需要的端口

名称	默认值	Customizable
IPsec NATT	4500/UDP	是
VXLAN	4800/UDP	否
Submariner 指标端口	8080/TCP	否

有关 先决条件的详情，请参阅 Submariner 上游先决条件文档。

1.1.2. subctl 命令工具

Submariner 包含 **subctl** 工具，它提供额外的命令来简化 Submariner 环境中运行的任务。

1.1.2.1. 安装 subctl 命令工具

subctl 实用程序包含在容器镜像中。完成以下步骤以在本地安装 **subctl** 工具：

1. 下载 **subctl** 容器，使用以下命令将 **subctl** 二进制文件的压缩版本提取到 **/tmp** 中：

```
oc image extract registry.redhat.io/rhacm2/subctl-rhel8:v0.12 --path=/dist/subctl-v0.12-linux-amd64.tar.xz:/tmp/ --confirm
```

2. 输入以下命令解压 **subctl** 工具：

```
tar -C /tmp/ -xf /tmp/subctl-v0.12-linux-amd64.tar.xz
```

3. 输入以下命令安装 **subctl** 工具：

```
install -m744 /tmp/subctl-v0.12/subctl-v0.12-linux-amd64 /$HOME/.local/bin/subctl
```

1.1.2.2. 使用 subctl 命令

在您的路径中添加实用程序后，请查看下表以了解可用命令的简短描述：

导出服务	为指定的服务创建一个 **ServiceExport** 资源，它会在 Submariner 部署中启用其他集群来发现对应的服务。
取消导出服务	为指定的服务删除 **ServiceExport** 资源，这会阻止 Submariner 部署中的其他集群发现对应的服务。
显示	提供有关 Submariner 资源的信息。
验证	当 Submariner 在一组集群中配置 Submariner 时，验证连接、服务发现和其他 Submariner 功能。
benchmark	使用 Submariner 或单一集群内启用的集群对的吞吐量和延迟基准。
诊断	运行检查来识别阻止 Submariner 部署正常工作的问题。
gather	从集群收集信息，以帮助对 Submariner 部署进行故障排除。
version	显示 **subctl** 二进制文件的版本详情。

有关 **subctl** 实用程序及其命令的更多信息，请参阅 Submariner 文档中的 SUBCTL。

1.1.3. Globalnet

Globalnet 是 Submariner 附加组件中包含的功能，支持使用重叠 CIDR 的集群间的连接。Globalnet 是一个集群范围的配置，当第一个受管集群添加到集群集时可选择。启用 Globalnet 时，从虚拟全局专用网络中为每个受管集群分配一个全局 CIDR。全局 CIDR 用于支持集群间通信。

如果运行 Submariner 的集群可能会有重叠的 CIDR，请考虑启用 Globalnet。当使用 Red Hat Advanced Cluster Management 控制台时，**ClusterAdmin** 可以通过选择选项 **Enable Globalnet** 为集群集启用 Globalnet，方法是在为集群集中的集群启用 Submariner add-on 时。启用 Globalnet 后，您无法在不删除 Submariner 的情况下禁用它。

在使用 Red Hat Advanced Cluster Management API 时，**ClusterAdmin** 可以通过在 <**ManagedClusterSet>-broker 命名空间中创建 submariner-broker** 对象来启用 Globalnet。

ClusterAdmin 角色在 broker 命名空间中创建此对象所需的权限。**ManagedClusterSetAdmin** 角色有时被创建来充当集群集的代理管理员，没有所需的权限。要提供所需的权限，**ClusterAdmin** 必须将 **access-to-brokers-submariner-crd** 的角色权限关联到 **ManagedClusterSetAdmin** 用户。

完成以下步骤以创建 **submariner-broker** 对象：

1. 创建 **submariner-broker** 对象，该对象通过创建名为 **submariner-broker.yaml** 的 YAML 文件来指定 Globalnet 配置，其中包含类似以下示例的内容：

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
  name: submariner-broker
  namespace: <broker-namespace>
spec:
  globalnetEnabled: <true-or-false>
```

   将 **broker-namespace** 替换为代理命名空间的名称。

   将 **true-or-false** 替换为 **true** 来启用 Globalnet。

2. 输入以下命令将文件应用到 YAML 文件：

```
oc apply -f submariner-broker.yaml
```

有关 Globalnet 的更多信息，请参阅 Submariner 文档中的 Globalnet 控制器。

1.1.4. 部署 Submariner

您可以将 Submariner 部署到以下供应商上的网络集群中：

自动部署过程：

- Amazon Web Services
- Google Cloud Platform

手动部署过程：

- Microsoft Azure
- IBM Cloud
- VMware vSphere
- 裸机
- Red Hat OpenStack Platform

1.1.4.1. 使用控制台部署 Submariner

您可以使用 Red Hat Advanced Cluster Management for Kubernetes 控制台，在 Amazon Web Services、Google Cloud Platform 和 VMware vSphere 上部署的 Red Hat OpenShift Container Platform 受管集群上部署 Submariner。要在其他供应商上部署 Submariner，请按照 部署 Submariner 中的说明操作。完成以下步骤，使用 Red Hat Advanced Cluster Management for Kubernetes 控制台部署 Submariner：

需要的访问权限：集群管理员

1. 在控制台导航菜单中选择 **Infrastructure** > **Clusters**。

2. 在 *Clusters* 页面上，选择 *Cluster sets* 选项卡。要使用 Submariner 启用的集群必须位于同一集群集合中。
3. 如果要在其上部署 Submariner 的集群已位于同一集群集中，请跳至第 5 步来部署 Submariner。
4. 如果要在其上部署 Submariner 的集群不在同一个集群集中，请完成以下步骤为它们创建一个集群集：
   a. 选择 **Create cluster set**。
   b. 对集群集进行命名，然后选择 **Create**。
   c. 选择 **Manage resource assignments** 以将集群分配到集群集。
   d. 选择您要与 Submariner 连接的受管集群，将它们添加到集群集合中。
   e. 选择 **Review** 来查看并确认您选择的集群。
   f. 选择 **Save** 保存集群集，并查看生成的集群设置页面。
5. 在集群集页面中，选择 *Submariner add-ons* 选项卡。
6. 选择 **Install Submariner add-ons**。
7. 选择您要在其上部署 Submariner 的集群。
8. 在 *Install Submariner add-ons* 编辑器中输入以下信息 ：
   - **AWS Access Key ID** - 此字段仅在导入 AWS 集群时可见。
   - **AWS Secret Access Key** - 此字段仅在导入 AWS 集群时可见。
   - **Google Cloud Platform service account JSON key** - 此字段仅在导入 Google Cloud Platform 集群时可见。
   - **Instance type** - 在受管集群中创建的网关节点的 Amazon Web Services EC2 实例类型。默认值为 **c5d.large**。只有在受管集群环境是 AWS 时，此字段才可见。
   - **IPsec NAT-T port** - IPsec NAT 遍历端口的默认值是 **4500**。如果您的受管集群环境是 VMware vSphere，请确保在防火墙中打开此端口。
   - **网关数** - 用于在受管集群中部署 Submariner 网关组件的 worker 节点数量。默认值为 **1**。如果值大于 1，则会自动启用 Submariner 网关高可用性(HA)。
   - **Cable driver** - 维护跨集群隧道的 Submariner 网关电缆引擎组件。默认值为 **Libreswan IPsec**。
9. 在编辑器末尾选择 **Next** 以移动到下一个集群的编辑器，并为您选择的每个剩余的集群完成这个步骤。
10. 验证每个受管集群的配置。
11. 点 **Install** 在所选受管集群上部署 Submariner。
    安装和配置完成可能需要几分钟时间。您可以在 *Submariner add-ons* 选项卡中的列表中检查 Submariner 状态：
    - **连接状态**指示在受管集群中建立多少个 Submariner 连接。

- **代理状**态代表 Submariner 是否成功部署到受管集群中。控制台可能会报告 **Degraded** 状态，直到安装和配置为止。
- **Gateway nodes labeled** 表示有多少个 worker 节点使用 Submariner 网关标签：**submariner.io/gateway=true**。

Submariner 现在在集群中部署。

1.1.4.2. 手动部署 Submariner

在使用 Red Hat Advanced Cluster Management for Kubernetes 部署 Submariner 前，您必须在托管环境中为连接准备集群。目前，您可以使用 **SubmarinerConfig** API 在 Amazon Web Services、Google Cloud Platform 和 VMware vSphere 上自动准备集群。对于其他平台，您需要手动准备它们，请参阅准备所选主机以部署 Submariner 的步骤。

1.1.4.2.1. 准备所选主机来部署 Submariner

在使用 Red Hat Advanced Cluster Management for Kubernetes 部署 Submariner 前，您必须在托管环境中为连接手动准备集群。不同主机环境的要求有所不同，因此请按照您的托管环境说明进行操作。

1.1.4.2.1.1. 为 Submariner 准备 Microsoft Azure

要准备 Microsoft Azure 上的集群来部署 Submariner 组件，请完成以下步骤：

1. 在 Microsoft Azure 环境中创建入站和出站防火墙规则，以打开 IP 安全 NAT 遍历端口（默认为 4500/UDP）以启用 Submariner 通信：

```
# Create inbound NAT rule
$ az network lb inbound-nat-rule create --lb-name <lb-name> \
--resource-group <res-group> \
--name <name> \
--protocol Udp --frontend-port <ipsec-port> \
--backend-port <ipsec-port> \
--frontend-ip-name <frontend-ip-name>

# Add VM network interface to the just-created inbound NAT rule
$ az network nic ip-config inbound-nat-rule add \
--lb-name <lb-name> --resource-group <res-group> \
--inbound-nat-rule <nat-name> \
--nic-name <nic-name> --ip-config-name <pipConfig>
```

   将 **lb-name** 替换为您的负载均衡器名称。

   将 **res-group** 替换为您的资源组名称。

   使用负载平衡入站 NAT 规则名称替换替换 **nat-name**。

   使用您的 IPsec 端口替换 **ipsec-port**。

   将 **pipConfig** 替换为集群前端 IP 配置名称。

   将 **nic-name** 替换为您的网卡（NIC）。

2. 创建一个负载均衡入站 NAT 规则以转发 Submariner 网关指标服务请求：

```
# Create inbound NAT rule
```

```
$ az network lb inbound-nat-rule create --lb-name <lb-name> \
--resource-group <res-group> \
--name <name> \
--protocol Tcp --frontend-port 8080 --backend-port 8080 \
--frontend-ip-name <frontend-ip-name>

# Add VM network interface to the just-created inbound NAT rule
$ az network nic ip-config inbound-nat-rule add \
--lb-name <lb-name> --resource-group <res-group> \
--inbound-nat-rule <nat-name> \
--nic-name <nic-name> --ip-config-name <pipConfig>
```

将 **lb-name** 替换为您的负载均衡器名称。

将 **res-group** 替换为您的资源组名称。

使用负载平衡入站 NAT 规则名称替换替换 **nat-name**。

将 **pipConfig** 替换为集群前端 IP 配置名称。

将 **nic-name** 替换为您的网卡（NIC）。

3. 在 Azure 上创建网络安全组（NSG）安全规则，为 Submariner 打开 NAT 遍历端口（默认为 4500/UDP）：

```
$ az network nsg rule create --resource-group <res-group> \
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Inbound --access Allow \
--protocol Udp --destination-port-ranges <ipsec-port>

$ az network nsg rule create --resource-group <res-group> \
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Outbound --access Allow \
--protocol Udp --destination-port-ranges <ipsec-port>
```

将 **res-group** 替换为您的资源组名称。

使用您的 NSG 名称替换 **nsg-name**。

将 **priority** 替换为您的规则优先级。

使用您的规则名称替换 **name**。

使用您的 IPsec 端口替换 **ipsec-port**。

4. 创建 NSG 规则以打开 4800/UDP 端口，将来自 worker 和 master 节点的 pod 流量封装到 Submariner 网关节点：

```
$ az network nsg rule create --resource-group <res-group> \
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Inbound --access Allow \
--protocol Udp --destination-port-ranges 4800 \

$ az network nsg rule create --resource-group <res-group> \
```

```
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Outbound --access Allow \
--protocol Udp --destination-port-ranges 4800
```

将 **res-group** 替换为您的资源组名称。

使用您的 NSG 名称替换 **nsg-name**。

将 **priority** 替换为您的规则优先级。

使用您的规则名称替换 **name**。

5. 创建 NSG 规则以打开 8080/TCP 端口，从 Submariner Gateway 节点导出指标服务：

```
$ az network nsg rule create --resource-group <res-group> \
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Inbound --access Allow \
--protocol Tcp --destination-port-ranges 8080 \

$ az network nsg rule create --resource-group <res-group> \
--nsg-name <nsg-name> --priority <priority> \
--name <name> --direction Outbound --access Allow \
--protocol Udp --destination-port-ranges 8080
```

将 **res-group** 替换为您的资源组名称。

使用您的 NSG 名称替换 **nsg-name**。

将 **priority** 替换为您的规则优先级。

使用您的规则名称替换 **name**。

6. 使用以下标签 **submariner.io/gateway=true** 标记集群中的 worker 节点。

1.1.4.2.1.2. 为 Submariner 准备 IBM Cloud

IBM Cloud 上有两种 Red Hat OpenShift Kubernetes Service（ROKS）：经典（classic）集群，以及在虚拟私有云（VPC）中的第二代计算基础架构。因为无法为经典集群配置 IPsec 端口，所以 Submariner无法在典型的 ROKS 集群上运行。

要将 VPC 上的 ROKS 集群配置为使用 Submariner，请完成以下链接中的步骤：

1. 在创建集群前，为 pod 和服务指定子网，这样可避免与其他集群重叠的 CIDR。如果使用现有的集群，请确保集群之间没有重叠的 pod 和服务 CIDR。详情请参阅 VPC 子网。
2. 将公共网关附加到集群中使用的子网。详情请参阅公共网关。
3. 通过完成安全组中的步骤，为集群的默认安全组创建入站规则。确保防火墙允许 4500/UDP 和 500/UDP 端口上的入站和出站流量，并为所有其他节点允许入站和出站 UDP/4800。
4. 将集群中带有公共网关的节点标记为 **submariner.io/gateway=true**。
5. 参阅 Calico 来配置 Calico CNI，在集群中创建 IPPools。

1.1.4.2.1.3. 为 Submariner 准备 VMware vSphere

Submariner 使用 IPsec 在网关节点上的集群之间建立安全隧道。您可以使用默认端口或指定自定义端口。当您运行这个步骤时，没有指定 IPsec NATT 端口，默认端口会自动用于通信。默认端口为 4500/UDP。

Submariner 使用虚拟可扩展 LAN (VXLAN) 在流量从 worker 和 master 节点移动到网关节点时封装流量。VXLAN 端口无法自定义，并且始终是端口 4800/UDP。

Submariner 使用 8080/TCP 在集群中的节点之间发送其指标信息，此端口无法自定义。

在启用 Submariner 前，VMWare vSphere 管理员必须打开以下端口：

表 1.2. VMware vSphere 和 Submariner 端口

名称	默认值	Customizable
IPsec NATT	4500/UDP	是
VXLAN	4800/UDP	否
Submariner 指标	8080/TCP	否

要准备 VMware vSphere 集群以部署 Submariner，请完成以下步骤：

1. 确保 IPsec NATT、VXLAN 和指标端口已打开。
2. 自定义并应用类似以下示例的 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

将 **managed-cluster-namespace** 替换为受管集群的命名空间。

注： **SubmarinerConfig** 的名称必须是 **submariner**，如示例中所示。

此配置对 Submariner 使用默认的网络地址转换 - 遍历 (NATT) 端口 (4500/UDP)，并在 vSphere 集群中使用一个 worker 节点作为 Submariner 网关。

Submariner 使用 IP 安全 (IPsec) 在网关节点上的集群之间建立安全隧道。您可以使用默认 IPsec NATT 端口，或者指定您配置的不同端口。当您运行这个步骤时，没有指定 IPsec NATT 端口 4500/UDP，将自动用于通信。

1.1.4.2.1.4. 为 Submariner 准备裸机

要准备裸机集群来部署 Submariner，请完成以下步骤：

1. 确保 IPsec NATT、VXLAN 和指标端口已打开。
2. 自定义并应用类似以下示例的 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
```

```
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

将 **managed-cluster-namespace** 替换为受管集群的命名空间。

注： SubmarinerConfig 的名称必须是 **submariner**，如示例中所示。

此配置对 Submariner 使用默认网络地址转换 - 遍历(NATT)端口(4500/UDP)，并且一个 worker 节点被标记为裸机集群上的 Submariner 网关。

Submariner 使用 IP 安全 (IPsec) 在网关节点上的集群之间建立安全隧道。您可以使用默认 IPsec NATT 端口，或者指定您配置的不同端口。当您运行这个步骤时，没有指定 IPsec NATT 端口 4500/UDP，将自动用于通信。

有关自定义选项的信息，请参阅自定义 Submariner 部署。

1.1.4.2.1.5. 为 Submariner 准备 Red Hat OpenStack Platform

您可以使用 **SubmarinerConfig** API 将 Red Hat OpenStack Platform 集群配置为与 Submariner 部署集成。要准备 Red Hat OpenStack Platform 集群来部署 Submariner，请完成以下步骤：

1. 使用 Red Hat OpenStack Platform 环境的身份验证信息创建一个名为 **clouds.yaml** 的基本 64 编码文件。该文件应类似以下示例：

```
clouds:
openstack:
auth:
  auth_url: https://rhos-d.infra.prod.upshift.rdu2.redhat.com:13000
  application_credential_id:
  application_credential_secret:
region_name: "regionOne"
interface: "public"
identity_api_version: 3
auth_type: "v3applicationcredential"
```

2. 在包含 Red Hat OpenStack Platform 凭证 secret 的受管集群的命名空间中创建一个 secret。

   a. 创建名为 **openstack_secret.yaml** 的文件并添加以下示例内容：

```
apiVersion: v1
kind: Secret
metadata:
  name: <managed-cluster-name>-rhos-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  clouds.yaml: <clouds.yaml>
  cloud : <rhos-cloud-name>
```

将 **managed-cluster-name** 替换为受管集群的名称。**managed-cluster-name-rhos-creds** 的值是 Red Hat OpenStack Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该 secret。

将 **managed-cluster-namespace** 替换为受管集群的命名空间。

将 **clouds.yaml** 替换为编码的 Red Hat OpenStack Platform **clouds.yaml 文件的路径：$(base64 -w0)** 文件。

将 **rhos-cloud-name** 替换为您编码的 Red Hat OpenStack Platform 云名称 < **cloud-name>: $base64 -w0**）

b. 运行以下命令以应用该文件：

```
oc apply -f openstack_secret.yaml
```

3. 如果您使用 Red Hat Advanced Cluster Management 创建受管集群，或者在上一步中创建 secret，请准备集群。

a. 创建名为 **submar_addon.yaml** 的文件，其内容类似以下示例：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   credentialsSecret:
     name: <managed-cluster-name>-rhos-creds
```

将 **managed-cluster-namespace** 替换为受管集群的命名空间。

将 **managed-cluster-name** 替换为受管集群的名称。**managed-cluster-name-rhos-creds** 的值是 Red Hat OpenStack Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该 secret。

注：SubmarinerConfig 自定义资源的名称必须是 **submariner**，如示例所示。

b. 运行以下命令以应用该文件：

```
oc apply -f submar_addon.yaml
```

此配置会自动打开 Submariner 所需的端口：网络地址转换 - 遍历(NATT)端口(4500/UDP)、虚拟可扩展 LAN(VXLAN)端口(4800/UCP)，以及 RHOS 实例上的 Submariner 指标端口(8080/TCP)。它还将一个 worker 节点标记为 Submariner 网关，并在 Red Hat OpenStack Platform 集群中启用此节点的公共 IP 地址。

如果要自定义网络地址转换-Traversal(NATT)端口、网关节点数量或部署的网关节点的实例类型，请参阅为必要的步骤 自定义 Submariner 部署。

1.1.4.2.2. 使用 ManagedClusterAddOn API 部署 Submariner

要使用 **ManagedClusterAddOn** API 部署 Submariner，请完成以下步骤：

1. 使用 管理集群 文档的 Creating *and ManagedClusterSets 主题中所述，在 hub 集群上创建一个* ***Manage*** *dClusterSet* 资源。**ManagedClusterSet** 的条目应类似以下内容：

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

```
metadata:
  name: <managed-cluster-set-name>
```

将 **managed-cluster-set-name** 替换为您要创建的 **ManagedClusterSet** 的名称。

注：Kubernetes 命名空间名称的最大长度为 63 个字符，因此 **<managed-cluster-set-name>** 的最大长度为 56 个字符。如果 **<managed-cluster-set-name>** 的长度超过 56，则 **<managed-cluster-set-name>** 将从头截断。

创建 **ManagedClusterSet** 后，**submariner-addon** 会创建一个名为 **<managed-cluster-set-name>-broker** 的命名空间，并将 Submariner 代理部署到其中。

2. 通过自定义并应用类似以下示例的 YAML 内容，在 hub 集群中创建 **Broker** 配置。

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
    name: submariner-broker
    namespace: <managed-cluster-set-name>-broker
spec:
    globalnetEnabled: false
```

将 **managed-cluster-set-name** 替换为受管集群的名称。

如果要在 **ManagedClusterSet** 中启用 Submariner Globalnet，将 **globalnetEnabled** 的值设置为 **true**。

3. 输入以下命令在 **ManagedClusterSet** 中添加一个受管集群：

```
oc label managedclusters <managed-cluster-name> "cluster.open-cluster-
management.io/clusterset=<managed-cluster-set-name>" --overwrite
```

将 **<managed-cluster-name>** 替换为您要添加到 **ManagedClusterSet** 的受管集群的名称。

将 **<managed-cluster-set-name>** 替换为您要添加受管集群的 **ManagedClusterSet** 的名称。

4. 通过自定义并应用类似以下示例的 YAML 内容，在受管集群中部署 Submariner：

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

将 **managed-cluster-name** 替换为您要使用 Submariner 的受管集群的名称。

ManagedClusterAddOn 的 spec 中的 **installNamespace** 字段是在受管集群上安装 Submariner 的命名空间。目前，Submariner 必须安装到 **submariner-operator** 命名空间中。

创建 **ManagedClusterAddOn** 后，**submariner-addon** 将 Submariner 部署到受管集群上的 **submariner-operator** 命名空间。您可以从这个 **ManagedClusterAddOn** 的状态查看 Submariner 的部署状态。

注： **ManagedClusterAddOn** 的名称必须是 **submariner**。

5. 对您要启用 Submariner 的所有受管集群重复三个和四个步骤。

6. 在受管集群中部署了 Submariner 后，您可以通过输入以下命令检查 submarinerr **ManagedClusterAddOn** 的状态来验证 Submariner 部署状态：

```
oc -n <managed-cluster-name> get managedclusteraddons submariner -oyaml
```

将 **managed-cluster-name** 替换为受管集群的名称。

在 Submariner **ManagedClusterAddOn** 的状态中，三个条件代表 Submariner 的部署状态：

- **SubmarinerGatewayNodesLabeled** 条件代表受管集群中是否存在标记为 Submariner 网关节点。
- **SubmarinerAgentDegraded** 条件指示 Submariner 是否成功部署到受管集群中。
- **SubmarinerConnectionDegraded** 条件指示受管集群上使用 Submariner 建立多少连接。

1.1.4.2.3. 自定义 Submariner 部署

您可以自定义 Submariner 部署的一些设置，包括网络地址转换-Traversal(NATT)端口、网关节点的数量以及网关节点的实例类型。这些自定义在所有提供程序中是一致的。

1.1.4.2.3.1. NATT 端口

如果要自定义 NATT 端口，请自定义并应用供应商环境的以下 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   credentialsSecret:
     name: <managed-cluster-name>-<provider>-creds
   IPSecNATTPort: <NATTPort>
```

- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 将 **managed-cluster-name** 替换为受管集群的名称
  - AWS：使用 **aws** 替换 **提供程序**。&lt **;managed-cluster-name>-aws-creds** 是 AWS 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该名称。
  - GCP：将 **供应商** 替换为 **gcp**。&lt **;managed-cluster-name>-gcp-creds** 是 Google Cloud Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到。
- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 将 **managed-cluster-name** 替换为受管集群的名称。**managed-cluster-name-gcp-creds** 的值是 Google Cloud Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该 secret。
- 将 **NATTPort** 替换为您要使用的 NATT 端口。

注： **SubmarinerConfig** 的名称必须是 **submariner**，如示例中所示。

要在 VMware vSphere 环境中自定义 NATT 端口，自定义并应用以下 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
    name: submariner
    namespace: <managed-cluster-namespace>
spec:
    IPSecNATTPort: <NATTPort>
```

- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 将 **NATTPort** 替换为您要使用的 NATT 端口。

注： **SubmarinerConfig** 的名称必须是 **submariner**，如示例中所示。

1.1.4.2.3.2. 网关节点数

如果要自定义网关节点的数量，自定义并应用类似以下示例的 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   credentialsSecret:
     name: <managed-cluster-name>-<provider>-creds
  gatewayConfig:
     gateways: <gateways>
```

- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 将 **managed-cluster-name** 替换为受管集群的名称。
  - AWS：使用 **aws** 替换 **提供程序**。**managed-cluster-name-aws-creds** 的值是 AWS 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该名称。
  - GCP：将 **供应商** 替换为 **gcp**。&lt **;managed-cluster-name>-gcp-creds** 是 Google Cloud Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到。
- 使用您要使用的网关数量替换 **gateway**。如果值大于 1，则 Submariner 网关会自动启用高可用性。

注： **SubmarinerConfig** 的名称必须是 **submariner**，如示例中所示。

如果要自定义 VMware vSphere 环境中网关节点的数量，请自定义并应用类似以下示例的 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
```

```
spec:
  gatewayConfig:
    gateways: <gateways>
```

- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 使用您要使用的网关数量替换 **gateway**。如果值大于 1，则 Submariner 网关会自动启用高可用性。

1.1.4.2.3.3. 网关节点的实例类型

如果要自定义网关节点的实例类型，请自定义并应用类似以下示例的 YAML 内容：

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
  gatewayConfig:
    instanceType: <instance-type>
```

- 将 **managed-cluster-namespace** 替换为受管集群的命名空间。
- 将 **managed-cluster-name** 替换为受管集群的名称。
  - AWS：使用 **aws** 替换 **提供程序**。**managed-cluster-name-aws-creds** 的值是 AWS 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到该名称。
  - GCP：将 **供应商** 替换为 **gcp**。&lt **;managed-cluster-name>-gcp-creds** 是 Google Cloud Platform 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到。
- 使用您要使用的 AWS 实例类型替换 **instance-type**。

注： **SubmarinerConfig** 的名称必须是 **submariner**，如示例中所示。

1.1.4.3. 为 Submariner 管理服务发现

在 Submariner 部署到与受管集群相同的环境中后，会配置路由用于受管集群的 pod 和服务间的安全 IP 路由。

1.1.4.3.1. 为 Submariner 启用服务发现

要使集群中的服务可见，并可以被受管集群集中的其他集群发现，您必须创建一个 **ServiceExport** 对象。使用 **ServiceExport** 对象导出服务后，您可以使用以下格式访问该服务：<service>.&lt **;namespace>.svc.clusterset.local**。如果多个集群导出具有相同名称的服务，并且来自同一命名空间中，则其他集群会把这个服务看作为一个单一的逻辑服务。

在本例在，在 **default** 命名空间中使用 **nginx** 服务，但您可以发现任何 Kubernetes **ClusterIP** 服务或无头服务：

1. 使用以下命令，在 **ManagedClusterSet** 中的受管集群中应用 **nginx** 服务实例：

```
oc -n default create deployment nginx --image=nginxinc/nginx-unprivileged:stable-alpine
oc -n default expose deployment nginx --port=8080
```

2. 使用类似以下命令的 **subctl** 工具创建一个 **ServiceExport** 条目来导出该服务：

```
subctl export service --namespace <service-namespace> <service-name>
```

将 **service-namespace** 替换为服务所在命名空间的名称。在本例中，是 **default**。

使用您要导出的服务的名称替换 **service-name**。在本例中是 **nginx**。

有关其他可用标记的更多信息，请参阅 Submariner 文档中的 **导出** 部分。

3. 在不同的受管集群中运行以下命令，确认它可以访问 **nginx** 服务：

```
oc -n default run --generator=run-pod/v1 tmp-shell --rm -i --tty --image
quay.io/submariner/nettest -- /bin/bash curl nginx.default.svc.clusterset.local:8080
```

nginx 服务发现现在已为 Submariner 配置。

1.1.4.3.2. 为 Submariner 禁用服务发现

要禁用服务被导出到其他集群，请为 **nginx** 输入类似以下示例的命令：

```
subctl unexport service --namespace <service-namespace> <service-name>
```

将 **service-namespace** 替换为服务所在命名空间的名称。

使用您要导出的服务的名称替换 **service-name**。

有关其他可用标记的更多信息，请参阅 Submariner 文档中的取消导出。
https://submariner.io/operations/deployment/subctl/#unexport

这个服务不再可用于集群发现。

1.1.4.4. 卸载 Submariner

您可以使用 Red Hat Advanced Cluster Management for Kubernetes 控制台或命令行从集群中卸载 Submariner 组件。对于 0.12 之前的 Submariner 版本，需要额外的步骤来完全删除所有 data plane 组件。

1.1.4.4.1. 控制台方法

要使用 Red Hat Advanced Cluster Management 控制台从集群卸载 Submariner，请完成以下步骤：

1.

在 Red Hat Advanced Cluster Management 控制台导航中，选择 Infrastructure > Clusters，然后选择 *Cluster sets* 选项卡。

2. 选择您要从中删除 Submariner 组件的集群的集群集。

3. 选择 Submariner Add-ons 选项卡在部署了 Submariner 的集群集中查看集群。

4. 在您要卸载 Submariner 的集群的 *Actions* 菜单中，选择 Uninstall Add-on。

5. 对要从中删除 Submariner 的其他集群重复这些步骤。

提示： 您可以通过选择多个集群并点 Actions，从同一集群集中的多个集群中删除 Submariner add-on。选择 Uninstall Submariner add-ons。

如果您要删除的 Submariner 版本早于 0.12 版本，请继续在 Submariner 的早期版本中手动删除步骤。如果 Submariner 版本为 0.12 或更高版本，则 Submariner 会被删除。

重要信息： 验证所有云资源已从云供应商中删除，以避免您的云供应商的额外费用。如需更多信息，请参阅 验证 Submariner 资源删除。

1.1.4.4.2. 命令行方法

要使用命令行卸载 Submariner，请完成以下步骤：

1. 输入以下命令查找包含 Submariner add-on 的集群：

```
oc get resource submariner-addon -n open-cluster-management
```

2. 运行类似以下示例的命令，从集群卸载 Submariner：

```
oc delete resource submariner-addon -n <CLUSTER_NAME>
```

将 CLUSTER_NAME 替换为集群的名称。

3. 确认您要从集群中删除所有 Submariner 组件。

4. 对每个集群重复这些步骤以删除 Submariner。

如果您要删除的 Submariner 版本早于 0.12 版本，请继续在 Submariner 的早期版本中手动删除步骤。如果 Submariner 版本为 0.12 或更高版本，则 Submariner 会被删除。

重要信息：验证所有云资源已从云供应商中删除，以避免您的云供应商的额外费用。如需更多信息，请参阅 验证 Submariner 资源删除。

1.1.4.4.3. 为 Submariner 的早期版本手动删除步骤

当卸载 0.12 版本早于 0.12 版本时，在 Submariner 文档中的 手动卸载 部分完成步骤 5-8。

完成这些步骤后，Submariner 组件会从集群中移除。

重要信息：验证所有云资源已从云供应商中删除，以避免您的云供应商的额外费用。如需更多信息，请参阅 验证 Submariner 资源删除。

1.1.4.4.4. 验证 Submariner 资源删除

卸载 Submariner 后，验证所有 Submariner 资源是否已从集群中移除。如果它们保留在集群上，某些资源将继续从基础架构供应商支付。通过完成以下步骤确保集群没有额外的 Submariner 资源：

1. 运行以下命令列出集群中保留的任何 Submariner 资源：

```
oc get cluster <CLUSTER_NAME> grep submariner
```

将 CLUSTER_NAME 替换为集群的名称。

2. 输入以下命令删除列表中的任何资源：

```
oc delete resource <RESOURCE_NAME> cluster <CLUSTER_NAME>
```

将 RESOURCE_NAME 替换为您要删除的 Submariner 资源的名称。

3. 对每个集群重复第 1-2 步，直到搜索不识别任何资源。

Submariner 资源从集群中移除。

1.2. VOLSYNC 持久性卷复制服务（技术预览）

VolSync 是一种 Kubernetes 操作器，支持异步复制集群中的持久性卷，或者在集群中使用存储类型不兼容进行复制的集群间复制。它使用容器存储接口(CSI)来克服兼容性限制。在您的环境中部署 VolSync Operator 后，您可以使用它来创建和维护持久数据的副本。VolSync 只能在位于版本 4.8 或更高版本的 Red Hat OpenShift Container Platform 集群中复制持久性卷声明。

- 使用 VolSync 复制持久性卷
- 将复制镜像转换为可用的持久性卷声明
- 调度同步

1.2.1. 使用 VolSync 复制持久性卷

您可以使用三种方法复制 VolSync 的持久性卷，这取决于您拥有的同步位置数量。本例中使用 Rsync 方法。有关其他方法以及 Rsync 的更多信息，请参阅 VolSync 文档中的使用情况。

rsync 复制常用于持久性卷的一个一对一复制。这用于将数据复制到远程站点。

VolSync 不自行创建命名空间，因此它与其他 OpenShift Container Platform all-namespace operator 位于同一个命名空间中。您对 VolSync 的 operator 设置所做的任何更改都会影响同一命名空间中的其他 Operator，例如，如果您更改为手动批准频道更新。

1.2.1.1. 先决条件

在集群上安装 VolSync 前，您必须满足以下要求：

- 配置了 Red Hat OpenShift Container Platform 环境，运行 Red Hat Advanced Cluster Management 版本 2.4 或更高版本、hub 集群

- 至少两个由同一 Red Hat Advanced Cluster Management hub 集群管理的集群

- 您要通过 VolSync 配置的集群之间的网络连接；如果集群不位于同一网络中，您可以配置 Submariner 多集群联网和服务发现，并使用 ServiceType 的 ClusterIP 值来联网 集群，或使用 ServiceType 值为 ServiceType 的负载均衡器。

- 您用于源持久性卷的存储驱动程序必须与 CSI 兼容，并能够支持快照。

1.2.1.2. 在受管集群上安装 VolSync

要启用 VolSync，将一个集群上的持久性卷声明复制到另一个集群的持久性卷声明中，您必须在源和目标集群上安装 VolSync。

您可以使用以下任一方法在环境中的两个集群中安装 VolSync。您可以向 hub 集群中的每个受管集群添加标签，也可以手动创建并应用 ManagedClusterAddOn，如下所示：

1.2.1.2.1. 使用标签安装 VolSync

通过添加标签，在受管集群中安装 VolSync。

- 从 Red Hat Advanced Cluster Management 控制台完成以下步骤：

  1. 从 hub 集群控制台的 Clusters 页面选择其中一个受管集群来查看其详情。

  2. 在 Labels 字段中，添加以下标签：

     ```
     addons.open-cluster-management.io/volsync=true
     ```

     VolSync 服务 pod 已安装在受管集群上。

  3.

3. 添加相同的标签，标记其他受管集群。

4. 在每个受管集群中运行以下命令，以确认已安装了 VolSync Operator：

```
oc get csv -n openshift-operators
```

安装时，会列出 VolSync 的 Operator。

- 使用命令行界面完成以下步骤：

1. 在 hub 集群中启动一个命令行会话。

2. 输入以下命令将标签添加到第一个集群：

```
oc label managedcluster <managed-cluster-1> "addons.open-cluster-
management.io/volsync"="true"
```

将 managed-cluster-1 替换为一个受管集群的名称。

3. 输入以下命令将标签添加到第二个集群：

```
oc label managedcluster <managed-cluster-2> "addons.open-cluster-
management.io/volsync"="true"
```

将 managed-cluster-2 替换为其他受管集群的名称。

在每个对应受管集群的命名空间的 hub 集群上自动创建一个 ManagedClusterAddOn 资源。

1.2.1.2.2. 使用 ManagedClusterAddOn 安装 VolSync

要通过手动添加 ManagedClusterSetAddOn 在受管集群中安装 VolSync，请完成以下步骤：

1.

在 hub 集群中，创建一个名为 volsync-mcao.yaml 的 YAML 文件，其中包含类似以下示例的内容：

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: volsync
  namespace: <managed-cluster-1-namespace>
spec: {}
```

将 managed-cluster-1-namespace 替换为其中一个受管集群的命名空间。此命名空间与受管集群的名称相同。

注： 名称必须是 volsync。

2. 输入类似以下示例的命令，将该文件应用到您的配置中：

```
oc apply -f volsync-mcao.yaml
```

3. 为其他受管集群重复上述步骤。

在每个对应受管集群的命名空间的 hub 集群上自动创建一个 ManagedClusterAddOn 资源。

1.2.1.3. 在受管集群中配置 Rsync 复制

对于基于 Rsync 的复制，请在源和目标集群上配置自定义资源。自定义资源使用 address 值将源连接到目的地，sshKeys 则用于确保传输的数据安全。

注： 您必须将 address 和 sshKeys 的值从目的地复制到源，因此请在配置源前配置目的地。

本例显示了一个步骤，将 Rsync 复制的配置从使用 source-ns 命名空间中的 source 集群中的持久性卷声明，改为使用 destination-ns 命名空间中的 destination 集群上的持久性卷声明。如果需要，您可以将这些值替换为其他值。

1. 配置您的目标集群。

a.

在目标集群中运行以下命令以创建命名空间：

```
$ kubectl create ns <destination-ns>
```

将 destination-ns 替换为将要包含目标持久性卷声明的命名空间的名称。

b.

复制以下 YAML 内容，以创建名为 replication_destination.yaml 的新文件：

```
---
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
  namespace: <destination-ns>
spec:
  rsync:
    serviceType: LoadBalancer
    copyMethod: Snapshot
    capacity: 2Gi
    accessModes: [ReadWriteOnce]
    storageClassName: gp2-csi
    volumeSnapshotClassName: csi-aws-vsc
```

注意：容量值应与正在复制的持久卷声明的容量匹配。

将 destination 替换为复制目的地 CR 的名称。

将 destination-ns 替换为目的地所在的命名空间的名称。

在本例中，使用 LoadBalancer 的 ServiceType 值。负载均衡器服务由源集群创建，以便您的源集群可以将信息传送到不同的目标受管集群。如果您的源和目标集群位于相同的集群中，或者配置了 Submariner 网络服务，您可以使用 ClusterIP 作为服务类型。请注意配置源集群时要指向的 secret 的地址和名称。

storageClassName 和 volumeSnapshotClassName 是可选参数。指定您的环境的值，特别是您使用的存储类和卷快照类名称与您的环境的默认值不同。

c.

在目标集群中运行以下命令以创建 replicationdestination 资源：

```
$ kubectl create -n <destination-ns> -f replication_destination.yaml
```

将 **destination-ns** 替换为目的地所在的命名空间的名称。

创建 **replicationdestination** 资源后，将以下参数和值添加到资源中：

参数	值
.status.rsync.address	用于连接的目标集群的 IP 地址，用于启用源和目标集群进行通信。
.status.rsync.sshKeys	启用保护从源集群到目标集群的数据传输的 SSH 密钥文件的名称。

d. 运行以下命令复制 **.status.rsync.address** 的值，以便在源集群中使用：

```
$ ADDRESS=`kubectl get replicationdestination <destination> -n <destination-ns> --
template={{.status.rsync.address}}`
$ echo $ADDRESS
```

将 **destination** 替换为复制目的地 **CR** 的名称。

将 **destination-ns** 替换为目的地所在的命名空间的名称。

输出结果应该类似以下示例，这适用于 **Amazon Web Services** 环境：

```
a831264645yhrjrjyer6f9e4a02eb2-5592c0b3d94dd376.elb.us-east-1.amazonaws.com
```

e. 运行以下命令复制 **secret** 的名称，以及作为 **.status.rsync.sshKeys** 的值提供的 **secret** 的内容。

```
$ SSHKEYS=`kubectl get replicationdestination <destination> -n <destination-ns> --
template={{.status.rsync.sshKeys}}`
$ echo $SSHKEYS
```

将 **destination** 替换为复制目的地 **CR** 的名称。

将 **destination-ns** 替换为目的地所在的命名空间的名称。

在配置源时，您必须在源集群中输入它。输出应该是 SSH 密钥 secret 文件的名称，该文件可能类似以下名称：

```
volsync-rsync-dst-src-destination-name
```

2. 找到您要复制的源持久性卷声明。

注： 源持久性卷声明必须位于 CSI 存储类中。

3. 创建 ReplicationSource 项。

a. 复制以下 YAML 内容，在源集群上创建一个名为 replication_source.yaml 的新文件：

```
---
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
spec:
  sourcePVC: <persistent_volume_claim>
  trigger:
    schedule: "*/3 * * * *"
  rsync:
    sshKeys: <mysshkeys>
    address: <my.host.com>
    copyMethod: Snapshot
    storageClassName: gp2-csi
    volumeSnapshotClassName: gp2-csi
```

将 **source** 替换为复制源 CR 的名称。有关如何 *自动替换该流程* 的说明，请参阅此流程的第 3 步。

将 **source-ns** 替换为源所在持久性卷声明的命名空间。有关如何 *自动替换该流程* 的说明，请参阅此流程的第 3 步。

将 persistent_volume_claim 替换为源持久性卷声明的名称。

使用您从 ReplicationDestination 的 .status.rsync.sshKeys 字段复制的密钥替换 mysshkeys。

将 my.host.com 替换为您在配置 ReplicationDestination 的 .status.rsync.address 字段复制的主机地址。

如果您的存储驱动程序支持克隆，使用 Clone 作为 copyMethod 的值，则可能是更精简的复制过程。

storageClassName 和 volumeSnapshotClassName 是可选参数。如果您使用与环境默认值不同的存储类和卷快照类名称，请指定这些值。

现在，您可以设置持久性卷的同步方法。

b. 通过针对目标集群输入以下命令从目标集群复制 SSH secret：

```
$ kubectl get secret -n <destination-ns> $SSHKEYS -o yaml > /tmp/secret.yaml
```

将 destination-ns 替换为目标所在持久性卷声明的命名空间。

c. 输入以下命令在 vi 编辑器中打开 secret 文件：

```
$ vi /tmp/secret.yaml
```

d. 在目标集群的 open secret 文件中进行以下更改：

- 将命名空间更改为源集群的命名空间。本例中是 source-ns。
- 删除所有者引用(.metadata.ownerReferences)。

e. **在源集群中，在源集群中输入以下命令来创建 secret 文件：**

```
$ kubectl create -f /tmp/secret.yaml
```

f. **在源集群中，使用以下命令将 ReplicationSource 对象中的 地址和 sshKeys 值替换为来自目标集群的值来修改 replication_source.yaml 文件：**

```
$ sed -i "s/<my.host.com>/$ADDRESS/g" replication_source.yaml
$ sed -i "s/<mysshkeys>/$SSHKEYS/g" replication_source.yaml
$ kubectl create -n <source> -f replication_source.yaml
```

将 my.host.com 替换为您在配置 ReplicationDestination 的 .status.rsync.address 字段复制的主机地址。

使用您从 ReplicationDestination 的 .status.rsync.sshKeys 字段复制的密钥替换 mysshkeys。

使用源所在的持久性卷声明的名称替换 source。

注： 您必须在与要复制的持久性卷声明相同的命名空间中创建该文件。

g. **在 ReplicationSource 对象中运行以下命令来验证复制是否完成：**

```
$ kubectl describe ReplicationSource -n <source-ns> <source>
```

将 source-ns 替换为源所在持久性卷声明的命名空间。

将 source 替换为复制源 CR 的名称。

如果复制成功，输出应类似以下示例：

```
Status:
  Conditions:
    Last Transition Time:  2021-10-14T20:48:00Z
    Message:               Synchronization in-progress
```

```
  Reason:              SyncInProgress
  Status:              True
  Type:                Synchronizing
  Last Transition Time:  2021-10-14T20:41:41Z
  Message:               Reconcile complete
  Reason:              ReconcileComplete
  Status:              True
  Type:                Reconciled
Last Sync Duration:      5m20.764642395s
Last Sync Time:          2021-10-14T20:47:01Z
Next Sync Time:          2021-10-14T20:48:00Z
```

如果 Last Sync Time 没有列出时间，则复制不会完成。

有原始持久性卷声明的副本。

1.2.2. 将复制镜像转换为可用的持久性卷声明

您可能需要使用复制的镜像来恢复数据，或创建持久性卷声明的新实例。镜像的副本必须转换为持久性卷声明，然后才能使用它。要将复制镜像转换为持久性卷声明，请完成以下步骤：

1. **复制完成后，输入以下命令识别 ReplicationDestination 对象的最新快照：**

```
$ kubectl get replicationdestination <destination> -n <destination-ns> --template=
{{.status.latestImage.name}}
```

记录下在创建持久性卷声明时的最新快照值。

将 destination 替换为复制目的地的名称。

将 destination-ns 替换为您的目的地的命名空间。

2. **创建一个类似以下示例的 pvc.yaml 文件：**

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <pvc-name>
  namespace: <destination-ns>
```

```
spec:
  accessModes:
    - ReadWriteOnce
  dataSource:
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
    name: <snapshot_to_replace>
  resources:
    requests:
      storage: 2Gi
```

将 pvc-name 替换为您的新持久性卷声明的名称。

将 destination-ns 替换为持久性卷声明所在的命名空间。

使用您在上一步中找到的 VolumeSnapshot 名称替换 snapshot_to_replace。

最佳实践： 当值至少与初始源持久性卷声明的大小相同时，您可以使用不同值更新 resources.requests.storage。

3. 输入以下命令验证您的持久性卷声明是否在环境中运行：

```
$ kubectl get pvc -n <destination-ns>
```

原始备份镜像作为主持久性卷声明运行。

1.2.3. 调度同步

在确定如何启动复制时，从三个选项中选择：始终运行、按照计划或手动运行。调度您的复制通常是经常选择的选项。

Schedule 选项在计划的时间运行复制。调度由 cronspec 定义，因此调度可配置为间隔或特定时间。调度值的顺序为：

"minute (0-59) hour (0-23) day-of-month (1-31) month (1-12) day-of-week (0-6)"

复制将在调度的时间发生时开始。您为此复制选项的设置可能类似以下内容：

```
spec:
  trigger:
    schedule: "*/6 * * * *"
```

启用其中一种方法后，同步调度会根据您配置的方法运行。

如需了解更多信息和选项，请参阅 VolSync 文档。

1.3. 在来自 KUBERNETES OPERATOR 的 MULTICLUSTER ENGINE 的集群上启用 KLUSTERLET 附加组件

安装 Red Hat Advanced Cluster Management for Kubernetes 后，然后使用 multicluster engine for Kubernetes operator 创建或导入集群，您可以为这些受管集群启用 klusterlet 附加组件。

如果使用 Kubernetes operator 的 multicluster engine 创建或导入的集群，则 klusterlet 附加组件不会被默认启用。另外，在安装 Red Hat Advanced Cluster Management 后，klusterlet 附加组件不会被默认启用。

请参阅以下可用的 klusterlet 附加组件：

- application-manager
- cert-policy-controller
- config-policy-controller
- iam-policy-controller
- governance-policy-framework

- search-collector

完成以下步骤，在安装 Red Hat Advanced Cluster Management 后为受管集群启用 klusterlet 附加组件：

1. 创建一个类似以下示例的 KlusterletAddonConfig 的 YAML 文件，其 spec 值代表附加组件：

```
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
```

注： policy-controller 附加组件被分成两个附加组件： governance-policy-framework 和 config-policy-controller。因此，policyController 控制 governance-policy-framework 和 config-policy-controller managedClusterAddons。

2. 将文件保存为 klusterlet-addon-config.yaml。

3. 通过在 hub 集群中运行以下命令来应用 YAML：

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

4. 要验证是否在创建 KlusterletAddonConfig 后创建启用的 managedClusterAddons，请运行以下命令：

```
oc get managedclusteraddons -n <cluster namespace>
```