Red Hat Advanced Cluster Management for Kubernetes 2.7

附加组件

请参阅更多信息，了解如何为集群使用附加组件。

Last Updated: 2024-09-05

请参阅更多信息，了解如何为集群使用附加组件。

法律通告

摘要

请参阅更多信息，了解如何为集群使用附加组件。

目录

第 1 章 附加组件概述 **3**
1.1. SUBMARINER MULTICLUSTER NETWORKING 和 SERVICE DISCOVERY 3
1.2. VOLSYNC 持久性卷复制服务 19
1.3. 在集群中为集群管理启用 KLUSTERLET 附加组件 31
1.4. 在现有集群附加组件上启用集群范围代理 32

第 1 章 附加组件概述

Red Hat Advanced Cluster Management for Kubernetes 附加组件可以提高某些性能方面，并添加用于增强应用程序的功能。以下小节提供了 Red Hat Advanced Cluster Management 可用的附加组件概述：

- Submariner multicluster networking 和 service discovery
- VolSync 持久性卷复制服务
- 在集群中为集群管理启用 klusterlet 附加组件
- 在现有集群附加组件上启用集群范围代理

1.1. SUBMARINER MULTICLUSTER NETWORKING 和 SERVICE DISCOVERY

Submariner 是一个开源工具，可与 Red Hat Advanced Cluster Management for Kubernetes 一起使用，用于在您的环境中（内部环境或云）提供两个或多个受管集群的直接网络和服务发现。Submariner 与 Multi-Cluster Services API (Kubernetes 增强 Proposal #1645) 兼容。有关 Submariner 的更多信息，请参阅 Submariner 站点。

有关基础架构供应商支持级别的详情，请查看 Red Hat Advanced Cluster Management 支持列表，包括哪些供应商支持 自动控制台部署或需要手动部署。

请参阅以下主题以了解有关如何使用 Submariner 的更多信息：

- 在断开连接的集群中部署 Submariner
- 配置 Submariner
- 安装 subctl 命令工具
- 使用控制台部署 Submariner
- 手动部署 Submariner
- 自定义 Submariner 部署
- 为 Submariner 管理服务发现
- 卸载 Submariner

1.1.1. 在断开连接的集群中部署 Submariner

在断开连接的集群中部署 Submariner 可以帮助出现安全问题，从而降低集群的外部攻击风险。要在断开连接的集群中使用 Red Hat Advanced Cluster Management for Kubernetes 部署 Submariner，您必须首先完成在断开连接的环境中安装中所述的步骤。

1.1.1.1. 在断开连接的集群中配置 Submariner

按照在断开连接的环境中安装中所述的步骤后，您必须在安装过程中配置 Submariner，以支持在断开连接的集群上部署。请参见以下主题：

1.1.1.1.1. 在本地 registry 中 mirror 镜像

在断开连接的集群上部署 Submariner 前，请确保在本地 registry 中部署 **Submariner Operator** 捆绑包镜像。

注：如果您使用 Red Hat Advanced Cluster Management 2.7.2 或更早版本，还必须镜像 **nettest-rhel8** 镜像。

1.1.1.1.2. 自定义 *catalogSource* 名称

默认情况下，**submariner-addon** 会搜索名为 **redhat-operators** 的 **catalogSource**。当使用具有不同名称的 **catalogSource** 时，您必须使用 **catalogSource** 自定义名称更新与您的受管集群关联的 **SubmarinerConfig** 中的 **SubmarinerConfig.Spec.subscriptionConfig.Source** 参数的值。

1.1.1.1.3. 在 *SubmarinerConfig* 中启用 *airGappedDeployment*

当从 Red Hat Advanced Cluster Management for Kubernetes 控制台在受管集群中安装 **submariner-addon** 时，您可以选择 **Disconnected cluster** 选项，以便 Submariner 不向外部服务器发出 API 查询。

如果要使用 API 安装 Submariner，您必须在与受管集群关联的 **SubmarinerConfig** 中将 **airGappedDeployment** 参数设置为 **true**。

1.1.2. 配置 Submariner

Red Hat Advanced Cluster Management for Kubernetes 提供 Submariner 作为 hub 集群的附加组件。您可以在 Submariner 开源项目文档中找到有关 Submariner 的更多信息。

1.1.2.1. 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。
- 在网关节点之间必须配置 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。
- 如果使用 OVN Kubernetes，集群必须位于 Red Hat OpenShift Container Platform 版本 4.11 或更高版本。
- 如果您的 Red Hat OpenShift Container Platform 集群使用 OpenShift SDN CNI，则每个受管集群中所有节点的防火墙配置必须同时允许 4800/UDP。
- 防火墙配置必须允许网关节点上的 4500/UDP 和 4490/UDP 在受管集群之间建立隧道。
- 如果网关节点可以通过其私有 IP 直接访问，且没有其中任何 NAT，请确保防火墙配置允许网关节点上的 ESP 协议。
  注：当您的集群部署在 Amazon Web Services、Google Cloud Platform、Microsoft Azure 或 Red Hat OpenStack 环境中时，这会自动配置，但必须为其他环境中的集群和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，并满足以下要求：
  - 包含 63 个字符或更少
  - 仅包含小写字母数字字符或 '-'
  - 以字母数字字符开头

- 以字母数字字符结尾

1.1.2.2. Submariner 端口表

查看下表以查看您需要启用的 Submariner 端口：

Name	默认值	Customizable	可选或必需的
IPsec NATT	4500/UDP	是	必填
VXLAN	4800/UDP	否	必填
NAT 发现端口	4490/UDP	否	必填

有关先决条件的详情，请参阅 Submariner 上游先决条件文档。

1.1.2.3. Globalnet

Globalnet 是 Submariner 附加组件中包含的功能，它支持具有重叠 CIDR 的集群之间的连接。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合中时，可以选择。启用 Globalnet 时，每个受管集群从虚拟全局专用网络中分配一个全局 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。在使用控制台时，**ClusterAdmin** 可以为集群集启用 Globalnet，方法是在为集群集中启用 Submariner 附加组件时选择 **Enable Globalnet** 选项。启用 Globalnet 后，您无法在不删除 Submariner 的情况下禁用它。

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 角色具有在代理命名空间中创建此对象所需的权限。**ManagedClusterSetAdmin** 角色 (有时被创建为作为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** 的角色权限与 **ManagedClusterSetAdmin** 用户关联。

完成以下步骤以创建 **submariner-broker** 对象：

1. 运行以下命令来检索 **<broker-namespace>** ：

```
oc get ManagedClusterSet <cluster-set-name> -o jsonpath="
{.metadata.annotations['cluster\.open-cluster-management\.io/submariner-broker-ns']}"
```

2. 创建一个 **submariner-broker** 对象，通过创建名为 **submariner-broker** 的 YAML 文件来指定 Globalnet 配置。在 YAML 文件中添加类似以下行的内容：

```
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。

注： 元数据 **name** 参数必须是 **submariner-broker**。

3. 输入以下命令将该文件应用到 YAML 文件中：

```
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。

1.1.3. 安装 subctl 命令工具

subctl 工具在容器镜像中提供。完成以下步骤，在本地安装 **subctl** 工具：

1. 运行以下命令登录到 registry，并在提示时输入您的凭证：

```
oc registry login --registry registry.redhat.io
```

2. 输入以下命令下载 **subctl** 容器，并将 **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:v0.14 --path="/dist/subctl-v0.14*-linux-amd64.tar.xz":/tmp/ --confirm
```

3. 输入以下命令解压缩 **subctl** 工具：

```
tar -C /tmp/ -xf /tmp/subctl-v0.14*-linux-amd64.tar.xz
```

4. 输入以下命令安装 **subctl** 工具：

```
install -m744 /tmp/subctl-v0.14*/subctl-v0.14*-linux-amd64 /$HOME/.local/bin/subctl
```

1.1.3.1. 使用 subctl 命令

在路径中添加工具后，请查看下表以了解可用命令的简短描述：

export service	为指定服务创建一个 **ServiceExport** 资源，它允许 Submariner 部署中的其他集群发现对应的服务。
unexport service	删除指定服务的 **ServiceExport** 资源，这可防止 Submariner 部署中的其他集群发现对应的服务。
show	提供有关 Submariner 资源的信息。
verify	当 Submariner 在一组集群中配置时，验证连接、服务发现和其他 Submariner 功能。
benchmark	使用 Submariner 或单个集群中启用的一对集群的基准测试吞吐量和延迟。

diagnose	运行检查以识别防止 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** 二进制工具的版本详情。

有关 **subctl** 工具及其命令的更多信息，请参阅 Submariner 文档中的**subctl** 部分。

1.1.4. 使用控制台部署 Submariner

在使用 Red Hat Advanced Cluster Management for Kubernetes 部署 Submariner 前，您必须在托管环境中准备集群。您可以使用 **SubmarinerConfig** API 或 Red Hat Advanced Cluster Management for Kubernetes 控制台在以下供应商上自动准备 Red Hat OpenShift Container Platform 集群：

- Amazon Web Services
- Google Cloud Platform
- Red Hat OpenStack Platform
- Microsoft Azure
- VMware vSphere

注： VMware vSphere 仅支持非NSX 部署。

要在其他供应商上部署 Submariner，请按照手动部署 Submariner 中的说明进行操作。

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。
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. 请参阅下表中的字段，并在 *Install Submariner add-ons* 编辑器中输入所需的信息：

字段	备注
AWS 访问密钥 **ID**	仅在导入 AWS 集群时可见。
AWS Secret 访问密钥	仅在导入 AWS 集群时可见。
基本域资源组名称	仅在导入 Azure 集群时可见。
客户端 **ID**	仅在导入 Azure 集群时可见。
客户端 **secret**	仅在导入 Azure 集群时可见。
订阅 **ID**	仅在导入 Azure 集群时可见。
租户 **ID**	仅在导入 Azure 集群时可见。
Google Cloud Platform 服务帐户 **JSON** 密钥	仅在导入 Google Cloud Platform 集群时可见。
实例类型	在受管集群上创建的网关节点的实例类型。
IPsec NAT-T 端口	IPsec NAT 遍历端口的默认值为 **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。
网关计数	要在受管集群中部署的网关节点数量。对于 AWS、GCP、Azure 和 OpenStack 集群，部署了专用网关节点。对于 VWware 集群，现有的 worker 节点被标记为网关节点。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)。
电缆驱动程序	维护跨集群隧道的 Submariner 网关电缆引擎组件。默认值为 **Libreswan IPsec**。
断开连接的集群	如果启用，请告知 Submariner 无法访问公共 IP 解析的任何外部服务器。
Globalnet CIDR	仅在集群集合中选择了 Globalnet 配置时才可见。用于受管集群的 Globalnet CIDR。如果留空，则会从集群设置池中分配一个 CIDR。

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。
    - 标签的网关节点表示受管集群中的网关节点数量。

Submariner 现在部署在所选集群中。

1.1.5. 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。请参阅使用控制台部署 Submariner 以了解如何使用控制台在支持的集群中自动部署 Submariner。

如果您的集群托管在不支持自动 Submariner 部署的供应商上，请参阅以下部分来手动准备基础架构。每个提供程序都有唯一的准备步骤，因此请确保选择正确的提供程序。

1.1.5.1. 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. 确保防火墙允许 4500/UDP 和 4490/UDP 端口上的外部客户端入站/出站流量。另外，如果集群使用 OpenShiftSDN CNI 部署，允许本地集群节点中的入站/出站 UDP/4800 流量。

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:
  gatewayConfig:
    gateways: 1
```

   将 **managed-cluster-namespace** 替换为受管集群的名称。**SubmarinerConfig** 的名称必须是 **submariner**，如示例所示。

   此配置将其中一个 worker 节点标记为裸机集群上的 Submariner 网关。

   默认情况下，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 用于连接。

3. 识别 Submariner 配置的网关节点，并启用防火墙配置，以允许用于外部流量的 IPsec NATT (UDP/4500) 和 NatDiscovery (UDP/4490) 端口。

如需有关自定义选项的信息，请参阅自定义 Submariner 部署。

1.1.5.2. 使用控制台为 Submariner 准备 Microsoft Azure Red Hat OpenShift （技术预览）

Microsoft Azure Red Hat OpenShift 服务组合了各种工具和资源，以帮助简化构建基于容器的应用程序的过程。要准备 Azure Red Hat OpenShift 集群以使用控制台部署 Submariner，请完成以下步骤：

1. 下载 Python wheel 和 CLI 扩展。
2. 在 Azure CLI 中运行以下命令安装扩展：

   ```
   az extension add --upgrade -s <path-to-extension>
   ```

   将 **path-to-extension** 替换为您下载 **.whl** 扩展文件的路径。
3. 运行以下命令验证是否使用了 CLI 扩展：

   ```
   az extension list
   ```

   如果使用扩展，输出可能类似以下示例：

   ```
   "experimental": false,
   "extensionType": "whl",
   "name": "aro",
   "path": "<path-to-extension>",
   "preview": true,
   "version": "1.0.x"
   ```
4. 在 Azure CLI 中，运行以下命令来注册 preview 功能：

   ```
   az feature registration create --namespace Microsoft.RedHatOpenShift --name
   AdminKubeconfig
   ```
5. 运行以下命令来检索管理员 **kubeconfig** ：

   ```
   az aro get-admin-kubeconfig -g <resource group> -n <cluster resource name>
   ```

   注： **az aro** 命令将 **kubeconfig** 保存到本地目录，并使用名称 **kubeconfig**。要使用它，设置环境变量 **KUBECONFIG** 以匹配文件的路径。请参见以下示例：

   ```
   export KUBECONFIG=<path-to-kubeconfig>
   oc get nodes
   ```
6. 从 Red Hat Advanced Cluster Management 控制台选择 **Infrastructure** > **Clusters** > **Import cluster**，将 Azure Red Hat OpenShift 集群导入到集群列表中。
7. 选择 **Kubeconfig** Import 模式，并在 Kubeconfig 窗口中输入 **kubeconfig** 文件中的内容。按照控制台中的说明完成导入。
   您可以通过进入到 **Infrastructure** > **Clusters** 来验证 Azure Red Hat OpenShift 集群是否已成功导入。
8. 进入到 **Infrastructure** > **Clusters** > **Cluster set**，然后选择要添加的集群集的名称。然后，点 **Submariner add-ons** 选项卡。

9. 点 **Install Submariner add-ons** 按钮，将 Azure Red Hat OpenShift 集群设置为您的 **Target** 集群。按照控制台中的说明完成安装。

10. 进入到 **Infrastructure** > **Clusters** > **Cluster sets** > **Submariner add-ons**，验证您的 Azure Red Hat OpenShift 集群的 **Connection status** 是 **Healthy**。

1.1.5.2.1. 使用 API 为 Submariner 准备 Microsoft Azure Red Hat OpenShift （技术预览）

要准备 Azure Red Hat OpenShift 集群以使用 API 部署 Submariner，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:
  loadBalancerEnable: true
```

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

SubmarinerConfig 的名称必须是 **submariner**，如示例所示。

此配置将其中一个 worker 节点标记为 Azure Red Hat OpenShift 集群上的 Submariner 网关。

默认情况下，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，如果没有指定 IPsec NATT 端口，会使用 4500/UDP 用于连接。

如需有关自定义选项的信息，请参阅自定义 Submariner 部署。

1.1.5.3. 使用控制台为 Submariner 准备 Red Hat OpenShift Service on AWS （技术预览）

Red Hat OpenShift Service on AWS 为应用程序开发和现代化提供了一个稳定而灵活的平台。要准备 OpenShift Service on AWS 集群来部署 Submariner，请完成以下步骤：

1. 运行以下命令，创建一个新节点来运行 Submariner 网关：

```
rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number of Submariner gateway > --labels='submariner.io/gateway=true'
```

2. 运行以下命令，登录到 OpenShift Service on AWS：

```
rosa login
oc login <rosa-cluster-url>:6443 --username cluster-admin --password <password>
```

3. 运行以下命令，在 AWS 集群上为 OpenShift Service 创建 **kubeconfig** ：

```
oc config view --flatten=true > rosa_kube/kubeconfig
```

4. 从 Red Hat Advanced Cluster Management 控制台选择 **Infrastructure** > **Clusters** > **Import cluster**，将 OpenShift Service on AWS 集群导入到集群列表中。

5. 选择 **Kubeconfig Import** 模式，并在 Kubeconfig 窗口中输入 **kubeconfig** 文件中的内容。按照控制台中的说明完成导入。
   您可以通过进入到 **Infrastructure** > **Clusters** 来验证 OpenShift Service on AWS 集群是否已成功导入。

6. 进入到 **Infrastructure** > **Clusters** > **Cluster set**，然后选择要添加的集群集的名称。然后，点 **Submariner add-ons** 选项卡。

7. 点 **Install Submariner add-ons** 按钮，并将 AWS 集群上的 OpenShift Service 设置为您的 **Target** 集群。按照控制台中的说明完成安装。

8. 进入到 **Infrastructure** > **Clusters** > **Cluster sets** > **Submariner add-ons**，验证您的 OpenShift Service on AWS 集群的 **Connection status** 为 **Healthy**。

1.1.5.3.1. 使用 API 为 Submariner 准备 Red Hat OpenShift Service on AWS （技术预览）

要使用 API 准备 OpenShift Service on AWS 集群以部署 Submariner，请完成以下步骤：

1. 运行以下命令，创建一个新节点来运行 Submariner 网关：

```
rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number of Submariner gateway > --labels='submariner.io/gateway=true'
```

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:
   loadBalancerEnable: true
```

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

   SubmarinerConfig 的名称必须是 **submariner**，如示例所示。

   默认情况下，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，如果没有指定 IPsec NATT 端口，会使用 4500/UDP 用于连接。

如需有关自定义选项的信息，请参阅自定义 Submariner 部署。

1.1.5.4. 使用 ManagedClusterAddOn API 部署 Submariner

手动准备所选托管环境后，您可以通过完成以下步骤来使用 **ManagedClusterAddOn** API 部署 Submariner：

1. 按照 Creating a *ManagedClusterSet* 在 hub 集群中创建一个 **ManagedClusterSet** 资源。确保您的 **ManagedClusterSet** 条目类似以下内容：

```
apiVersion: cluster.open-cluster-management.io/v1beta2
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。

重要：Kubernetes 命名空间的最大字符长度为 63 个字符。可用于 **<managed-cluster-set-name>** 的最大字符长度为 56 个字符。如果 **<managed-cluster-set-name>** 的字符长度超过 56 个字符，则 **<managed-cluster-set-name>** 会从头开始切断。

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 **<managed-cluster-set-name>-broker** 命名空间中的 hub 集群上创建 **Broker** 配置：

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
    name: submariner-broker
    namespace: <managed-cluster-set-name>-broker
    labels:
        cluster.open-cluster-management.io/backup: submariner
spec:
    globalnetEnabled: <true-or-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，请将 **globalnetEnabled** 的值设置为 **true**。

3. 运行以下命令，将一个受管集群添加到 **ManagedClusterSet** 中：

```
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. 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。

5. 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**。

6. 对您要启用 Submariner 的所有受管集群重复第三、第四和第五步骤。

7. 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。

1.1.6. 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(NATT)端口、网关节点数量和网关节点的实例类型。这些自定义在所有供应商间都是一致的。

1.1.6.1. NATT 端口

如果要自定义 NATT 端口，请自定义并应用您的供应商环境以下 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 **provider**。**<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它。
- GCP: 使用 **gcp** 替换 **provider**。**<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它。
- OpenStack ：将 **provider** 替换为 **osp**。**<managed-cluster-name>-osp-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它。
- Azure ：使用 **azure** 替换 **provider**。**<managed-cluster-name>-azure-creds** 的值是 Microsoft Azure 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到它。

- 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。

1.1.6.2. 网关节点数量

如果要自定义网关节点的数量，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:
  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 **provider**。**<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它。
  - GCP: 使用 **gcp** 替换 **provider**。**<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它。
  - OpenStack ：将 **provider** 替换为 **osp**。**<managed-cluster-name>-osp-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它。
  - Azure ：使用 **azure** 替换 **provider**。**<managed-cluster-name>-azure-creds** 的值是 Microsoft Azure 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到它。
- 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。

1.1.6.3. 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 **provider**。**<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它。
  - GCP: 使用 **gcp** 替换 **provider**。**<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它。
  - OpenStack ：将 **provider** 替换为 **osp**。**<managed-cluster-name>-osp-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它。
  - Azure ：使用 **azure** 替换 **provider**。**<managed-cluster-name>-azure-creds** 的值是 Microsoft Azure 凭证 secret 名称，您可以在 hub 集群的集群命名空间中找到它。
- 将 **instance-type** 替换为您要使用的 AWS 实例类型。

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

1.1.6.4. 电缆驱动程序

Submariner Gateway Engine 组件创建到其他集群的安全隧道。电缆驱动程序组件通过使用网关引擎组件中的可插拔架构来维护隧道。您可以使用 Libreswan 或 VXLAN 实现用于电缆引擎组件的 **cableDriver** 配置。请参见以下示例：

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

最佳实践：不要在公共网络上使用 VXLAN 电缆驱动程序。VXLAN 电缆驱动程序是未加密的。仅在为了避免在私有网络中进行不必要的双加密的情况下才使用 VXLAN。例如，一些内部环境可能会使用专用的线一级的硬件设备处理隧道的加密。

1.1.7. 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，会将路由配置为受管集群集中的 pod 和服务间的安全 IP 路由。

1.1.7.1. 为 Submariner 启用服务发现

要从集群可见服务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>.<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. 通过输入带有类似以下示例的 **subctl** 工具的命令创建一个 **ServiceExport** 条目来导出服务：

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

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

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

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。

1.1.7.2. 为 Submariner 禁用服务发现

要禁用将服务导出到其他集群，请为 **nginx** 输入一个类似以下示例的命令：

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

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

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。

集群不再可用于发现该服务。

1.1.8. 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删除 Submariner 组件。对于早于 0.12 的 Submariner 版本，需要额外的步骤来完全删除所有数据平面组件。Submariner uninstall 是幂等的，因此您可以在没有任何问题的情况下重复步骤。

1.1.8.1. 使用控制台卸载 Submariner

要使用控制台从集群卸载 Submariner，请完成以下步骤：

1. 在控制台导航中选择 **Infrastructure** > **Clusters**，然后选择 *Cluster sets* 选项卡。
2. 选择包含您要从中删除 Submariner 组件的集群集合。
3. 选择 **Submariner Add-ons** 选项卡来查看部署了 Submariner 的集群集合中的集群。
4. 在您要卸载 Submariner 的集群的 *Actions* 菜单中，选择 **Uninstall Add-on**。
5. 在您要卸载 Submariner 的集群的 *Actions* 菜单中，选择 **Delete cluster set**。
6. 对您要从中删除 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 附加组件。选择 **Uninstall Submariner add-ons**。

如果您要删除的 Submariner 版本早于 0.12 版本，请手动使用 Uninstalling Submariner。如果 Submariner 版本为 0.12 或更高版本，则 Submariner 会被删除。

重要：验证所有云资源是否已从云供应商中删除，以避免您的云供应商额外的费用。如需更多信息，请参阅验证 Submariner 资源删除。

1.1.8.2. 使用 CLI 卸载 Submariner

要使用命令行卸载 Submariner，请完成以下步骤：

1. 运行以下命令来删除集群的 Submariner 部署：

   ```
   oc -n <managed-cluster-namespace> delete managedclusteraddon submariner
   ```

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

2. 运行以下命令删除集群的云资源：

   ```
   oc -n <managed-cluster-namespace> delete submarinerconfig submariner
   ```

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

3. 运行以下命令删除集群集以删除代理详情：

   ```
   oc delete managedclusterset <managedclusterset>
   ```

   将 **managedclusterset** 替换为受管集群集的名称。

如果您要删除的 Submariner 版本早于 0.12 版本，请手动使用 Uninstalling Submariner。如果 Submariner 版本为 0.12 或更高版本，则 Submariner 会被删除。

重要：验证所有云资源是否已从云供应商中删除，以避免您的云供应商额外的费用。如需更多信息，请参阅验证 Submariner 资源删除。

1.1.8.3. 手动卸载 Submariner

卸载早于 0.12 版本的 Submariner 版本时，在 Submariner 文档中的 Manual Uninstall 部分中完成步骤 5-8。

完成这些步骤后，Submariner 组件将从集群中移除。

重要：验证所有云资源是否已从云供应商中删除，以避免您的云供应商额外的费用。如需更多信息，请参阅验证 Submariner 资源删除。

1.1.8.4. 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中，某些资源将继续获得基础架构供应商的费用。通过完成以下步骤，确保集群中没有额外的 Submariner 资源：

1. 运行以下命令列出集群中保留的所有 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. 对每个集群重复步骤 1-2，直到搜索无法识别任何资源。

Submariner 资源从集群中移除。

1.2. VOLSYNC 持久性卷复制服务

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 4.8 或更高版本的 Red Hat OpenShift Container Platform 集群上复制持久性卷声明。

重要： VolSync 只支持复制带有 **volumeMode** 的 **Filesystem** 的持久性卷声明。如果您没有选择 **volumeMode**，则默认为 **Filesystem**。

- 使用 VolSync 复制持久性卷
  - 在受管集群上安装 VolSync
  - 配置 Rsync 复制
  - 配置剩余的备份
  - 配置 Rclone 复制
- 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

您可以使用三种支持的方法来复制带有 VolSync 的持久性卷，这取决于您拥有的同步位置数量： Rsync、restic 或 Rclone。

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：

- 配置了运行 Red Hat Advanced Cluster Management 版本 2.4 或更高版本的 hub 集群的 Red Hat OpenShift Container Platform 环境
- 至少配置两个由同一 Red Hat Advanced Cluster Management hub 集群管理的集群
- 使用 VolSync 配置的集群之间的网络连接。如果集群不在同一网络中，您可以配置 Submariner multicluster networking 和 service discovery，并使用 **ServiceType** 的 **ClusterIP** 值来联网集群，或使用带有 **LoadBalancer** 值的 **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，您必须在源和目标受管集群中安装 VolSync。

VolSync 不创建自己的命名空间，因此它与其他 OpenShift Container Platform all-namespace operator 相同的命名空间中。对 VolSync 的 Operator 设置所做的任何更改也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。

您可以使用两种方式之一在环境中的两个集群中安装 VolSync。您可以为 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**，如以下部分所述：

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** 页面中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. 为其他受管集群添加相同的标签。
  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 VolSync 时，会列出该 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. 输入以下命令为第一个集群添加标签：

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

将 **managed-cluster-1** 替换为其中一个受管集群的名称。

3. 输入以下命令在第二个集群中添加标签：

```
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。

应该在每个对应受管集群的命名空间中自动创建一个 **ManagedClusterAddOn** 资源。

1.2.1.2.2. 使用 ManagedClusterAddOn 安装 VolSync

要通过手动添加 **ManagedClusterAddOn** 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。
应该在每个对应受管集群的命名空间中自动创建一个 **ManagedClusterAddOn** 资源。

1.2.1.3. 配置 Rsync 复制

您可以使用 Rsync 复制创建持久性卷的 1:1 异步复制。您可以使用基于 Rsync 的复制进行灾难恢复，或者将数据发送到远程站点。

以下示例演示了如何使用 Rsync 方法配置。有关 Rsync 的更多信息，请参阅 VolSync 文档中的使用情况。

1.2.1.3.1. 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：

   ```
   oc create ns <destination-ns>
   ```

   将 **destination-ns** 替换为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。如果您的源和目标位于同一集群中，或者配置了 Submariner 网络服务，则可以使用 **ClusterIP** 作为服务类型。记录配置源集群时要引用的 secret 的地址和名称。

   storageClassName 和 **volumeSnapshotClassName** 是可选参数。指定您的环境的值，特别是如果您使用与环境默认值不同的存储类和卷快照类名称。

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

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

将 **destination** 替换为复制目标自定义资源的名称。

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

将 **destination** 替换为复制目标自定义资源的名称。

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** 替换为复制源自定义资源的名称。有关如何替换此功能的说明，请参阅此流程的第 *3-vi* 步。

将 **source-ns** 替换为源所在持久性卷声明的命名空间。有关如何替换此功能的说明，请参阅此流程的第 *3-vi* 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：

```
oc 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：

```
oc create -f /tmp/secret.yaml
```

f. 在源集群中，输入以下命令替换 **ReplicationSource** 对象中的 **address** 和 **sshKeys** 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
oc 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：

```
oc 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** 替换为复制源自定义资源的名称。

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。

您有原始持久性卷声明的副本。

1.2.1.4. 配置剩余的备份

基于 restic 的备份将持久性卷的 restic 备份副本复制到在 **restic-config.yaml** secret 文件中指定的位置。剩余的备份不会在集群之间同步数据，而是提供数据备份。

完成以下步骤以配置基于剩余的备份：

1. 通过创建类似以下 YAML 内容的 secret 来指定存储备份镜像的存储库：

```
apiVersion: v1
kind: Secret
```

```
metadata:
  name: restic-config
type: Opaque
stringData:
  RESTIC_REPOSITORY: <my-restic-repository>
  RESTIC_PASSWORD: <my-restic-password>
  AWS_ACCESS_KEY_ID: access
  AWS_SECRET_ACCESS_KEY: password
```

将 **my-restic-repository** 替换为您要存储备份文件的 S3 存储桶存储库的位置。

将 **my-restic-password** 替换为访问存储库所需的加密密钥。

如果需要，将 **access** 和 **password** 替换为您的供应商凭证。如需更多信息，请参阅准备新存储库。

如果您需要准备新存储库，请参阅为流程准备新存储库。如果使用这个步骤，请跳过运行 **restic init** 命令的步骤来初始化存储库。VolSync 在第一个备份过程中自动初始化存储库。

重要：当将多个持久性卷声明备份到同一 S3 存储桶时，存储桶的路径对于每个持久性卷声明来说必须是唯一的。每个持久性卷声明都使用单独的 **ReplicationSource** 备份，每个声明都需要单独的 restic-config secret。

通过共享相同的 S3 存储桶，每个 **ReplicationSource** 具有对整个 S3 存储桶的写入访问权限。

2. 通过创建类似以下 YAML 内容的 **ReplicationSource** 对象来配置备份策略：

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: mydata-backup
spec:
  sourcePVC: <source>
  trigger:
    schedule: "*/30 * * * *" #\*
  restic:
    pruneIntervalDays: 14
    repository: <restic-config>
    retain:
      hourly: 6
      daily: 5
      weekly: 4
      monthly: 2
      yearly: 1
    copyMethod: Clone
  # The StorageClass to use when creating the PiT copy (same as source PVC if omitted)
  #storageClassName: my-sc-name
  # The VSC to use if the copy method is Snapshot (default if omitted)
  #volumeSnapshotClassName: my-vsc-name
```

使用您要备份的持久性卷声明替换 **source**。

将 **schedule** 值替换为运行备份的频率。这个示例有每 30 分钟的调度。如需更多信息，请参阅 调度同步。

将 **PruneIntervalDays** 值替换为重新打包数据实例之间经过的天数，以节省空间。修剪操作可在其运行时生成大量 I/O 流量。

将 **restic-config** 替换为在第 1 步中创建的 secret 的名称。

将 **retain** 的值设置为备份镜像的保留策略。

最佳实践： 将 **Clone** 用于 **CopyMethod** 的值，以确保保存点镜像。

有关备份选项的更多信息，请参阅 VolSync 文档中的备份选项。

注：默认情况下，Restic movers 在没有 root 权限的情况下运行。如果要以 root 用户身份运行 restic movers，请运行以下命令将升级的权限注解添加到您的命名空间。

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

将 **<namespace>** 替换为您的命名空间的名称。

1.2.1.4.1. 恢复剩余的备份

您可以将复制的数据从其余备份恢复到新的持久性卷声明。最佳实践：仅将一个备份恢复到新的持久性卷声明中。要恢复剩余的备份，请完成以下步骤：

1. 创建新的持久性卷声明，使其包含类似以下示例的新数据：

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

将 **pvc-name** 替换为新持久性卷声明的名称。

2. 创建一个 **ReplicationDestination** 自定义资源，该资源类似以下示例来指定恢复数据的位置：

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: <destination>
spec:
  trigger:
    manual: restore-once
  restic:
    repository: <restic-repo>
    destinationPVC: <pvc-name>
    copyMethod: Direct
```

将 **destination** 替换为复制目的地 CR 的名称。

使用存储源的仓库的路径替换 **restic-repo**。

使用您要恢复数据的新持久性卷声明的名称替换 **pvc-name**。使用现有的持久性卷声明，而不是置备一个新的持久性卷声明。

恢复过程只需要完成一次，本例恢复最新的备份。有关恢复选项的更多信息，请参阅 VolSync 文档中的恢复选项。

1.2.1.5. 配置 Rclone 复制

Rclone 备份通过中间对象存储位置（如 AWS S3）使用 Rclone 将单个持久性卷复制到多个位置。将数据分发到多个位置时非常有用。

完成以下步骤以配置 Rclone 复制：

1. 创建一个类似以下示例的 **ReplicationSource** 自定义资源：

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationSource
metadata:
  name: <source>
  namespace: <source-ns>
spec:
  sourcePVC: <source-pvc>
  trigger:
    schedule: "*/6 * * * *" #\*
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    storageClassName: <my-sc-name>
    volumeSnapshotClassName: <my-vsc>
```

将 **source-pvc** 替换为复制源自定义资源的名称。

将 **source-ns** 替换为源所在持久性卷声明的命名空间。

使用您要复制的持久性卷声明替换 **source**。

将 **schedule** 值替换为运行复制的频率。这个示例有每 6 分钟的调度。这个值必须在引号内。如需更多信息，请参阅调度同步。

将 **intermediate-s3-bucket** 替换为 Rclone 配置文件配置部分的路径。

将 **destination-bucket** 替换为您要复制文件的对象存储桶的路径。

将 **rclone-secret** 替换为包含您的 Rclone 配置信息的 secret 名称。

将 **copyMethod** 的值设置为 **Clone**、**Direct** 或 **Snapshot**。这个值指定是否生成点时复制，如果是，则使用什么方法生成它。

将 **my-sc-name** 替换为您要用于点复制的存储类的名称。如果没有指定，则使用源卷的存储类。

如果您将 **my-vsc** 指定为 **copyMethod**，则将 **my-vsc** 替换为 **VolumeSnapshotClass** 的名称。对于其他类型的 **copyMethod**，这并不是必需的。

2. 创建一个类似以下示例的 **ReplicationDestination** 自定义资源：

```
apiVersion: volsync.backube/v1alpha1
kind: ReplicationDestination
metadata:
  name: database-destination
  namespace: dest
spec:
  trigger:
    schedule: "3,9,15,21,27,33,39,45,51,57 * * * *" #/*
  rclone:
    rcloneConfigSection: <intermediate-s3-bucket>
    rcloneDestPath: <destination-bucket>
    rcloneConfig: <rclone-secret>
    copyMethod: Snapshot
    accessModes: [ReadWriteOnce]
    capacity: 10Gi
    storageClassName: <my-sc>
    volumeSnapshotClassName: <my-vsc>
```

将 **schedule** 值替换为将复制移到目的地的频率。源和目标的调度必须是偏移的，以允许数据在从目的地拉取前完成复制。这个示例有每 6 分钟的调度，将偏移 3 分钟。这个值必须在引号内。如需更多信息，请参阅调度同步。

将 **intermediate-s3-bucket** 替换为 Rclone 配置文件配置部分的路径。

将 **destination-bucket** 替换为您要复制文件的对象存储桶的路径。

将 **rclone-secret** 替换为包含您的 Rclone 配置信息的 secret 名称。

将 **copyMethod** 的值设置为 **Clone**、**Direct** 或 **Snapshot**。这个值指定是否生成点时复制，如果是，则使用什么方法生成它。

accessModes 的值指定持久性卷声明的访问模式。有效值为 **ReadWriteOnce** 或 **ReadWriteMany**。

capacity 指定目标卷的大小，它必须足够大来包含传入的数据。

将 **my-sc** 替换为您要用作点时副本的存储类的名称。如果没有指定，则使用系统存储类。

如果您将 **my-vsc** 指定为 **copyMethod**，则将 **my-vsc** 替换为 **VolumeSnapshotClass** 的名称。对于其他类型的 **copyMethod**，这并不是必需的。如果没有包括，则使用系统默认 **VolumeSnapshotClass**。

注：默认情况下，Rclone movers 运行没有 root 权限。如果要以 root 用户身份运行 Rclone movers，请运行以下命令将升级的权限注解添加到您的命名空间。

```
oc annotate namespace <namespace> volsync.backube/privileged-movers=true
```

将 **<namespace>** 替换为您的命名空间的名称。

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镜像来恢复数据，或者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大小相同时，您可以使用不同的值更新 **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>
```

您的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、按计划或手动运行。调度复制是一个经常选择的选项。

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. 在集群中为集群管理启用 KLUSTERLET 附加组件

安装 Red Hat Advanced Cluster Management for Kubernetes 后，然后使用多集群引擎 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。如果您创建或导入集群，否则 Red Hat Advanced Cluster Management 控制台创建或导入，则不会启用 klusterlet 附加组件。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. 创建一个类似于以下 **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** 附加组件分为两个附加组件： **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. 在 hub 集群中运行以下命令来应用 YAML：

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

4. 要验证在创建 **KlusterletAddonConfig** 后是否创建了已启用的 **managedClusterAddons**，请运行以下命令：

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

1.4. 在现有集群附加组件上启用集群范围代理

您可以在集群命名空间中配置 **KlusterletAddonConfig**，将代理环境变量添加到受管 Red Hat OpenShift Container Platform 集群的所有 klusterlet 附加组件 pod 中。完成以下步骤以配置 **KlusterletAddonConfig**，将三个环境变量添加到 klusterlet 附加组件的 pod 中：

1. 编辑位于需要代理的集群的 **KlusterletAddonConfig** 文件。您可以使用控制台查找资源，也可以使用以下命令在终端中编辑：

```
oc -n <my-cluster-name> edit klusterletaddonconfig <my-cluster-name>
```

注： 如果您只使用一个集群，则不需要命令末尾的 **<my-cluster-name>**。使用以下命令：

```
oc -n <my-cluster-name> edit klusterletaddonconfig
```

2. 编辑文件的 **.spec.proxyConfig** 部分，使其类似以下示例。**spec.proxyConfig** 是一个可选部分：

```
spec
  proxyConfig:
    httpProxy: "<proxy_not_secure>"
    httpsProxy: "<proxy_secure>"
    noProxy: "<no_proxy>"
```

将 **proxy_not_secure** 替换为 **http** 请求的代理服务器的地址。例如，使用 **http://192.168.123.145:3128**。

使用 **https** 请求的代理服务器的地址替换 **proxy_secure**。例如，使用 **https://192.168.123.145:3128**。

使用以逗号分隔的 IP 地址、主机名和域名列表替换 **no_proxy**，其中不会通过代理路由流量。例如，使用 **.cluster.local,.svc,10.128.0.0/14,example.com**。

如果使用在 hub 集群上配置的集群范围内的代理创建 OpenShift Container Platform 集群，则集群范围的代理配置值会在满足以下条件时添加到 klusterlet add-ons 的 pod 中：

- **addon** 部分中的 **.spec.policyController.proxyPolicy** 被启用并设置为 **OCPGlobalProxy**。
- **.spec.applicationManager.proxyPolocy** 被启用并设置为 **CustomProxy**。
  注： **addon** 部分中的 **proxyPolicy** 默认值是 **Disabled**。

  请参阅以下 **proxyPolicy** 条目示例：

```
apiVersion: agent.open-cluster-management.io/v1
   kind: KlusterletAddonConfig
   metadata:
     name: clusterName
```

```
  namespace: clusterName
spec:
  proxyConfig:
    httpProxy: http://pxuser:12345@10.0.81.15:3128
    httpsProxy: http://pxuser:12345@10.0.81.15:3128
    noProxy: .cluster.local,.svc,10.128.0.0/14, example.com
  applicationManager:
    enabled: true
    proxyPolicy: CustomProxy
  policyController:
    enabled: true
    proxyPolicy: OCPGlobalProxy
  searchCollector:
    enabled: true
    proxyPolicy: Disabled
  certPolicyController:
    enabled: true
    proxyPolicy: Disabled
  iamPolicyController:
    enabled: true
    proxyPolicy: Disabled
```

重要： 全局代理设置不会影响警报转发。要为使用集群范围代理的 Red Hat Advanced Cluster Management hub 集群设置警报转发，请参阅 转发警报 以了解更多详细信息。