Red Hat OpenShift Data Foundation 4.15

为 OpenShift Workloads 配置 OpenShift Data Foundation 灾难恢复

Metropolitan 和 Regional 区域的 OpenShift Data Foundation 灾难恢复功能现已正式发布，它还包括使用扩展集群的灾难恢复。

Last Updated: 2024-05-18

Red Hat OpenShift Data Foundation 4.15 为 OpenShift Workloads 配置 OpenShift Data Foundation 灾难恢复

Metropolitan 和 Regional 区域的 OpenShift Data Foundation 灾难恢复功能现已正式发布，它还包括使用扩展集群的灾难恢复。

法律通告

摘要

本指南旨在详细介绍使用 Advanced Cluster Management 部署 OpenShift Data Foundation 进行灾难恢复所需的步骤，并扩展集群来实现高度可用的存储基础架构。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请告诉我们如何让它更好。

要提供反馈，请创建一个 Bugzilla ticket：

1. 进入 Bugzilla 网站。
2. 在 **Component** 部分中，选择 **文档**。
3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
4. 点 **Submit Bug**。

第 1 章 OPENSHIFT DATA FOUNDATION 灾难恢复简介

灾难恢复（DR）是从自然或人为的灾难中恢复并继续业务关键应用程序的能力。它是任何主要组织的整体业务整合战略的一个组件，其设计旨在在重要事件期间保持业务操作的连续性。

OpenShift Data Foundation DR 功能可在多个 Red Hat OpenShift Container Platform 集群间启用 DR，并分类如下：

- **Metro-DR**
  Metro-DR 可确保在数据中心出现问题时保持业务的连续性，并不会造成数据丢失。在公有云中，它们类似于防止可用性区域失败。

- **Regional-DR**
  区域 DR 可以在一个地理区域出现问题时确保业务的连续性（在可以接受一些可预测数量的数据丢失的情况下）。在公有云中，它们类似于防止区域故障。

- **使用扩展集群进行灾难恢复**
  扩展集群解决方案可确保在单一 OpenShift 集群中通过基于 OpenShift Data Foundation 的同步复制，在具有低延迟和一个仲裁节点的两个数据中心间扩展，且无数据丢失灾难恢复保护。

Metro-DR 中的区故障以及 Regional-DR 中的区域故障通常使用术语 **Recovery Point Objective (RPO)** 和 **Recovery Time Objective (RTO)**。

- **RPO** 是一种衡量持久性数据备份或快照的频率。实际上，RPO 表示在中断后将丢失或需要重新输入的数据量。

- **RTO** 是企业可以容忍的停机时间。RTO 回答了这个问题，"在收到业务中断通知后，我们的系统需要多久才能恢复？"

本指南旨在详细介绍灾难恢复步骤和命令，将应用程序从一个 OpenShift Container Platform (OCP) 集群切换到另一个集群，然后将同一应用程序恢复到原始集群。

第 2 章 灾难恢复订阅要求

Red Hat OpenShift Data Foundation 支持的灾难恢复功能需要满足以下所有先决条件，才能成功实施灾难恢复解决方案：

- 有效的 Red Hat OpenShift Data Foundation 高级授权
- 有效的 Red Hat Advanced Cluster Management for Kubernetes 订阅

任何包含 PV（包括作为源或目标）的 PV 的 Red Hat OpenShift Data Foundation 集群都需要 OpenShift Data Foundation 高级授权。此订阅应该在源和目标集群上处于活跃状态。

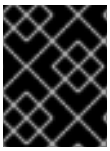
要了解 OpenShift Data Foundation 订阅如何工作，请参阅与 OpenShift Data Foundation 订阅相关的知识库文章。

第 3 章 用于 OPENSHIFT DATA FOUNDATION 的 METRO-DR 解决方案

本指南详细介绍了区域灾难恢复(Metro DR)步骤和命令，可以将应用程序从一个 OpenShift Container Platform 集群切换到另一个集群，然后将同一应用程序恢复到原始集群。在这种情况下，会使用 Red Hat Advanced Cluster Management (RHACM) 创建或导入 OCP 集群，且各个 OCP 集群间的距离小于 10ms RTT 延迟。

应用程序的持久存储将由外部 Red Hat Ceph Storage (RHCS) 集群提供，在连接此存储集群的 OpenShift Container Platform 实例的两个位置之间扩展。在站点出现问题时，在第三个位置中（与部署 OpenShift Container Platform 实例不同的位置）需要一个带有存储监控器服务的仲裁节点用于为 RHCS 集群创建仲裁。第三个位置可以位于连接到 OpenShift Container Platform 实例的存储集群中的 ~100ms RTT 范围。

这是在由距离分开的两个不同 OpenShift Container Platform 集群中使用 OpenShift Data Foundation 和 RHACM 配置和执行 OpenShift 灾难恢复 (ODR) 功能所需的 Metro DR 步骤的一般概述。除了这两个被称为受管集群的集群外，还需要第三个 OpenShift Container Platform 集群，它将是 Red Hat Advanced Cluster Management (RHACM) hub 集群。



重要

现在，您可以使用 OpenShift Data Foundation 为基于 OpenShift virtualization 技术的工作负载轻松设置 Metropolitan 灾难恢复解决方案。如需更多信息，请参阅知识库文章。

3.1. METRO-DR 解决方案的组件

Metro-DR 由 Red Hat Advanced Cluster Management for Kubernetes、Red Hat Ceph Storage 和 OpenShift Data Foundation 组件组成，以便在 OpenShift Container Platform 集群中提供应用程序和数据移动性。

Red Hat Advanced Cluster Management for Kubernetes

Red Hat Advanced Cluster Management (RHACM)提供了管理多个集群和应用程序生命周期的功能。因此，它充当多集群环境中的控制平面。

RHACM 分为两个部分：

- RHACM Hub：在多集群 control plane 上运行的组件。
- 受管集群：在受管理的集群中运行的组件。

有关此产品的更多信息，请参阅 RHACM 文档 和 RHACM"管理应用程序"文档。

Red Hat Ceph Storage

Red Hat Ceph Storage 是一个可大规模扩展、开放、软件定义的存储平台，它将最稳定版本的 Ceph 存储系统与 Ceph 管理平台、部署实用程序和支持服务相结合。它显著降低了存储企业数据的成本，帮助组织管理指数级数据增长。此软件是一款强大、现代化的 PB 级存储平台，适用于公有云或私有云部署。

如需更多信息，请参阅 Red Hat Ceph Storage 。

OpenShift Data Foundation

OpenShift Data Foundation 为 OpenShift Container Platform 集群中有状态应用程序提供部署和管理存储的功能。它由 Ceph 作为存储提供商提供支持，其生命周期由 OpenShift Data Foundation 组件堆栈和 Ceph-CSI 管理，它们的生命周期为有状态应用提供持久卷的调配和管理。

OpenShift DR

OpenShift DR 是跨一组使用 RHACM 部署和管理的有状态应用程序的灾难恢复编排器，并提供云原生接口来编排应用程序状态在持久性卷上的生命周期。它们是：

- 保护跨 OpenShift 集群的应用程序及其状态关系
- 将一个应用程序及其状态转移到一个对等集群
- 将一个应用程序及其状态重新定位到之前部署的集群

OpenShift DR 被分成三个组件：

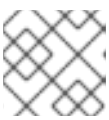
- **ODF Multicluster Orchestrator**: 安装在多集群控制平面 (RHACM Hub) 中，它会编配配置并针对 Metro 和 Regional D 关系对等 OpenShift Data Foundation 集群。
- **OpenShift DR Hub Operator**：在 hub 集群上作为 ODF 多集群安装的一部分自动安装，以编配启用 DR 应用程序的故障转移或重新定位。
- **OpenShift DR Cluster Operator**：在属于 Metro 和 Regional DR relationship的每个受管集群中自动安装，用于管理应用程序的所有 PVC 的生命周期。

3.2. METRO-DR 部署工作流

本节概述使用最新版本的 Red Hat OpenShift Data Foundation、Red Hat Ceph Storage (RHCS)和 Red Hat Advanced Cluster Management for Kubernetes (RHACM)版本 2.10 或更高版本，在两个不同的 OpenShift Container Platform 集群中配置和部署 Metro-DR 功能所需的步骤。除了两个受管集群外，还需要第三个 OpenShift Container Platform 集群来部署 Advanced Cluster Management。

要配置基础架构，请按照给定的顺序执行以下步骤：

1. 确保满足作为 DR 解决方案的三个集群（Hub、Primary 和 Secondary Openshift Container Platform）集群的要求。请参阅启用 Metro-DR 的要求。
2. 确保您具有仲裁者部署 Red Hat Ceph Storage 扩展集群的要求。请参阅部署 Red Hat Ceph Storage 的要求。
3. 部署和配置 Red Hat Ceph Storage 扩展模式。有关使用扩展模式功能在两个不同数据中心启用 Ceph 集群的说明，请参阅部署 Red Hat Ceph Storage 。
4. 安装 OpenShift Data Foundation operator，并在 Primary 和 Secondary 受管集群上创建存储系统。请参阅在受管集群中安装 OpenShift Data Foundation。
5. 在 Hub 集群上安装 ODF Multicluster Orchestrator。请参阅在 Hub 集群上安装 ODF 多集群编排器。
6. 配置 Hub、Primary 和 Secondary 集群之间的 SSL 访问。请参阅配置跨集群的 SSL 访问。
7. 创建 DRPolicy 资源，以用于在跨 Primary 和 Secondary 应用程序所需的 DR 保护的应用程序。请参阅在 Hub 集群上创建灾难恢复策略。



注意

Metro-DR 解决方案只能有一个 DRpolicy。

8. 测试您的灾难恢复解决方案：

a. **基于订阅的应用程序** ：

- 创建示例应用程序。请参阅 创建示例应用程序。
- 在受管集群之间使用示例应用程序测试故障切换和重定位操作。请参阅基于订阅的应用程序故障切换 和 重新定位基于订阅的应用程序。

b. **基于 ApplicationSet 的应用程序**：

- 创建示例应用程序。请参阅 创建基于 ApplicationSet 的应用程序。
- 在受管集群之间使用示例应用程序测试故障切换和重定位操作。请参阅 基于 ApplicationSet 的应用程序故障切换和 重新定位基于 ApplicationSet 的应用程序。

3.3. 启用 METRO-DR 的要求

安装 Red Hat OpenShift Data Foundation 支持的灾难恢复解决方案的先决条件如下：

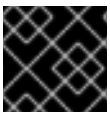
- 您必须有以下在它们之间具有网络可访问性的 OpenShift 集群：
  - 安装 Red Hat Advanced Cluster Management (RHACM) for Kubernetes operator 的 hub **集群**。
  - 运行 OpenShift Data Foundation 的**主受管集群**。
  - 运行 OpenShift Data Foundation 的**从受管集群**。

注意

要配置 hub 恢复设置，您需要 4 个集群，它充当被动 hub。主受管集群(Site-1)可以与活跃的 RHACM hub 集群共存，而被动 hub 集群与二级受管集群(Site-2)一起存在。或者，活跃的 RHACM hub 集群可以放在中立站点(Site-3)中，该集群不受 Site-1 或 Site-2 中主受管集群的故障的影响。在这种情况下，如果使用被动 hub 集群，它可以被放在 Site-2 的次要集群中。如需更多信息，请参阅为 hub 恢复配置被动 hub 集群。

hub 恢复是一个技术预览功能，受技术预览支持限制。

- 确保在 Hub 集群中安装 RHACM 操作符和 MultiClusterHub。具体步骤请查看 RHACM 安装指南。
  成功安装 Operator 后，用户界面中会显示一个带有 Web 控制台更新可用消息的弹出窗口。点此弹出窗口中的 **Refresh Web 控制台**来反映控制台更改。



重要

确保正确配置了应用程序流量路由和重定向。

- 在 Hub 集群中
  - 进入到 **All Clusters → Infrastructure → Clusters**。
  - 使用 RHACM 控制台导入或创建**主受管集群**和**次受管集群**。
  - 为您的环境选择适当的选项。

成功创建或导入受管集群后，您可以看到在控制台中导入或创建的集群列表。具体步骤请参阅创建集群并将 目标受管集群导入到 hub 集群。

警告

Openshift Container Platform 受管集群和 Red Hat Ceph Storage (RHCS) 节点有距离的限制。站点之间的网络延迟必须低于 10 毫秒往返时间 (RTT)。

3.4. 使用仲裁器部署 RED HAT CEPH STORAGE 的要求

Red Hat Ceph Storage 是一个开源企业平台，可为标准、经济的服务器和磁盘提供统一软件定义型存储。通过将块、对象和文件存储整合为一个平台，Red Hat Ceph Storage 可以高效并自动管理所有数据，因此您可以专注于使用它的应用程序和工作负载。

本节提供了 Red Hat Ceph Storage 部署的基本概述。如需更复杂的部署，请参阅 Red Hat Ceph Storage 7 的官方文档指南。

注意

仅支持 Flash 介质，因为它在降级时使用 **min_size=1**。仅对 all-flash OSD 使用扩展模式。使用 all-flash OSD 会最大程度减少在恢复连接后恢复所需的时间，从而尽量减少数据丢失的可能性。

重要

纠删代码池无法用于扩展模式。

3.4.1. 硬件要求

有关部署 Red Hat Ceph Storage 的最低硬件要求的信息，请参阅 容器化 Ceph 的最低硬件建议。

表 3.1. Red Hat Ceph Storage 集群部署的物理服务器位置和 Ceph 组件布局：

节点名	数据中心	Ceph 组件
ceph1	DC1	OSD+MON+MGR
ceph2	DC1	OSD+MON
ceph3	DC1	OSD+MDS+RGW
ceph4	DC2	OSD+MON+MGR
ceph5	DC2	OSD+MON
ceph6	DC2	OSD+MDS+RGW

节点名	数据中心	Ceph 组件
ceph7	DC3	MON

3.4.2. 软件要求

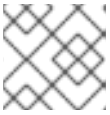
使用 **Red Hat Ceph Storage 7**的最新软件版本。

有关 Red Hat Ceph Storage 支持的操作系统版本的更多信息，请参阅 Red Hat Ceph Storage: 支持的配置。

3.4.3. 网络配置要求

推荐的 Red Hat Ceph Storage 配置如下：

- 您必须有两个独立的网络，一个公共网络和一个专用网络。
- 您必须有三个不同的数据中心，支持所有数据中心的 Ceph 私有和公共网络的 VLAN 和子网。



注意

您可以为每个数据中心使用不同的子网。

- 运行 Red Hat Ceph Storage Object Storage Devices (OSD) 的两个数据中心之间的延迟不能超过 10 ms RTT。对于 **arbiter** 数据中心，这已进行了测试，其值为 100 ms RTT 到其他两个 OSD 数据中心。

以下是我们在本指南中使用的基本网络配置示例：

- **DC1: Ceph public/private network:**10.0.40.0/24
- **DC2: Ceph public/private network:**10.0.40.0/24
- **DC3: Ceph public/private network:**10.0.40.0/24

有关所需网络环境的更多信息，请参阅 Ceph 网络配置。

3.5. 部署 RED HAT CEPH STORAGE

3.5.1. 节点预部署步骤

在安装 Red Hat Ceph Storage Ceph 集群前，请执行以下步骤来满足所有必要的要求。

1. 将所有节点注册到 Red Hat Network 或 Red Hat Satellite 中，并订阅到有效的池：

```
subscription-manager register
subscription-manager subscribe --pool=8a8XXXXXX9e0
```

2. 为以下软件仓库启用 Ceph 集群中的所有节点的访问权限：

   - **rhel9-for-x86_64-baseos-rpms**

- **rhel9-for-x86_64-appstream-rpms**

```
subscription-manager repos --disable="*" --enable="rhel9-for-x86_64-baseos-rpms" --
enable="rhel9-for-x86_64-appstream-rpms"
```

3. 如果需要，将操作系统 RPM 更新至最新版本并重新引导：

```
dnf update -y
reboot
```

4. 从集群中选择节点作为 bootstrap 节点。**ceph1** 是本例中的 bootstrap 节点。
仅在 bootstrap 节点 **ceph1** 上，启用 **ansible-2.9-for-rhel-9-x86_64-rpms** 和 **rhceph-6-tools-for-rhel-9-x86_64-rpms** 仓库：

```
subscription-manager repos --enable="ansible-2.9-for-rhel-9-x86_64-rpms" --
enable="rhceph-6-tools-for-rhel-9-x86_64-rpms"
```

5. 在所有主机中使用裸机/短主机名配置**主机名**。

```
hostnamectl set-hostname <short_name>
```

6. 使用 cephadm 验证用于部署 Red Hat Ceph Storage 的主机名配置。

```
$ hostname
```

输出示例：

```
ceph1
```

7. 使用 DNS 域名设置 DOMAIN 变量，修改 /etc/hosts 文件并将 fqdn 条目添加到 127.0.0.1 IP。

```
DOMAIN="example.domain.com"

cat <<EOF >/etc/hosts
127.0.0.1 $(hostname).${DOMAIN} $(hostname) localhost localhost.localdomain localhost4
localhost4.localdomain4
::1       $(hostname).${DOMAIN} $(hostname) localhost6 localhost6.localdomain6
EOF
```

8. 使用 **hostname -f** 选项通过 **fqdn** 检查长主机名。

```
$ hostname -f
```

输出示例：

```
ceph1.example.domain.com
```

注意

要了解更多有关需要这些更改的信息，请参阅完全限定域名和裸机主机名。

9. 在 bootstrap 节点上执行以下步骤。在我们的示例中，bootstrap 节点为 **ceph1**。

   a. 安装 **cephadm-ansible** RPM 软件包：

   ```
   $ sudo dnf install -y cephadm-ansible
   ```

   重要

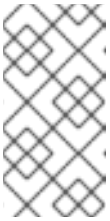
   要运行 ansible playbook，您必须有 **ssh** 免密码访问配置 Red Hat Ceph Storage 集群的所有节点。确保配置的用户（如 **deployment-user**）具有可调用 **sudo** 命令的 root 特权，而无需输入密码。

   b. 要使用自定义密钥，请配置所选用户（如 **deployment-user**）ssh 配置文件以指定将用于通过 ssh 连接到节点的 id/key：

   ```
   cat <<EOF > ~/.ssh/config
   Host ceph*
     User deployment-user
     IdentityFile ~/.ssh/ceph.pem
   EOF
   ```

   c. 构建 ansible 清单

   ```
   cat <<EOF > /usr/share/cephadm-ansible/inventory
   ceph1
   ceph2
   ceph3
   ceph4
   ceph5
   ceph6
   ceph7
   [admin]
   ceph1
   ceph4
   EOF
   ```

   注意

   此处，属于两个不同的数据中心的主机 (**Ceph1** and **Ceph4**) 配置为清单文件中的 [admin] 组的一部分，并使用 **cephadm** 标记为 **_admin**。每个管理节点都会在 bootstrap 过程中收到 admin ceph 密钥环，以便在一个数据中心停机时，我们都可以使用其他可用的管理节点进行检查。

   d. 在运行 pre-flight playbook 前，验证 **ansible** 是否可以使用 ping 模块访问所有节点。

   ```
   $ ansible -i /usr/share/cephadm-ansible/inventory -m ping all -b
   ```

   输出示例：

   ```
   ceph6 | SUCCESS => {
       "ansible_facts": {
           "discovered_interpreter_python": "/usr/libexec/platform-python"
       },
   ```

```
    "changed": false,
    "ping": "pong"
}
ceph4 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph3 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph5 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
ceph7 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```


e. 进入 **/usr/share/cephadm-ansible** 目录。

f. 使用相对文件路径运行 ansible-playbook。

```
$ ansible-playbook -i /usr/share/cephadm-ansible/inventory /usr/share/cephadm-ansible/cephadm-preflight.yml --extra-vars "ceph_origin=rhcs"
```

preflight playbook Ansible playbook 配置 RHCS **dnf** 存储库，并为引导准备存储集群。它还安装 podman、lvm2、chronyd 和 cephadm。**cephadm-ansible** 和 **cephadm-preflight.yml** 的默认位置为 **/usr/share/cephadm-ansible**。如需更多信息，请参阅 运行 preflight

playbook

3.5.2. 使用 cephadm 实用程序进行集群引导和服务部署

cephadm 实用程序将安装并启动一个单独的 Ceph Monitor 守护进程，以及运行 cephadm bootstrap 命令本地节点上的新 Red Hat Ceph Storage 集群的 Ceph 管理器守护进程。

在本指南中，我们将使用集群规格 yaml 文件，在一个步骤中使用集群规格 yaml 来部署所有需要的 Red Hat Ceph Storage 服务。

如果您在部署过程中发现问题，通过将部署分成两个步骤来更加轻松地对错误进行故障排除：

1. bootstrap
2. 服务部署

注意

有关 bootstrap 过程的更多信息， 请参阅引导新存储集群。

步骤

1. 使用 json 文件创建 json 文件以针对容器 registry 进行身份验证，如下所示：

```
$ cat <<EOF > /root/registry.json
{
 "url":"registry.redhat.io",
 "username":"User",
 "password":"Pass"
}
EOF
```

2. 创建一个 **cluster-spec.yaml**，将节点添加到 Red Hat Ceph Storage 集群，并为服务设置应当运行下表 3.1 的特定标签。

```
cat <<EOF > /root/cluster-spec.yaml
service_type: host
addr: 10.0.40.78  ## <XXX.XXX.XXX.XXX>
hostname: ceph1   ##  <ceph-hostname-1>
location:
  root: default
  datacenter: DC1
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.35
hostname: ceph2
location:
  datacenter: DC1
labels:
  - osd
  - mon
```

```
---
service_type: host
addr: 10.0.40.24
hostname: ceph3
location:
  datacenter: DC1
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: 10.0.40.185
hostname: ceph4
location:
  root: default
  datacenter: DC2
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.88
hostname: ceph5
location:
  datacenter: DC2
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.66
hostname: ceph6
location:
  datacenter: DC2
labels:
  - osd
  - mds
  - rgw
---
service_type: host
addr: 10.0.40.221
hostname: ceph7
labels:
  - mon
---
service_type: mon
placement:
  label: "mon"
---
service_type: mds
service_id: cephfs
placement:
  label: "mds"
---
```

```
service_type: mgr
service_name: mgr
placement:
  label: "mgr"
---
service_type: osd
service_id: all-available-devices
service_name: osd.all-available-devices
placement:
  label: "osd"
spec:
  data_devices:
    all: true
---
service_type: rgw
service_id: objectgw
service_name: rgw.objectgw
placement:
  count: 2
  label: "rgw"
spec:
  rgw_frontend_port: 8080
EOF
```

3. 使用从 bootstrap 节点配置的 Red Hat Ceph Storage 公共网络，检索 NIC 的 IP。将 **10.0.40.0** 替换为您在 ceph 公共网络中定义的子网后，执行以下命令。

```
$ ip a | grep 10.0.40
```

输出示例：

```
10.0.40.78
```

4. 以 **root** 用户身份在将作为集群中初始 monitor 节点的节点运行 **Cephadm** bootstrap 命令。**IP_ADDRESS** 选项是您用于运行 **cephadm bootstrap** 命令的节点 IP 地址。

注意

如果您配置了不同的用户而不是 **root** 进行免密码 SSH 访问，则使用带有 **cepadm bootstrap** 命令的 **--ssh-user=** 标志。

如果您使用非 default/id_rsa ssh 密钥名称，请使用 **--ssh-private-key** 和 **--ssh-public-key** 选项及 **cephadm** 命令。

```
$ cephadm  bootstrap --ssh-user=deployment-user --mon-ip 10.0.40.78 --apply-spec
/root/cluster-spec.yaml --registry-json /root/registry.json
```

重要

如果本地节点使用完全限定域名 (FQDN)，则将 **--allow-fqdn-hostname** 选项添加到命令行上的 **cephadm bootstrap**。

bootstrap 完成后，您将看到来自之前 cephadm bootstrap 命令的以下输出：

```
You can access the Ceph CLI with:

 sudo /usr/sbin/cephadm shell --fsid dd77f050-9afe-11ec-a56c-029f8148ea14 -c
/etc/ceph/ceph.conf -k /etc/ceph/ceph.client.admin.keyring

Consider enabling telemetry to help improve Ceph:

 ceph telemetry on

For more information see:

 https://docs.ceph.com/docs/pacific/mgr/telemetry/
```

5. 使用 ceph1 中的 ceph cli 客户端，验证 Red Hat Ceph Storage 集群部署的状态：

```
$ ceph -s
```

输出示例：

```
cluster:
  id:     3a801754-e01f-11ec-b7ab-005056838602
  health: HEALTH_OK

services:
  mon: 5 daemons, quorum ceph1,ceph2,ceph4,ceph5,ceph7 (age 4m)
  mgr: ceph1.khuuot(active, since 5m), standbys: ceph4.zotfsp
  osd: 12 osds: 12 up (since 3m), 12 in (since 4m)
  rgw: 2 daemons active (2 hosts, 1 zones)

data:
  pools:   5 pools, 107 pgs
  objects: 191 objects, 5.3 KiB
  usage:   105 MiB used, 600 GiB / 600 GiB avail
           105 active+clean
```

注意

启动所有服务可能需要几分钟时间。

在您未配置任何 OSD 时，获取全局恢复事件是正常的。

您可以使用 **ceph orch ps** 和 **ceph orch ls** 来进一步检查服务的状态。

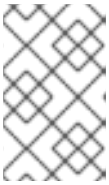
6. 验证所有节点是否是 **cephadm** 集群的一部分。

```
$ ceph orch host ls
```

输出示例：

```
HOST   ADDR         LABELS  STATUS
ceph1  10.0.40.78   _admin osd mon mgr
ceph2  10.0.40.35   osd mon
ceph3  10.0.40.24   osd mds rgw
```

```
ceph4  10.0.40.185   osd mon mgr
ceph5  10.0.40.88    osd mon
ceph6  10.0.40.66    osd mds rgw
ceph7  10.0.40.221    mon
```



注意

您可以直接从主机运行 Ceph 命令，因为 **ceph1** 在 **cephadm-ansible** 清单中配置，作为 [admin] 组的一部分。Ceph 管理密钥在 **cephadm bootstrap** 过程中复制到主机。

7. 检查数据中心的 Ceph 监控服务的当前位置。

```
$ ceph orch ps | grep mon | awk '{print $1 " " $2}'
```

输出示例：

```
mon.ceph1 ceph1
mon.ceph2 ceph2
mon.ceph4 ceph4
mon.ceph5 ceph5
mon.ceph7 ceph7
```

8. 检查数据中心的 Ceph 管理器服务的当前位置。

```
$ ceph orch ps | grep mgr | awk '{print $1 " " $2}'
```

输出示例：

```
mgr.ceph2.ycgwyz ceph2
mgr.ceph5.kremtt ceph5
```

9. 检查 ceph osd crush map 布局，以确保每个主机都配置了 OSD，其状态为 **UP**。此外，再次检查每个节点在表 3.1 中指定的右侧数据中心 bucket 下。

```
$ ceph osd tree
```

输出示例：

```
ID   CLASS  WEIGHT   TYPE NAME          STATUS  REWEIGHT  PRI-AFF
-1         0.87900  root default
-16        0.43950      datacenter DC1
-11        0.14650          host ceph1
  2   ssd  0.14650              osd.2       up   1.00000  1.00000
 -3        0.14650          host ceph2
  3   ssd  0.14650              osd.3       up   1.00000  1.00000
-13        0.14650          host ceph3
  4   ssd  0.14650              osd.4       up   1.00000  1.00000
-17        0.43950      datacenter DC2
 -5        0.14650          host ceph4
  0   ssd  0.14650              osd.0       up   1.00000  1.00000
 -9        0.14650          host ceph5
```

```
 1   ssd  0.14650          osd.1      up   1.00000  1.00000
-7       0.14650        host ceph6
 5   ssd  0.14650          osd.5      up   1.00000  1.00000
```

10. 创建并启用新的 RDB 块池。

```
$ ceph osd pool create 32 32
$ ceph osd pool application enable rbdpool rbd
```

注意

命令末尾的数字 32 是分配给这个池的 PG 数量。PG 数量可能会因集群中的 OSD 数量、使用池的预期%的不同而有所不同。您可以使用以下计算器来确定所需的 PG 数量：Ceph Placement Groups(PG)per Pool Calculator。

11. 验证 RBD 池已创建好。

```
$ ceph osd lspools | grep rbdpool
```

输出示例：

```
 3 rbdpool
```

12. 验证 MDS 服务是否处于活动状态，并且每个数据中心上有一个服务。

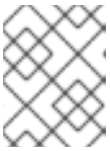
```
$ ceph orch ps | grep mds
```

输出示例：

```
mds.cephfs.ceph3.cjpbqo    ceph3          running (17m)   117s ago  17m    16.1M      -
16.2.9
mds.cephfs.ceph6.lqmgqt    ceph6          running (17m)   117s ago  17m    16.1M      -
16.2.9
```

13. 创建 CephFS 卷。

```
$ ceph fs volume create cephfs
```



注意

ceph fs volume create 命令还会创建所需的数据和 meta CephFS 池。有关更多信息，请参阅配置和挂载 Ceph 文件系统。

14. 检查 **Ceph** 状态，以验证 MDS 守护进程的部署方式。确保状态为 active，其中 **ceph6** 是这个文件系统的主 MDS，**ceph3** 是次 MDS。

```
$ ceph fs status
```

输出示例：

```
cephfs - 0 clients
```

```
======
RANK  STATE          MDS            ACTIVITY     DNS    INOS   DIRS   CAPS
 0    active  cephfs.ceph6.ggjywj  Reqs:   0 /s    10     13     12     0
       POOL          TYPE     USED  AVAIL
cephfs.cephfs.meta  metadata  96.0k   284G
cephfs.cephfs.data    data       0    284G
   STANDBY MDS
cephfs.ceph3.ogcqkl
```

15. 验证 RGW 服务是否处于活动状态。

```
$ ceph orch ps | grep rgw
```

输出示例：

```
rgw.objectgw.ceph3.kkmxgb  ceph3  *:8080       running (7m)      3m ago   7m    52.7M        -
16.2.9
rgw.objectgw.ceph6.xmnpah  ceph6  *:8080       running (7m)      3m ago   7m    53.3M        -
16.2.9
```

3.5.3. 配置 Red Hat Ceph Storage 扩展模式

使用 **cephadm** 完全部署了红帽 Ceph 存储集群后，请使用以下步骤来配置扩展集群模式。新的扩展模式旨在处理 2 个站点的情况。

步骤

1. 使用 ceph mon dump 命令检查监视器所使用的当前选择策略。默认情况下，在 ceph 集群中，连接设置为经典。

```
ceph mon dump | grep election_strategy
```

输出示例：

```
dumped monmap epoch 9
election_strategy: 1
```

2. 将 monitor 选举更改为连接性。

```
ceph mon set election_strategy connectivity
```

3. 再次运行前面的 ceph mon dump 命令，以验证 election_strategy 值。

```
$ ceph mon dump | grep election_strategy
```

输出示例：

```
dumped monmap epoch 10
election_strategy: 3
```

要了解有关不同选择策略的更多信息，请参阅配置监控选择策略。

4. 设置所有 Ceph 监视器的位置：

```
ceph mon set_location ceph1 datacenter=DC1
ceph mon set_location ceph2 datacenter=DC1
ceph mon set_location ceph4 datacenter=DC2
ceph mon set_location ceph5 datacenter=DC2
ceph mon set_location ceph7 datacenter=DC3
```

5. 验证每个监控器是否具有正确的位置。

```
$ ceph mon dump
```

输出示例：

```
epoch 17
fsid dd77f050-9afe-11ec-a56c-029f8148ea14
last_changed 2022-03-04T07:17:26.913330+0000
created 2022-03-03T14:33:22.957190+0000
min_mon_release 16 (pacific)
election_strategy: 3
0: [v2:10.0.143.78:3300/0,v1:10.0.143.78:6789/0] mon.ceph1; crush_location
{datacenter=DC1}
1: [v2:10.0.155.185:3300/0,v1:10.0.155.185:6789/0] mon.ceph4; crush_location
{datacenter=DC2}
2: [v2:10.0.139.88:3300/0,v1:10.0.139.88:6789/0] mon.ceph5; crush_location
{datacenter=DC2}
3: [v2:10.0.150.221:3300/0,v1:10.0.150.221:6789/0] mon.ceph7; crush_location
{datacenter=DC3}
4: [v2:10.0.155.35:3300/0,v1:10.0.155.35:6789/0] mon.ceph2; crush_location
{datacenter=DC1}
```

6. 通过安装 **ceph-base** RPM 软件包来创建使用此 OSD 拓扑的 CRUSH 规则，以便使用 **crushtool** 命令：

```
$ dnf -y install ceph-base
```

要了解有关 CRUSH 规则集的更多信息，请参阅 Ceph CRUSH 规则集。

7. 从集群获取编译的 CRUSH map：

```
$ ceph osd getcrushmap > /etc/ceph/crushmap.bin
```

8. 解译 CRUSH map，并将其转换为文本文件，以便能编辑它：

```
$ crushtool -d /etc/ceph/crushmap.bin -o /etc/ceph/crushmap.txt
```

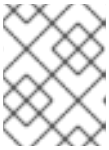
9. 编辑文件 **/etc/ceph/crushmap.txt**，将以下规则添加到 CRUSH map。

```
$ vim /etc/ceph/crushmap.txt
```

```
rule stretch_rule {
        id 1
        type replicated
```

```
        min_size 1
        max_size 10
        step take default
        step choose firstn 0 type datacenter
        step chooseleaf firstn 2 type host
        step emit
}
# end crush map
```

本例适用于所有 OpenShift Container Platform 集群中的活动应用程序。



注意

规则 **ID** 必须是唯一的。在示例中，我们只有一个带有 id 0 的 crush 规则，因此我们正在使用 id 1。如果您的部署创建了更多规则，则使用下一个可用 ID。

声明的 CRUSH 规则包含以下信息：

- **运行名称**
  - Description: 用于标识规则的唯一名称。
  - Value: **stretch_rule**
- **id**
  - Description：用于标识规则的唯一整数。
  - Value: **1**
- **type**
  - Description：描述存储驱动器复制或纠删代码的规则。
  - Value: **replicated**
- **min_size**
  - Description: 如果池制作的副本数少于这个数字，CRUSH 不会选择这一规则。
  - 值 ：1
- **max_size**
  - Description: 如果池制作的副本数多于这个数字，CRUSH 不会选择这一规则。
  - 值 ：10
- **步骤需要默认**
  - 描述：获取名为 **default** 的根存储桶，并开始迭代树。
- **步骤选择第 0 类型数据中心**
  - Description：选择数据中心存储桶，并放入它的子树中。
- **step chooseleaf firstn 2 type host**

- Description：选择给定类型的存储桶数量。在本例中，两个不同的主机位于其在上一级别输入的数据中心。

- **step emit**
  - Description: 输出当前的值并清除堆栈。通常在规则末尾使用，但也可用于从同一规则的不同树中选取。

10. 从文件 **/etc/ceph/crushmap.txt** 中编译新的 CRUSH map，并将其转换为名为 **/etc/ceph/crushmap2.bin** 的二进制文件：

```
$ crushtool -c /etc/ceph/crushmap.txt -o /etc/ceph/crushmap2.bin
```

11. 注入我们创建回集群的新 crushmap：

```
$ ceph osd setcrushmap -i /etc/ceph/crushmap2.bin
```

输出示例：

```
17
```

注意

数字 17 是一个计数器，它将根据您对 crush 映射所做的更改来增加（18,19 等）。

12. 验证创建的扩展规则现已可供使用。

```
ceph osd crush rule ls
```

输出示例：

```
replicated_rule
stretch_rule
```

13. 启用扩展群集模式。

```
$ ceph mon enable_stretch_mode ceph7 stretch_rule datacenter
```

在本例中，**ceph7** 是仲裁节点，**stretch_rule** 是在上一步中创建的 crush 规则，**datacenter** 是分开的存储桶。

14. 验证我们的所有池正在使用我们在 Ceph 集群中创建的 **stretch_rule** CRUSH 规则：

```
$ for pool in $(rados lspools);do echo -n "Pool: ${pool}; ";ceph osd pool get ${pool}
crush_rule;done
```

输出示例：

```
Pool: device_health_metrics; crush_rule: stretch_rule
Pool: cephfs.cephfs.meta; crush_rule: stretch_rule
Pool: cephfs.cephfs.data; crush_rule: stretch_rule
Pool: .rgw.root; crush_rule: stretch_rule
Pool: default.rgw.log; crush_rule: stretch_rule
```

```
Pool: default.rgw.control; crush_rule: stretch_rule
Pool: default.rgw.meta; crush_rule: stretch_rule
Pool: rbdpool; crush_rule: stretch_rule
```

这表明正在运行的红帽 Ceph 存储扩展群集，现在具有仲裁模式。

3.6. 在受管集群中安装 OPENSHIFT DATA FOUNDATION

要在两个 OpenShift Container Platform 集群之间配置存储复制，OpenShift Data Foundation Operator 必须首先安装在每个受管集群上，如下所示：

先决条件

- 确保您已满足 OpenShift Data Foundation 外部部署的硬件要求。有关硬件要求的详情，请参阅外部模式要求。

流程

1. 在每个受管集群中，安装和配置最新的 **OpenShift Data Foundation** 集群。
2. 在安装 operator 后，创建一个 StorageSystem，使用选择 **Full deployment** 类型和 **Connect with external storage platform**，其中您的**后端存储类型**为 **Red Hat Ceph Storage**。
   具体步骤请参考 以外部模式部署 OpenShift Data Foundation。

   将以下标记与 **ceph-external-cluster-details-exporter.py** 脚本一起使用。

   a. 您在 **ceph-external-cluster-details-exporter.py script** 中需要最少使用以下三个标记：

   --rbd-data-pool-name

   使用在为 OpenShift Container Platform 部署 RHCS 时创建的 RBD 池的名称。例如，池的名称可能为 **rbdpool**。

   --rgw-endpoint

   以 **<ip_address>:<port>** 格式提供端点。它是与您要配置的 OpenShift Container Platform 集群相同的站点中运行的 RGW 守护进程的 RGW IP。

   --run-as-user

   每个站点使用不同的客户端名称。

   b. 如果在 RHCS 部署过程中使用了默认值，则以下标记是**可选的** ：

   --cephfs-filesystem-name

   使用在为 OpenShift Container Platform 在 RHCS 部署期间创建的 CephFS 文件系统的名称，默认的文件系统名称是 **cephfs**。

   --cephfs-data-pool-name

   在用于 OpenShift Container Platform 的 RHCS 部署期间创建的 CephFS 数据池名称后，默认的池称为 **cephfs.data**。

   --cephfs-metadata-pool-name

   在用于 OpenShift Container Platform 的 RHCS 部署期间创建的 CephFS 元数据池的名称后，默认的池名为 **cephfs.meta**。

   c. 在 bootstrap 节点（**ceph1**）上运行以下命令，为 datacenter1 和 datacenter2 中的 RGW 端点获取 IP：

```
ceph orch ps | grep rgw.objectgw
```

输出示例：

```
rgw.objectgw.ceph3.mecpzm  ceph3  *:8080       running (5d)     31s ago   7w     204M
-  16.2.7-112.el8cp
rgw.objectgw.ceph6.mecpzm  ceph6  *:8080       running (5d)     31s ago   7w     204M
-  16.2.7-112.el8cp
```

```
host ceph3.example.com
host ceph6.example.com
```

输出示例：

```
ceph3.example.com has address 10.0.40.24
ceph6.example.com has address 10.0.40.66
```

d. 使用在 bootstrapped 节点 **ceph1** 上为第一个 OpenShift Container Platform 受管集群 **cluster1** 配置的参数，运行 **ceph-external-cluster-details-exporter.py**。

```
python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbdpool --cephfs-
filesystem-name cephfs --cephfs-data-pool-name cephfs.cephfs.data  --cephfs-metadata-
pool-name cephfs.cephfs.meta --<rgw-endpoint> XXX.XXX.XXX.XXX:8080 --run-as-user
client.odf.cluster1 > ocp-cluster1.json
```

注意

根据您的环境修改 <rgw-endpoint> XXX.XXX.XXX.XXX。

e. 使用在 bootstrapped 节点 **ceph1** 上为第一个 OpenShift Container Platform 受管集群 **cluster2** 配置的参数，运行 **ceph-external-cluster-details-exporter.py**。

```
python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbdpool --cephfs-
filesystem-name cephfs --cephfs-data-pool-name cephfs.cephfs.data  --cephfs-metadata-
pool-name cephfs.cephfs.meta --rgw-endpoint XXX.XXX.XXX.XXX:8080 --run-as-user
client.odf.cluster2 > ocp-cluster2.json
```

注意

根据您的环境修改 <rgw-endpoint> XXX.XXX.XXX.XXX。

- 将 bootstrap 集群 (ceph1) **ocp-cluster1.json** 和 **ocp-cluster2.json** 这两个文件保存到本地机器中。
- 在部署外部 OpenShift Data Foundation 的 **cluster1** 上，使用 OpenShift Container Platform 控制台上的文件 **ocp-cluster1.json** 的内容。
- 在部署外部 OpenShift Data Foundation 的 **cluster2** 上，使用 OpenShift Container Platform 控制台上的文件 **ocp-cluster2.json** 的内容。

3. 检查设置，然后选择 **Create StorageSystem**。

4. 使用以下命令验证每个受管集群中 OpenShift Data Foundation 部署是否成功：

```
$ oc get storagecluster -n openshift-storage ocs-external-storagecluster -o
jsonpath='{.status.phase}{"\n"}'
```

对于 Multicloud 网关(MCG)：

```
$ oc get noobaa -n openshift-storage noobaa -o jsonpath='{.status.phase}{"\n"}'
```

等待在**主受管集群**和**从受管集群上**的查询的状态结果变为 Ready。

5. 在 OpenShift Web 控制台中，进入到 **Installed Operators → OpenShift Data Foundation → Storage System → ocs-external-storagecluster-storagesystem → Resources**。验证 **StorageCluster** 的 **Status** 是否为 **Ready**，其旁边有一个绿色勾号标记。

3.7. 安装 OPENSHIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multicluster Orchestrator 是一个控制器，从 Hub 集群上的 OpenShift Container Platform OperatorHub 安装。

流程

1. 在 **Hub 集群中**，进入到 **OperatorHub** 并使用关键字过滤器搜索 **ODF Multicluster Orchestrator**。
2. 点 **ODF Multicluster Orchestrator** 标题。
3. 保留所有默认设置，然后点**安装**。
   确保 operator 资源安装在 **openshift-operators** 中，并可用于所有命名空间。

注意

ODF Multicluster Orchestrator 还会在 RHACM hub 集群上作为依赖项安装 **Openshift DR Hub Operator**。

4. 验证 Operator **Pod** 处于 **Running** 状态。**OpenShift DR Hub operator** 也安装在 **openshift-operators** 命名空间中。

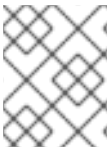
```
$ oc get pods -n openshift-operators
```

输出示例：

```
NAME                                  READY   STATUS      RESTARTS    AGE
odf-multicluster-console-6845b795b9-blxrn   1/1     Running     0          4d20h
odfmo-controller-manager-f9d9dfb59-jbrsd    1/1     Running     0          4d20h
ramen-hub-operator-6fb887f885-fss4w         2/2     Running     0          4d20h
```

3.8. 在集群间配置 SSL 访问

在 **primary 和 secondary 集群**间配置网络 SSL 访问，因此元数据可以以一个安全的方式保持在不同的集群中的 Multicloud Gateway (MCG) **对象存储桶**，以及保持在 **Hub 集群**中验证对对象存储桶的访问。



注意

如果所有 OpenShift 集群都为您的环境使用签名的有效证书集进行部署，则可以跳过本节。

步骤

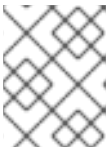
1. 提取主受管集群的入口证书，并将输出保存到 **primary.crt**。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > primary.crt
```

2. 提取二级受管集群的入口证书，并将输出保存到 **secondary.crt**。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > secondary.crt
```

3. 创建新的 ConfigMap 文件，以使用文件名 **cm-clusters-crt.yaml** 来保存远程集群的证书捆绑包。



注意

如本示例文件所示，每个集群可能有超过三个证书。另外，请确保在从之前创建的 **primary.crt** 和 **secondary.crt** 文件中复制并粘贴后，证书内容会被正确缩进。

```
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 primary.crt here>
    -----END CERTIFICATE----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 from secondary.crt here>
    -----END CERTIFICATE-----
kind: ConfigMap
```

```
metadata:
  name: user-ca-bundle
  namespace: openshift-config
```

4. 在 **Primary 受管集群**, **Secondary 受管集群**, 和 **Hub 集群** 中创建 **ConfigMap**。

```
$ oc create -f cm-clusters-crt.yaml
```

输出示例：

```
configmap/user-ca-bundle created
```

5. 对**主受管集群**、**二级受管集群**和 **Hub 集群**上的默认代理资源进行补丁。

```
$ oc patch proxy cluster --type=merge  --patch='{"spec":{"trustedCA":{"name":"user-ca-bundle"}}}'
```

输出示例：

```
proxy.config.openshift.io/cluster patched
```

3.9. 在 HUB 集群上创建灾难恢复策略

OpenShift 灾难恢复策略(DRPolicy)资源指定参与灾难恢复解决方案和所需的复制间隔的 OpenShift Container Platform 集群。DRPolicy 是一个集群范围的资源，用户可以应用到需要灾难恢复解决方案的应用程序。

ODF MultiCluster Orchestrator Operator 通过 **Multicluster Web console** 促进每个 **DRPolicy** 和相应的 **DRClusters** 的创建。

先决条件

- 确保至少一组两个受管集群。

流程

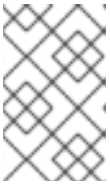
1. 在 **OpenShift 控制台**中，进入到 **All Clusters → Data Services → Data policies**。
2. 点 **Create DRPolicy**。
3. 输入 **Policy name**。确保每个 DRPolicy 都有一个唯一名称（例如： **ocp4perf1-ocp4perf2**）。
4. 从与此新策略关联的受管集群列表中选择两个集群。
5. **复制策略**根据所选的 OpenShift 集群自动设置为 **sync**。
6. 点 **Create**。
7. 验证 **DRPolicy** 是否已成功创建。在 **Hub 集群**中，为创建的每个 DRPolicy 资源运行这个命令，使用您的唯一名称替换 *<drpolicy_name>*。

```
$ oc get drpolicy <drpolicy_name> -o jsonpath='{.status.conditions[].reason}{"\n"}'
```

输出示例：

```
Succeeded
```

创建 DRPolicy 时，也会创建两个 DRCluster 资源。验证所有三个资源并且状态显示为 **Succeeded** 最多需要 10 分钟。



注意

DRPolicy 不支持编辑 **SchedulingInterval**、**ReplicationClassSelector**、**VolumeSnapshotClassSelector** 和 **DRClusters** 字段值。

8. 验证对象存储桶可以从 **Hub 集群**、**Primary 受管集群**和 **Secondary 受管集群**访问。

   a. 获取 Hub 集群上的 **DRClusters** 的名称。

   ```
   $ oc get drclusters
   ```

   输出示例：

   b. 检查 S3 访问每个受管集群上创建的每个存储桶。使用 DRCluster 验证命令，使用您的唯一名称替换 *<drcluster_name>*。

   注意

   DRClusters 不支持编辑 **Region** 和 **S3ProfileName** 字段值。

   ```
   $ oc get drcluster <drcluster_name> -o jsonpath='{.status.conditions[2].reason}{"\n"}'
   ```

   输出示例：

   ```
   Succeeded
   ```

   注意

   确保为在 **Hub 集群**中的两个 **DRClusters** 运行命令。

9. 验证 **OpenShift DR Cluster operator** 已成功在 **Primary 受管集群**和 **Secondary 受管集群**上成功安装。

   ```
   $ oc get csv,pod -n openshift-dr-system
   ```

   输出示例：

   ```
   NAME                                                                 DISPLAY                  VERSION
   REPLACES   PHASE
   clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.15.0        Openshift DR
   Cluster Operator   4.15.0                  Succeeded
   clusterserviceversion.operators.coreos.com/volsync-product.v0.8.0              VolSync
   0.8.0                   Succeeded
   ```

```
NAME                                      READY   STATUS    RESTARTS   AGE
pod/ramen-dr-cluster-operator-6467cf5d4c-cc8kz   2/2     Running   0          3d12h
```

您还可以验证 **OpenShift DR Cluster Operator** 是否在每个受管集群的 **OperatorHub** 上成功安装。

10. 验证 secret 是否在主受管集群和次受管集群中正确传播。

```
oc get secrets -n openshift-dr-system | grep Opaque
```

将输出与 Hub 集群的 s3SecretRef 匹配：

```
oc get cm -n openshift-operators ramen-hub-operator-config -oyaml
```

3.10. 为隔离自动化配置 DRCLUSTERS

在应用程序故障切换前启用隔离需要此配置。为了防止在集群中写入持久性卷（由灾难达到的，OpenShift DR 指示 Red Hat Ceph Storage (RHCS)指示 Red Hat Ceph Storage (RHCS)从 RHCS 外部存储隔离节点。本节介绍了如何为 DRCluster 节点添加 IP 或 IP 地址范围。

3.10.1. 将节点 IP 地址添加到 DRClusters

1. 通过在 **Primary 受管集群** 和 **secondary 受管集群**中运行此命令，在受管集群中查找所有 OpenShift 节点的 **IP 地址**。

```
$ oc get nodes -o jsonpath='{range .items[*]}{.status.addresses[?
(@.type=="ExternalIP")].address}{"\n"}{end}'
```

输出示例：

```
10.70.56.118
10.70.56.193
10.70.56.154
10.70.56.242
10.70.56.136
10.70.56.99
```

获得 **IP 地址**后，可以为每个受管集群修改 **DRCluster** 资源。

2. 在 Hub 集群上查找 **DRCluster** 名称。

```
$ oc get drcluster
```

输出示例：

```
NAME       AGE
ocp4perf1   5m35s
ocp4perf2   5m35s
```

3. 编辑每个 **DRCluster**，在使用您的唯一名称替换 **<drcluster_name>** 后添加您的唯一的 IP地址。

```
$ oc edit drcluster <drcluster_name>
```

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRCluster
metadata:
[...]
spec:
  s3ProfileName: s3profile-<drcluster_name>-ocs-external-storagecluster
  ## Add this section
  cidrs:
    - <IP_Address1>/32
    - <IP_Address2>/32
    - <IP_Address3>/32
    - <IP_Address4>/32
    - <IP_Address5>/32
    - <IP_Address6>/32
[...]
```

输出示例：

```
drcluster.ramendr.openshift.io/ocp4perf1 edited
```

注意

有超过 6 个 IP 地址。

在对等 **DRCluster** 资源（如 ocp4perf2）中，为 **Secondary 受管集群**上的 **IP 地址** 修改此 DRCluster 配置。

3.10.2. 在 DRClusters 中添加隔离注解

在所有 DRCluster 资源中添加以下注解。这些注解包括以下说明（测试应用程序故障切换）创建的 **NetworkFence** 资源所需的详细信息。

注意

将 *<drcluster_name>* 替换为您的唯一名称。

```
$ oc edit drcluster <drcluster_name>
```

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRCluster
metadata:
  ## Add this section
  annotations:
    drcluster.ramendr.openshift.io/storage-clusterid: openshift-storage
    drcluster.ramendr.openshift.io/storage-driver: openshift-storage.rbd.csi.ceph.com
    drcluster.ramendr.openshift.io/storage-secret-name: rook-csi-rbd-provisioner
    drcluster.ramendr.openshift.io/storage-secret-namespace: openshift-storage
[...]
```

输出示例：

```
drcluster.ramendr.openshift.io/ocp4perf1 edited
```

确保为 **DRCluster** 资源添加这些注解（例如：**ocp4perf1** 和 **ocp4perf2**）。

3.11. 为测试灾难恢复解决方案创建示例应用程序

OpenShift Data Foundation 灾难恢复 (DR) 解决方案支持针对由 RHACM 管理的基于 Subscription 和 ApplicationSet 的应用程序的灾难恢复。如需了解更多详细信息，请参阅 订阅和 ApplicationSet 文档。

以下小节详细介绍了如何创建应用程序并将 DRPolicy 应用到应用程序。

- 基于订阅的应用程序
  没有 cluster-admin 权限的 OpenShift 用户，请参阅知识库文章 如何为应用程序用户分配必要的权限来执行灾难恢复操作。
- 基于 ApplicationSet 的应用程序
  没有 cluster-admin 权限的 OpenShift 用户无法创建基于 ApplicationSet 的应用程序。

3.11.1. 基于订阅的应用程序

3.11.1.1. 创建基于订阅的应用程序示例

为了测试从 **Primary 受管集群**到 **Secondary 受管集群**的 **failover** 和 **relocate**，我们需要一个简单的应用程序。

先决条件

- 在创建用于常规消耗的应用程序时，请确保应用程序仅部署到一个集群。
- 使用名为 **busybox** 的示例应用作为示例。
- 当应用程序通过或重新定位失败时，确保使用全局流量管理器(GTM)或 Global Server Load Balancing (GLSB)服务配置应用程序的所有外部路由。
- 作为最佳实践，将属于 Red Hat Advanced Cluster Management (RHACM) 订阅的 Red Hat Advanced Cluster Management (RHACM) 订阅分组在一起，以指代单个 Placement Rule 到 DR 保护它们作为组。进一步将它们创建为一个应用程序，用于订阅的逻辑分组，以备将来 DR 操作，如故障转移和重新定位。

注意

如果不相关的订阅引用与放置操作相同的放置规则，它们也会受到 DR 保护，因为 DR 工作流控制引用放置规则的所有订阅。

流程

1. 在 Hub 集群中，进入到 **Applications**，再点 **Create application**。
2. 选择类型 **Subscription**。
3. 输入应用程序 **Name** （如 **busybox**）和 **Namespace** （如 **busybox-sample**）。
4. 在 Repository location for resources 部分中，选择 **Repository type Git**。

5. 输入示例应用程序的 Git 存储库 URL、github **Branch** 和 **Path**，在其中创建资源 **busybox** Pod 和 PVC。
   使用示例应用程序存储库作为 **https://github.com/red-hat-storage/ocm-ramen-samples**，其中 **Branch** 是 **release-4.15**，**Path** 是 **busybox-odr-metro**。

6. 向下滚动，直到您看到 **Deploy application resources on cluster with all specified labels**。
   - 选择全局 **Cluster set** 或包含环境正确受管集群的全局集群。
   - 添加一个 *<name>* 标签，将其值设置为**受管集群**的名称。

7. 点位于右上角的 **Create**。
   在接下来的屏幕中，进入 **Topology** 选项卡。您可以看到应用程序拓扑上有所有的绿色对勾图标。

注意

要获取更多信息，请点击任何拓扑元素，拓扑视图右侧会显示一个窗口。

8. 验证示例应用程序部署。
   现在，**busybox** 应用程序已部署到首选集群，现在可以验证部署。

   登录到您的受管集群，其中 **busybox** 由 RHACM 部署。

```
$ oc get pods,pvc -n busybox-sample
```

   输出示例：

```
NAME                     READY   STATUS    RESTARTS   AGE
pod/busybox-67bf494b9-zl5tr   1/1     Running   0          77s


NAME                                 STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS                  AGE
persistentvolumeclaim/busybox-pvc    Bound    pvc-c732e5fe-daaf-4c4d-99dd-462e04c18412
5Gi        RWO            ocs-storagecluster-ceph-rbd   77s
```

3.11.1.2. 将数据策略应用到示例应用程序

先决条件

- 确保 Data 策略中引用的两个受管集群都可以访问。如果没有，在两个集群都在线前，应用程序不会受到灾难恢复的影响。

流程

1. 在 Hub 集群中，进入到 **All Clusters** → **Applications**。
2. 点应用程序末尾的 Actions 菜单，以查看可用操作的列表。
3. 点 **Manage data policy**→ **Assign data policy**。
4. 选择 **Policy**，再单击 **Next**。

5. 选择 **Application resource**，然后使用 **PVC 标签选择器** 为所选应用程序资源选择 **PVC 标签**。

注意

您可以为所选应用程序资源选择多个 PVC 标签。您还可以使用 **Add application resource** 选项添加多个资源。

6. 添加所有应用程序资源后，点 **Next**。

7. 检查 **Policy 配置详情**并点 **Assign**。新分配的 Data 策略显示在 **Manage data policy** modal 列表视图中。

8. 验证您可以在 Applications 页面中查看分配的策略详情。

   a. 在 Applications 页面上，进入到 **Data policy** 列，再单击 **策略链接** 以展开视图。

   b. 验证您可以看到分配的策略数量以及故障转移和重新定位状态。

   c. 单击 **View more details**，以查看与应用程序一起使用的策略持续活动的状态。

9. 将 DRPolicy 应用到应用程序后，在 drpc yaml 输出中确认 **ClusterDataProtected** 是否已设置为 **True**。

3.11.2. 基于 ApplicationSet 的应用程序

3.11.2.1. 创建基于 ApplicationSet 的应用程序

前提条件

- 确保 Hub 集群上安装了 Red Hat OpenShift GitOps Operator。具体步骤请查看 RHACM 文档。
- 确保主和从受管集群都注册到 GitOps。有关注册说明，请参阅 将受管集群注册到 GitOps。然后，检查 **GitOpsCluster** 资源使用的放置是否用于注册两个受管集群，具有处理集群的容限。您可以使用 **oc get placement <placement-name> -n openshift-gitops -o yaml** 命令来验证以下容限是否已添加到放置中。

```
tolerations:
- key: cluster.open-cluster-management.io/unreachable
  operator: Exists
- key: cluster.open-cluster-management.io/unavailable
  operator: Exists
```

如果没有添加容限，请参阅为 Red Hat Advanced Cluster Management 和 OpenShift GitOps 配置应用程序放置容限。

流程

1. 在 Hub 集群中，进入到 **All Clusters → Applications** 并点 **Create application**。
2. 选择应用程序类型作为 **Argo CD ApplicationSet - Push model**
3. 在常规步骤 1 中，输入您的**应用程序集名称**。
4. 选择 **Argo server openshift-gitops** 和 **Requeue time** 为 **180** 秒。

5. 点击 **Next**。
6. 在 Repository location for resources 部分中，选择 **Repository type Git**。
7. 输入示例应用程序的 Git 存储库 URL、github Branch 和 Path，在其中创建资源 busybox Pod 和 PVC。
   a. 使用示例应用程序存储库作为 https://github.com/red-hat-storage/ocm-ramen-samples
   b. 选择 **Revision** 作为 **release-4.15**
   c. 选择 Path 为 **busybox-odr-metro**。
8. 输入 **Remote namespace** 值。(例如 busybox-sample)，然后点 **Next**。
9. 选择 **Sync policy** 设置并点 **Next**。
   您可以选择一个或多个选项。
10. 添加一个 *<name>* 标签，将其值设置为**受管集群**的名称。
11. 点击 **Next**。
12. 检查设置详情并点 **Submit**。

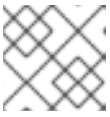
3.11.2.2. 将 Data policy 应用到基于 ApplicationSet 的应用程序示例

先决条件

- 确保 Data 策略中引用的两个受管集群都可以访问。如果没有，在两个集群都在线前，应用程序不会受到灾难恢复的影响。

流程

1. 在 Hub 集群中，进入到 **All Clusters → Applications**。
2. 点应用程序末尾的 Actions 菜单，以查看可用操作的列表。
3. 点 **Manage data policy→ Assign data policy**。
4. 选择 **Policy**，再单击 **Next**。
5. 选择 **Application resource**，然后使用 **PVC 标签选择器** 为所选应用程序资源选择 **PVC 标签**。



注意

您可以为所选应用程序资源选择多个 PVC 标签。

6. 添加所有应用程序资源后，点 **Next**。
7. 检查 **Policy 配置详情**并点 **Assign**。新分配的 Data 策略显示在 **Manage data policy** modal 列表视图中。
8. 验证您可以在 Applications 页面中查看分配的策略详情。
   a. 在 Applications 页面上，进入到 **Data policy** 列，再单击 **策略链接** 以展开视图。

b. 验证您可以看到分配的策略数量以及故障转移和重新定位状态。

9. 将 DRPolicy 应用到应用程序后，在 drpc yaml 输出中确认 **ClusterDataProtected** 是否已设置为 **True**。

3.11.3. 删除示例应用程序

本节提供了使用 RHACM 控制台删除示例应用程序 **busybox** 的说明。

重要

当删除 DR 保护的应用程序时，需要可以访问属于 DRPolicy 的这两个集群。这是为了确保根据删除 DR 保护的一部分清理相应 S3 存储中的所有受保护的 API 资源和资源。如果访问其中一个集群不是健康，删除应用程序的 **DRPlacementControl** 资源（在 hub 上）将处于 Deleting 状态。

先决条件

- 这些说明在故障转移和重新定位测试完成前不应执行示例应用程序，且应用程序已准备好从 RHACM 和受管集群中删除。

流程

1. 在 RHACM 控制台上，前往 **Applications**。
2. 搜索要删除的示例应用程序（例如 **busybox**）。
3. 点击您要删除的应用程序旁边的 Action Menu **(⋮)**。
4. 点击 **Delete application**。
   选择了 **Delete application** 时，将显示一个新屏幕，询问是否还应删除与应用相关的资源。
5. 选中 **Remove application 相关资源**复选框，以删除 Subscription 和 PlacementRule。
6. 单击 **Delete**。这将删除主受管集群（或应用程序所运行的任何群集）上的 busybox 应用程序。
7. 除了使用 RHACM 控制台删除的资源外，如果删除 **busybox** 应用后没有自动删除，则删除 **DRPlacementControl**。

   a. 登录到 Hub 集群的 OpenShift Web 控制台，再进入到 项目 **busybox-sample** 的 Installed Operators。
   对于 ApplicationSet 应用程序，将项目选为 **openshift-gitops**。

   b. 单击 **OpenShift DR Hub Operator**，然后单击 **DRPlacementControl** 选项卡。

   c. 点击您要删除的 **busybox** 应用程序 DRPlacementControl 旁边的 Action Menu **(⋮)**。

   d. 单击 **Delete DRPlacementControl**。

   e. 单击 **Delete**。

注意

此过程可用于使用 **DRPlacementControl** 资源删除任何应用。

3.12. 受管集群间基于订阅的应用程序故障切换

当受管集群因任何原因而不可用时，执行故障转移。这个故障转移方法基于应用程序。

先决条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为它可能需要一些时间才能更新。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行故障转移操作前，请检查两个受管集群的状态。
     但是，当您切换到的集群处于 *Ready* 状态时，故障转移操作仍然可以运行。

流程

1. 在 **Hub 集群**上启用隔离。
   a. 打开 CLI 终端并编辑 **DRCluster 资源**，其中 *<drcluster_name>* 是您的唯一名称。

   小心

   在隔离了受管集群后，**所有** 从应用程序到 OpenShift Data Foundation 外部存储集群的通信将失败，一些 **Pod** 都将处于不健康状态（例如：**CreateContainerError**,**CrashLoopBackOff**）现已被隔离的集群。

   ```
   $ oc edit drcluster <drcluster_name>
   ```

   ```
   apiVersion: ramendr.openshift.io/v1alpha1
   kind: DRCluster
   metadata:
   [...]
   spec:
     ## Add this line
     clusterFence: Fenced
     cidrs:
     [...]
   [...]
   ```

   输出示例：

   ```
   drcluster.ramendr.openshift.io/ocp4perf1 edited
   ```

   b. 为**主受管集群**验证 **Hub 集群**上的隔离状态，使用您的唯一标识符替换 <drcluster_name>。

   ```
   $ oc get drcluster.ramendr.openshift.io <drcluster_name> -o jsonpath='{.status.phase}
   {"\n"}'
   ```

   输出示例：

```
Fenced
```

c. 验证属于 OpenShift Container Platform 集群节点的 IP 现在是否在 blocklist 中。

```
$ ceph osd blocklist ls
```

输出示例

```
cidr:10.1.161.1:0/32 2028-10-30T22:30:03.585634+0000
cidr:10.1.161.14:0/32 2028-10-30T22:30:02.483561+0000
cidr:10.1.161.51:0/32 2028-10-30T22:30:01.272267+0000
cidr:10.1.161.63:0/32 2028-10-30T22:30:05.099655+0000
cidr:10.1.161.129:0/32 2028-10-30T22:29:58.335390+0000
cidr:10.1.161.130:0/32 2028-10-30T22:29:59.861518+0000
```

2. 在 Hub 集群中，进入到 **Applications**。
3. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
4. 点 **Failover application**。
5. 在显示了 **Failover 应用程序** 模态后，选择**策略**和**目标集群**，相关的应用程序将在出现灾难时故障转移。
6. 点**选择订阅组**下拉菜单，验证默认选择或修改此设置。
   默认情况下，选择为应用程序资源复制的订阅组。
7. 检查 **Failover readiness** 的状态。
   - 如果状态是 **Ready** 且带有一个绿色勾号，这表示目标集群已就绪，可启动故障转移。继续执行第 7 步。
   - 如果状态是 **Unknown** 或 **Not ready**，请等待到状态变为 **Ready**。
8. 点 **Initiate**。busybox 应用程序现在故障转移到 **Secondary-managed 集群**。
9. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
10. 验证活动状态是否为应用的 **FailedOver**。
    a. 进入 **Applications → Overview** 选项卡。
    b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
    c. 在 **Data policy** 弹出窗口中，点 **View more details** 链接。

3.13. 受管集群之间基于 APPLICATIONSET 的应用程序故障切换

当受管集群因任何原因而不可用时，执行故障转移。这个故障转移方法基于应用程序。

先决条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。

- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为它可能需要一些时间才能更新。

  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。

  2. 在执行故障转移操作前，请检查两个受管集群的状态。
     但是，当您切换到的集群处于 *Ready* 状态时，故障转移操作仍然可以运行。

流程

1. 在 **Hub 集群**上启用隔离。

   a. 打开 CLI 终端并编辑 **DRCluster 资源**，其中 *<drcluster_name>* 是您的唯一名称。

      小心

      在隔离了受管集群后，**所有** 从应用程序到 OpenShift Data Foundation 外部存储集群的通信将失败，一些 **Pod** 都将处于不健康状态（例如：**CreateContainerError**,**CrashLoopBackOff**）现已被隔离的集群。

      ```
      $ oc edit drcluster <drcluster_name>
      ```

      ```
      apiVersion: ramendr.openshift.io/v1alpha1
      kind: DRCluster
      metadata:
      [...]
      spec:
        ## Add this line
        clusterFence: Fenced
        cidrs:
        [...]
      [...]
      ```

      输出示例：

      ```
      drcluster.ramendr.openshift.io/ocp4perf1 edited
      ```

   b. 为**主受管集群**验证 **Hub 集群**上的隔离状态，使用您的唯一标识符替换 <drcluster_name>。

      ```
      $ oc get drcluster.ramendr.openshift.io <drcluster_name> -o jsonpath='{.status.phase}
      {"\n"}'
      ```

      输出示例：

      ```
      Fenced
      ```

   c. 验证属于 OpenShift Container Platform 集群节点的 IP 现在是否在 blocklist 中。

      ```
      $ ceph osd blocklist ls
      ```

      输出示例

      ```
      cidr:10.1.161.1:0/32 2028-10-30T22:30:03.585634+0000
      ```

```
cidr:10.1.161.14:0/32 2028-10-30T22:30:02.483561+0000
cidr:10.1.161.51:0/32 2028-10-30T22:30:01.272267+0000
cidr:10.1.161.63:0/32 2028-10-30T22:30:05.099655+0000
cidr:10.1.161.129:0/32 2028-10-30T22:29:58.335390+0000
cidr:10.1.161.130:0/32 2028-10-30T22:29:59.861518+0000
```

2. 在 Hub 集群中，进入到 **Applications**。
3. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
4. 点 **Failover application**。
5. 显示 **Failover 应用程序** 模态时，验证显示的详细信息是否正确并检查 Failover **readiness** 的状态。如果状态是 **Ready** 且带有绿色勾号，这表示目标集群已准备好故障转移启动。
6. 点 **Initiate**。busybox 资源现在在目标集群上创建。
7. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
8. 验证活动状态是否为应用的 **FailedOver**。
   a. 进入 **Applications → Overview** 选项卡。
   b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   c. 在 **Data policy** 弹出窗口中，验证您可以看到一个或多个策略名称以及与应用程序中使用的策略关联的持续活动。

3.14. 在受管集群间重新定位基于订阅的应用程序

当所有受管集群都可用时，将应用程序重新定位到首选位置。

前提条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为可能需要一些时间来更新。只有在主集群和首选集群启动并运行时才可执行重新定位。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行重新定位操作前，请单独检查两个受管集群的状态。
- 在取消隔离前，验证应用程序是否已从集群中清理。

流程

1. 在 Hub 集群中禁用隔离功能。
   a. 编辑此集群的 **DRCluster 资源**，将 <drcluster_name> 替换为唯一名称。

```
$ oc edit drcluster <drcluster_name>
```

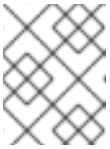
```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRCluster
```

```
metadata:
[...]
spec:
  cidrs:
  [...]
  ## Modify this line
  clusterFence: Unfenced
  [...]
[...]
```

输出示例：

```
drcluster.ramendr.openshift.io/ocp4perf1 edited
```

b. 正常重新引导了 **Fenced** 的 OpenShift Container Platform 节点。重启后，需要重启来恢复 I/O 操作，以避免进一步恢复编配失败。按照正常重新引导节点 的步骤，重新引导集群中的所有节点。



注意

先确保所有节点已被封锁和排空，然后再重新引导并在节点上执行 uncordon 操作。

c. 在所有 OpenShift 节点被重新引导并处于 **Ready** 状态后，通过在主受管集群中运行此命令（或任何集群是 Unfenced），验证所有 Pod 都处于健康状态。

```
oc get pods -A | egrep -v 'Running|Completed'
```

输出示例：

```
NAMESPACE                                  NAME
READY   STATUS      RESTARTS      AGE
```

此查询的输出应该为零个 Pod，然后继续下一步。

重要

如果因为严重的存储通信导致 pod 仍然处于不健康状态，请在继续操作前进行故障排除并解决。由于存储集群位于 OpenShift 的外部，因此在 OpenShift 应用正常运行的站点中断后，还必须正确恢复它。

另外，您可以使用 OpenShift Web 控制台仪表板和 Overview 选项卡来评估应用程序和外部 ODF 存储集群的健康状态。详细的 OpenShift Data Foundation 仪表板可通过 **Storage → Data Foundation** 找到。

d. 验证 **Unfenced** 集群是否处于健康状态。为 Primary-managed 集群验证 Hub 集群中的隔离状态，将 <drcluster_name> 替换为唯一名称。

```
$ oc get drcluster.ramendr.openshift.io <drcluster_name> -o jsonpath='{.status.phase}
{"\n"}'
```

输出示例：

```
Unfenced
```

e. 验证属于 OpenShift Container Platform 集群节点的 IP 不在 blocklist 中。

```
$ ceph osd blocklist ls
```

确保您没有看到隔离过程中添加的 IP。

2. 在 Hub 集群中，进入到 **Applications**。
3. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
4. 点 **Relocate application**。
5. 当 **Relocate application** 模态显示时，选择 **policy** 和 **target cluster**，在出现灾难时相关的应用程序将重新定位到其中。
6. 默认情况下，选择部署应用程序资源的订阅组。**点选择订阅组**下拉菜单，验证默认选择或修改此设置。
7. 检查 **Relocation readiness** 的状态。
   - 如果状态是 **Ready** 且带有一个绿色勾号，这表示目标集群已准备好重定位来进行启动。继续执行第 7 步。
   - 如果状态是 **Unknown** 或 **Not ready**，请等待到状态变为 **Ready**。
8. 点 **Initiate**。busybox 资源现在在目标集群上创建。
9. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
10. 验证应用程序的活动状态是否显示为 **Relocated**。
    a. 进入 **Applications → Overview** 选项卡。
    b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
    c. 在 **Data policy** 弹出窗口中，点 **View more details** 链接。

3.15. 在受管集群间重新定位基于 APPLICATIONSET 的应用程序

当所有受管集群都可用时，将应用程序重新定位到首选位置。

前提条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为可能需要一些时间来更新。只有在主集群和首选集群启动并运行时才可执行重新定位。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行重新定位操作前，请单独检查两个受管集群的状态。
- 在取消隔离前，验证应用程序是否已从集群中清理。

流程

1. 在 Hub 集群中禁用隔离功能。

   a. 编辑此集群的 **DRCluster 资源**，将 <drcluster_name> 替换为唯一名称。

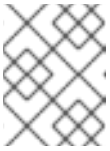
   ```
   $ oc edit drcluster <drcluster_name>
   ```

   ```
   apiVersion: ramendr.openshift.io/v1alpha1
   kind: DRCluster
   metadata:
   [...]
   spec:
     cidrs:
     [...]
     ## Modify this line
     clusterFence: Unfenced
     [...]
   [...]
   ```

   输出示例：

   ```
   drcluster.ramendr.openshift.io/ocp4perf1 edited
   ```

   b. 正常重新引导了 **Fenced** 的 OpenShift Container Platform 节点。重启后，需要重启来恢复 I/O 操作，以避免进一步恢复编配失败。按照正常重新引导节点 的步骤，重新引导集群中的所有节点。

   注意

   先确保所有节点已被封锁和排空，然后再重新引导并在节点上执行 uncordon 操作。

   c. 在所有 OpenShift 节点被重新引导并处于 **Ready** 状态后，通过在主受管集群中运行此命令（或任何集群是 Unfenced），验证所有 Pod 都处于健康状态。

   ```
   oc get pods -A | egrep -v 'Running|Completed'
   ```

   输出示例：

   ```
   NAMESPACE                                  NAME
   READY   STATUS      RESTARTS      AGE
   ```

   此查询的输出应该为零个 Pod，然后继续下一步。

   重要

   如果因为严重的存储通信导致 pod 仍然处于不健康状态，请在继续操作前进行故障排除并解决。由于存储集群位于 OpenShift 的外部，因此在 OpenShift 应用正常运行的站点中断后，还必须正确恢复它。

另外，您可以使用 OpenShift Web 控制台仪表板和 Overview 选项卡来评估应用程序和外部 ODF 存储集群的健康状态。详细的 OpenShift Data Foundation 仪表板可通过 **Storage → Data Foundation** 找到。

d. 验证 **Unfenced** 集群是否处于健康状态。为 Primary-managed 集群验证 Hub 集群中的隔离状态，将 <drcluster_name> 替换为唯一名称。

```
$ oc get drcluster.ramendr.openshift.io <drcluster_name> -o jsonpath='{.status.phase}
{"\n"}'
```

输出示例：

```
Unfenced
```

e. 验证属于 OpenShift Container Platform 集群节点的 IP 不在 blocklist 中。

```
$ ceph osd blocklist ls
```

确保您没有看到隔离过程中添加的 IP。

2. 在 Hub 集群中，进入到 **Applications**。
3. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
4. 点 **Relocate application**。
5. 当 **Relocate application** 模态显示时，选择 **policy** 和 **target cluster**，在出现灾难时相关的应用程序将重新定位到其中。
6. 点 **Initiate**。busybox 资源现在在目标集群上创建。
7. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
8. 验证应用程序的活动状态是否显示为 **Relocated**。
   a. 进入 **Applications → Overview** 选项卡。
   b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   c. 在 **Data policy** 弹出窗口中，验证您可以看到一个或多个策略名称以及与应用程序一起使用的策略关联的重新定位状态。

3.16. 使用 METRO-DR 恢复替换集群

当主集群出现问题时，您可以获得修复选项，等待现有集群恢复，或者如果集群不可取，请完全替换集群。此解决方案指导您在将失败的主集群替换为新集群时，并对这个新集群启用故障恢复（重复）。

在这些说明中，我们假设 RHACM 受管集群必须在安装和保护应用程序后被替换。在本小节中，RHACM 受管集群是 **替换集群**，而没有替换的集群是 **Surviving 集群**，新的集群是 **恢复集群**。

前提条件

- 确保 Metro-DR 环境已配置了使用 Red Hat Advance Cluster Management (RHACM)安装的应用程序。
- 确保为应用程序分配了一个数据策略，该策略可防止它们出现集群故障。

流程

1. 在 **Hub 集群中** 执行以下步骤：

   a. 使用 CLI 终端编辑 **DRCluster** 资源来隔离替换集群，其中 < *drcluster_name>* 是替换集群名称。

   ```
   oc edit drcluster <drcluster_name>
   ```

   ```
   apiVersion: ramendr.openshift.io/v1alpha1
   kind: DRCluster
   metadata:
   [...]
   spec:
     ## Add or modify this line
     clusterFence: Fenced
     cidrs:
     [...]
   [...]
   ```

   b. 使用 RHACM 控制台，进入到 **Applications**，并将所有受保护的应用程序从故障集群故障转移到存活的集群。

   c. 验证并确保所有受保护的应用程序现在都在存活的集群中运行。

   注意

   每个应用程序 DRPlacementControl 的 **PROGRESSION** 状态将显示为 **Cleaning Up**。如果替换集群离线或关闭，则这是预期的。

2. 取消隔离替换群集。
   使用 CLI 终端，编辑 DRCluster 资源，其中 <*drcluster_name*& gt; 是替换集群名称。

   ```
   $ oc edit drcluster <drcluster_name>
   ```

   ```
   apiVersion: ramendr.openshift.io/v1alpha1
   kind: DRCluster
   metadata:
   [...]
   spec:
     ## Modify this line
     clusterFence: Unfenced
     cidrs:
     [...]
   [...]
   ```

3. 删除替换集群的 DRCluster。

   ```
   $ oc delete drcluster <drcluster_name> --wait=false
   ```

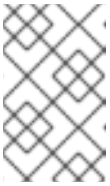
   

   注意

   使用 --wait=false，因为 DRCluster 在稍后的步骤前不会删除。

4. 在 Surviving **集群中为每个受保护的应用程序禁用 Hub** 集群上的灾难恢复。

   a. 对于每个应用程序，编辑放置并确保选择了 Surviving 集群。



注意

对于基于 Subscription 的应用程序，相关的放置可以在与受管集群类似的 hub 集群上的同一命名空间中找到。对于基于 ApplicationSets 的应用程序，相关的放置可在 hub 集群的 **openshift-gitops** 命名空间中找到。

```
$ oc edit placement <placement_name> -n <namespace>
```

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
annotations:
  cluster.open-cluster-management.io/experimental-scheduling-disable: "true"
[...]
spec:
clusterSets:
- submariner
predicates:
- requiredClusterSelector:
    claimSelector: {}
    labelSelector:
      matchExpressions:
      - key: name
        operator: In
        values:
        - cluster1  <-- Modify to be surviving cluster name
[...]
```

   b. 在每个受保护的应用程序的 VolumeReplicationGroup 上运行以下命令来验证为替换集群删除了 **s3Profile**。

```
$ oc get vrg -n <application_namespace> -o jsonpath='{.items[0].spec.s3Profiles}' | jq
```

   c. 在将受保护的应用程序 **放置资源** 配置为使用 Surviving 集群，并替换集群 s3Profile (s)从受保护的应用程序中删除后，所有 **DRPlacementControl** 资源都必须从 **Hub 集群中删除**。

```
$ oc delete drpc <drpc_name> -n <namespace>
```

注意

对于基于 Subscription 的应用程序，相关的 DRPlacementControl 可以在与 hub 集群上的受管集群相同的命名空间中找到。对于基于 ApplicationSets 的应用程序，相关的 DRPlacementControl 可以在 hub 集群的 **openshift-gitops** 命名空间中找到。

   d. 验证所有 DRPlacementControl 资源是否已删除，然后继续下一步。此命令是所有命名空间的查询。应该没有找到资源。

```
$ oc get drpc -A
```

-

e. 最后一步是编辑每个应用程序 **放置** 并删除注解 **cluster.open-cluster-management.io/experimental-scheduling-disable: "true** "。

```
$ oc edit placement <placement_name> -n <namespace>
```

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
annotations:
  ## Remove this annotation
  cluster.open-cluster-management.io/experimental-scheduling-disable: "true"
[...]
```

5. 在存活集群中，重复上一步中详述的流程，以及每个受保护应用程序的子步骤。为受保护的应用程序禁用 DR 现已完成。

6. 在 Hub 集群中，运行以下脚本从 **存活** 集群和 **hub** 集群中删除所有灾难恢复配置。

```
#!/bin/bash
secrets=$(oc get secrets -n openshift-operators | grep Opaque | cut -d" " -f1)
echo $secrets
for secret in $secrets
do
   oc patch -n openshift-operators secret/$secret -p '{"metadata":{"finalizers":null}}' --
type=merge
done
mirrorpeers=$(oc get mirrorpeer -o name)
echo $mirrorpeers
for mp in $mirrorpeers
do
   oc patch $mp -p '{"metadata":{"finalizers":null}}' --type=merge
   oc delete $mp
done
drpolicies=$(oc get drpolicy -o name)
echo $drpolicies
for drp in $drpolicies
do
   oc patch $drp -p '{"metadata":{"finalizers":null}}' --type=merge
   oc delete $drp
done
drclusters=$(oc get drcluster -o name)
echo $drclusters
for drp in $drclusters
do
   oc patch $drp -p '{"metadata":{"finalizers":null}}' --type=merge
   oc delete $drp
done
oc delete project openshift-operators
managedclusters=$(oc get managedclusters -o name | cut -d"/" -f2)
echo $managedclusters
for mc in $managedclusters
do
   secrets=$(oc get secrets -n $mc | grep multicluster.odf.openshift.io/secret-type | cut -d" " -
f1)
```

```
    echo $secrets
    for secret in $secrets
    do
      set -x
      oc patch -n $mc secret/$secret -p '{"metadata":{"finalizers":null}}' --type=merge
      oc delete -n $mc secret/$secret
    done
done

oc delete clusterrolebinding spoke-clusterrole-bindings
```

注意

此脚本使用 **oc delete project openshift-operators** 命令来删除 hub 集群上的这个命名空间中的灾难恢复(DR) Operator。如果在这个命名空间中存在其他非DR operator，则必须从 OperatorHub 再次安装它们。

7. 再次创建命名空间 **openshift-operators** 后，添加监控标签以收集灾难恢复指标。

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

8. 在 Surviving 集群中，确保删除在 DR 安装期间创建的对象存储桶。如果对象存储桶没有被脚本删除，请删除它。用于 DR 的对象存储桶的名称以 **odrbucket** 开始。

```
$ oc get obc -n openshift-storage
```

9. 在 RHACM 控制台中，进入到 **Infrastructure → Clusters** 视图。

   a. 分离替换集群。

   b. 创建新的 OpenShift 集群（恢复集群），并将新集群导入到 RHACM 控制台。具体步骤请参阅创建集群并将 目标受管集群导入到 hub 集群。

10. 在恢复集群中安装 OpenShift Data Foundation 操作器，并将它连接到与存活集群相同的外部 Ceph 存储系统。具体步骤请参考 以外部模式部署 OpenShift Data Foundation。

注意

确保 OpenShift Data Foundation 版本为 4.15 （或更高），并且同一版本的 OpenShift Data Foundation 位于存活集群中。

11. 在 hub 集群中，从 OperatorHub 安装 ODF Multicluster Orchestrator operator。具体步骤，请参阅安装 OpenShift Data Foundation Multicluster Orchestrator operator 的章节。

12. 使用 RHACM 控制台，导航到 **Data Services → Data policies**。

    a. 选择 **Create DRPolicy** 并为您的策略命名。

    b. 选择 **恢复** 群集和 **存活集群**。

    c. 创建策略。具体步骤请查看在 Hub 集群上创建灾难恢复策略 一章。

只有在 DRPolicy 状态变为 **Validated** 后，继续下一步。

13. 将 DRPolicy 应用到最初在替换集群失败前保护的存活集群中的应用程序。

14. 将新受保护的应用重新定位到存活群集上，返回到新的恢复（主）集群。使用 RHACM 控制台，进入到 **Applications** 菜单来执行重新定位。

3.17. 使用 RED HAT ADVANCED CLUSTER MANAGEMENT 进行 HUB 恢复 [技术预览]

当您的设置有主动和被动 Red Hat Advanced Cluster Management for Kubernetes (RHACM) hub 集群时，如果活跃 hub 停机，您可以使用被动 hub 故障切换或重新定位灾难恢复保护的工作负载。

重要

hub 恢复是一个技术预览功能，受技术预览支持限制。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需更多信息，请参阅技术预览功能支持范围。

3.17.1. 配置被动 hub 集群

在活跃 hub 停机或无法访问时执行 hub 恢复，请按照本节中的步骤配置被动 hub 集群，然后故障转移或重新定位灾难恢复受保护的工作负载。

流程

1. 确保在被动 hub 集群上安装 RHACM operator 和 **MultiClusterHub**。具体步骤请查看 RHACM 安装指南。
   成功安装 Operator 后，用户界面中会显示一个带有 Web 控制台更新可用消息的弹出窗口。点此弹出窗口中的 **Refresh Web 控制台**来反映控制台更改。

2. 在 hub 恢复前，配置备份和恢复。请参阅 *RHACM 业务连续性* 指南的 备份和恢复 主题。

3. 在恢复前，在被动 RHACM hub 上安装多集群编配器(MCO) Operator 和 Red Hat OpenShift GitOps operator。有关恢复 RHACM hub 的说明，请参阅安装 OpenShift Data Foundation Multicluster Orchestrator operator。

4. 确保 **Restore.cluster.open-cluster-management.io** 资源将 **.spec.cleanupBeforeRestore** 设置为 **None**。详情请参阅 RHACM 文档的检查备份一章时恢复被动资源。

5. 如果在设置过程中手动配置 SSL 访问，则在集群中重新配置 SSL 访问。具体步骤请参阅 在集群中配置 SSL 访问章节。

6. 在被动 hub 上，添加用于收集灾难恢复指标的监控标签。有关警报详情，请参阅 灾难恢复警报章节。

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

3.17.2. 切换到被动 hub 集群

当活跃 hub 停机或无法访问时，请使用这个步骤。

流程

1. 在被动 hub 集群上恢复备份。如需更多信息，请参阅 从备份中恢复 hub 集群。

重要

将失败的 hub 恢复到其被动实例，仅将应用程序及其 DR 保护状态恢复到其上次调度的备份。在最后一次调度的备份后，任何受 DR 保护的应用程序都需要在新 hub 上再次保护。

2. 验证 Primary 和 Seconday 受管集群是否已成功导入到 RHACM 控制台，并可以访问它们。如果任何受管集群停机或无法访问，则不会成功导入它们。

3. 等待 DRPolicy 验证成功。

4. 验证 **DRPolicy** 是否已成功创建。对于创建的每个 DRPolicy 资源，在 **Hub 集群** 中运行这个命令，其中 <*drpolicy_name* > 替换为唯一名称。

```
$ oc get drpolicy <drpolicy_name> -o jsonpath='{.status.conditions[].reason}{"\n"}'
```

输出示例：

```
Succeeded
```

5. 刷新 RHACM 控制台，以便在 Active hub 集群中启用 DR 监控仪表板标签页访问它。

6. 如果只有活跃 hub 集群停机，请通过执行 hub 恢复来恢复 hub，并在被动 hub 中恢复备份。如果受管集群仍然可以访问，则不需要进一步的操作。

7. 如果主受管集群与活跃 hub 集群一起停机，则需要将工作负载从主受管集群切换到二级受管集群。有关故障转移说明，根据您的工作负载类型，请参阅 基于订阅的应用程序或基于 ApplicationSet 的应用程序。

8. 验证故障转移是否成功。当主受管集群停机时，工作负载的 PROGRESSION 状态将处于 **Cleaning Up** 阶段，直到 down 受管集群恢复在线并成功导入到 RHACM 控制台中。
在被动 hub 集群中，运行以下命令来检查 PROGRESSION 状态。

```
$ oc get drpc -o wide -A
```

输出示例：

```
NAMESPACE NAME AGE PREFERREDCLUSTER
FAILOVERCLUSTER DESIREDSTATE CURRENTSTATE PROGRESSION START
TIME DURATION PEER READY
[...]
busybox cephfs-busybox-placement-1-drpc 103m cluster-1 cluster-2
Failover FailedOver Cleaning Up 2024-04-15T09:12:23Z False
busybox cephfs-busybox-placement-1-drpc 102m cluster-1
Deployed Completed 2024-04-15T07:40:09Z 37.200569819s True
[...]
```

第 4 章 用于 OPENSHIFT DATA FOUNDATION 的 REGION-DR 解决方案

4.1. 区域 DR 解决方案的组件

Region-DR 由 Red Hat Advanced Cluster Management for Kubernetes 和 OpenShift Data Foundation 组件组成，以便在 Red Hat OpenShift Container Platform 集群中提供应用程序和数据移动性。

Red Hat Advanced Cluster Management for Kubernetes

Red Hat Advanced Cluster Management (RHACM)提供了管理多个集群和应用程序生命周期的功能。因此，它充当多集群环境中的控制平面。

RHACM 分为两个部分：

- RHACM Hub：在多集群 control plane 上运行的组件。
- 受管集群：在受管理的集群中运行的组件。

有关此产品的更多信息，请参阅 RHACM 文档 和 RHACM"管理应用程序"文档。

OpenShift Data Foundation

OpenShift Data Foundation 为 OpenShift Container Platform 集群中有状态应用程序提供部署和管理存储的功能。

OpenShift Data Foundation 由 Ceph 作为存储提供商提供支持，其生命周期由 OpenShift Data Foundation 组件堆栈中的 Rook 进行管理。Ceph-CSI 为有状态应用提供持久卷的调配与管理。

OpenShift Data Foundation 堆栈现在增强了以下灾难恢复功能：

- 启用 RBD 块池，以便在 OpenShift Data Foundation 实例（集群）之间进行镜像。
- 可以在一个 RBD 块池中 mirror 特定的镜像
- 提供 csi-addons 以管理每个持久性卷声明(PVC)镜像

OpenShift DR

OpenShift DR 是跨一组使用 RHACM 部署和管理的有状态应用程序的灾难恢复编排器，并提供云原生接口来编排应用程序状态在持久性卷上的生命周期。它们是：

- 保护跨 OpenShift 集群的应用程序及其状态关系
- 将一个应用程序及其状态转移到一个对等集群
- 将一个应用程序及其状态重新定位到之前部署的集群

OpenShift DR 被分成三个组件：

- **ODF Multicluster Orchestrator**: 安装在多集群控制平面 (RHACM Hub) 中，它会编配配置并针对 Metro 和 Regional D 关系对等 OpenShift Data Foundation 集群
- **OpenShift DR Hub Operator**：在 hub 集群上作为 ODF 多集群安装的一部分自动安装，以编配启用 DR 应用程序的故障转移或重新定位。

- **OpenShift DR Cluster Operator**：在属于 Metro 和 Regional DR relationship的每个受管集群中自动安装，用于管理应用程序的所有 PVC 的生命周期。

4.2. 区域DR 部署工作流

本节概述了在两个不同的 OpenShift Container Platform 集群中使用最新版本的 Red Hat OpenShift Data Foundation 配置和部署 Regional-DR 功能所需的步骤。除了两个受管集群外，还需要第三个 OpenShift Container Platform 集群来部署 Red Hat Advanced Cluster Management (RHACM)。

要配置基础架构，请按照给定的顺序执行以下步骤：

1. 确保满足作为 DR 解决方案的三个集群（Hub、Primary 和 Secondary Openshift Container Platform）集群的要求。请参阅启用区域DR 的要求。
2. 安装 OpenShift Data Foundation operator，并在 Primary 和 Secondary 受管集群上创建存储系统。请参阅 在受管集群上创建 OpenShift Data Foundation 集群。
3. 在 Hub 集群上安装 ODF Multicluster Orchestrator。请参阅在 Hub 集群上安装 ODF 多集群编排器。
4. 配置 Hub、Primary 和 Secondary 集群之间的 SSL 访问。请参阅配置跨集群的 SSL 访问。
5. 创建 DRPolicy 资源，以用于在跨 Primary 和 Secondary 应用程序所需的 DR 保护的应用程序。请参阅在 Hub 集群上创建灾难恢复策略。

   注意

   可以有多个策略。

6. 测试您的灾难恢复解决方案：
   a. **基于订阅的应用程序** ：
      - 创建基于订阅的应用程序。请参阅 创建示例应用程序。
      - 使用受管集群之间的基于订阅的示例应用程序测试故障切换和重新定位操作。请参阅基于订阅的应用程序故障切换 和 重新定位基于订阅的应用程序。

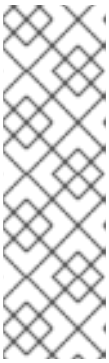
   b. **基于 ApplicationSet 的应用程序**：
      - 创建示例应用程序。请参阅 创建基于 ApplicationSet 的应用程序。
      - 在受管集群之间使用示例应用程序测试故障切换和重定位操作。请参阅 基于 ApplicationSet 的应用程序故障切换和 重新定位基于 ApplicationSet 的应用程序。

4.3. 启用区域DR 的要求

安装 Red Hat OpenShift Data Foundation 支持的灾难恢复解决方案的先决条件如下：

- 您必须有三个在它们之间具有网络可访问性的 OpenShift 集群：
  - 安装 Red Hat Advanced Cluster Management (RHACM) for Kubernetes operator 的 **hub 集群**。
  - 运行 OpenShift Data Foundation 的**主受管集群**。

- 运行 OpenShift Data Foundation 的**从受管集群**。



注意

要配置 hub 恢复设置，您需要 4 个集群，它充当被动 hub。主受管集群(Site-1)可以与活跃的 RHACM hub 集群共存，而被动 hub 集群与二级受管集群(Site-2)一起存在。或者，活跃的 RHACM hub 集群可以放在中立站点(Site-3)中，该集群不受 Site-1 或 Site-2 中主受管集群的故障的影响。在这种情况下，如果使用被动 hub 集群，它可以被放在 Site-2 的次要集群中。如需更多信息，请参阅为 hub 恢复配置被动 hub 集群。

hub 恢复是一个技术预览功能，受技术预览支持限制。

- 确保在 Hub 集群中安装 RHACM 操作符和 MultiClusterHub。具体步骤请查看 RHACM 安装指南。
成功安装 Operator 后，用户界面中会显示一个带有 Web 控制台更新可用消息的弹出窗口。点此弹出窗口中的 **Refresh Web 控制台**来反映控制台更改。

重要

确保正确配置了应用程序流量路由和重定向。

- 在 Hub 集群中
  - 进入到 **All Clusters → Infrastructure → Clusters**。
  - 使用 RHACM 控制台导入或创建**主受管集群**和**次受管集群**。
  - 为您的环境选择适当的选项。

  具体步骤请参阅创建集群并将 目标受管集群导入到 hub 集群。

- 使用 RHACM Submariner 附加组件连接私有 OpenShift 集群和服务网络。验证两个集群是否没有重叠的服务和集群专用网络。否则，请确保在 Submariner 附加组件安装过程中启用了 Globalnet。
为每个受管集群运行以下命令，以确定是否需要启用 Globalnet。此处显示的示例用于非重叠的集群和服务网络，因此不会启用 Globalnet。

```
$ oc get networks.config.openshift.io cluster -o json | jq .spec
```

Primary 集群的示例输出：

```
{
  "clusterNetwork": [
    {
      "cidr": "10.5.0.0/16",
      "hostPrefix": 23
    }
  ],
  "externalIP": {
    "policy": {}
  },
  "networkType": "OVNKubernetes",
  "serviceNetwork": [
```

```
    "10.15.0.0/16"
  ]
}
```

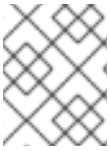
Secondary 集群的输出示例：

```
{
  "clusterNetwork": [
    {
      "cidr": "10.6.0.0/16",
      "hostPrefix": 23
    }
  ],
  "externalIP": {
    "policy": {}
  },
  "networkType": "OVNKubernetes",
  "serviceNetwork": [
    "10.16.0.0/16"
  ]
}
```

如需更多信息，请参阅 Submariner 文档。

4.4. 在受管集群上创建 OPENSHIFT DATA FOUNDATION 集群。

要在两个 OpenShift Container Platform 集群间配置存储复制，请在安装 OpenShift Data Foundation Operator 后创建一个 OpenShift Data Foundation 存储系统。



注意

请参阅 OpenShift Data Foundation 部署指南和特定于基础架构的说明（如 AWS、VMware、BM、Azure 等）。

流程

1. 在每个受管集群中，安装和配置最新的 **OpenShift Data Foundation** 集群。
   如需有关 OpenShift Data Foundation 部署的信息，请参阅基础架构特定的部署指南 （如 AWS、VMware、Bare metal、Azure）。

注意

在创建存储集群时，在 **数据保护** 步骤中，您必须选择 **Prepare cluster for disaster recovery (Regional-DR only)** 复选框。

2. 使用以下命令验证每个受管集群中 OpenShift Data Foundation 部署是否成功：

```
$ oc get storagecluster -n openshift-storage ocs-storagecluster -o jsonpath='{.status.phase}{"\n"}'
```

对于 Multicloud 网关(MCG)：

```
$ oc get noobaa -n openshift-storage noobaa -o jsonpath='{.status.phase}{"\n"}'
```

如果状态结果是 **Ready**，用于**主受管集群**和**二级受管集群**上的查询，则继续执行下一步。

3. 在 OpenShift Web 控制台中，进入到 **Installed Operators → OpenShift Data Foundation → Storage System → ocs-storagecluster-storagesystem → Resources**，并验证 **StorageCluster** 的 **Status** 是否为 **Ready**，并在其旁边有一个绿色勾号标记。

4. [可选] 如果在安装 Submariner 时启用了 Globalnet，然后在 OpenShift Data Foundation 安装完成后编辑 **StorageCluster**。
对于 Globalnet 网络，手动编辑 **StorageCluster** yaml 以添加 **clusterID** 并将 **enabled** 设置为 **true**。将 *<clustername>* 替换为您的 RHACM 导入或新创建的受管集群名称。在主受管集群和次受管集群中编辑 **StorageCluster**。

警告

不要在 **StorageCluster** 中进行这个更改，除非您在安装 Submariner 时启用了 Globalnet。

```
$ oc edit storagecluster -o yaml -n openshift-storage
```

```
spec:
  network:
    multiClusterService:
      clusterID: <clustername>
      enabled: true
```

5. 进行上述更改后，

   a. 等待 OSD pod 重启，并且创建 OSD 服务。

   b. 等待所有 MONS 故障转移。

   c. 确保已导出 MONS 和 OSD 服务。

```
$ oc get serviceexport -n openshift-storage
```

```
NAME              AGE
rook-ceph-mon-d   4d14h
rook-ceph-mon-e   4d14h
rook-ceph-mon-f   4d14h
rook-ceph-osd-0   4d14h
rook-ceph-osd-1   4d14h
rook-ceph-osd-2   4d14h
```

   d. 确保集群处于 **Ready** 状态，集群健康状态有一个绿色勾号表示 **Health ok**。使用第 3 步进行验证。

4.5. 安装 OPENSHIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multicluster Orchestrator 是一个控制器，从 Hub 集群上的 OpenShift Container Platform OperatorHub 安装。

流程

1. 在 **Hub 集群中**，进入到 **OperatorHub** 并使用关键字过滤器搜索 **ODF Multicluster Orchestrator**。
2. 点 **ODF Multicluster Orchestrator** 标题。
3. 保留所有默认设置，然后点**安装**。
   确保 operator 资源安装在 **openshift-operators** 中，并可用于所有命名空间。

注意

ODF Multicluster Orchestrator 还会在 RHACM hub 集群上作为依赖项安装 **Openshift DR Hub Operator**。

4. 验证 Operator **Pod** 处于 **Running** 状态。**OpenShift DR Hub operator** 也安装在 **openshift-operators** 命名空间中。

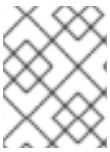
```
$ oc get pods -n openshift-operators
```

输出示例：

```
NAME                                  READY   STATUS      RESTARTS    AGE
odf-multicluster-console-6845b795b9-blxrn   1/1     Running     0          4d20h
odfmo-controller-manager-f9d9dfb59-jbrsd    1/1     Running     0          4d20h
ramen-hub-operator-6fb887f885-fss4w         2/2     Running     0          4d20h
```

4.6. 在集群间配置 SSL 访问

在 **primary 和 secondary 集群**间配置网络 SSL 访问，因此元数据可以以一个安全的方式保持在不同的集群中的 Multicloud Gateway (MCG) **对象存储桶**，以及保持在 **Hub 集群**中验证对对象存储桶的访问。



注意

如果所有 OpenShift 集群都为您的环境使用签名的有效证书集进行部署，则可以跳过本节。

步骤

1. 提取主受管集群的入口证书，并将输出保存到 **primary.crt**。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > primary.crt
```

2. 提取二级受管集群的入口证书，并将输出保存到 **secondary.crt**。

```
$ oc get cm default-ingress-cert -n openshift-config-managed -o jsonpath="{['data']['ca-bundle\.crt']}" > secondary.crt
```

3. 创建新的 ConfigMap 文件，以使用文件名 **cm-clusters-crt.yaml** 来保存远程集群的证书捆绑包。

注意

如本示例文件所示，每个集群可能有超过三个证书。另外，请确保在从之前创建的 **primary.crt** 和 **secondary.crt** 文件中复制并粘贴后，证书内容会被正确缩进。

```
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from primary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 primary.crt here>
    -----END CERTIFICATE----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert1 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert2 from secondary.crt here>
    -----END CERTIFICATE-----

    -----BEGIN CERTIFICATE-----
    <copy contents of cert3 from secondary.crt here>
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: openshift-config
```

4. 在 **Primary 受管集群**, **Secondary 受管集群**, 和 **Hub 集群** 中创建 **ConfigMap**。

```
$ oc create -f cm-clusters-crt.yaml
```

输出示例：

```
configmap/user-ca-bundle created
```

5. 对**主受管集群**、**二级受管集群**和 **Hub 集群**上的默认代理资源进行补丁。

```
$ oc patch proxy cluster --type=merge  --patch='{"spec":{"trustedCA":{"name":"user-ca-bundle"}}}'
```

输出示例：

```
proxy.config.openshift.io/cluster patched
```

4.7. 在 HUB 集群上创建灾难恢复策略

OpenShift 灾难恢复策略(DRPolicy)资源指定参与灾难恢复解决方案和所需的复制间隔的 OpenShift Container Platform 集群。DRPolicy 是一个集群范围的资源，用户可以应用到需要灾难恢复解决方案的应用程序。

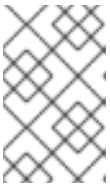
ODF MultiCluster Orchestrator Operator 通过 **Multicluster Web console** 促进每个 **DRPolicy** 和相应的 **DRClusters** 的创建。

先决条件

- 确保至少一组两个受管集群。

流程

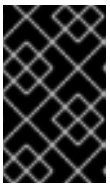
1. 在 **OpenShift 控制台**中，进入到 **All Clusters → Data Services → Data policies**。
2. 点 **Create DRPolicy**。
3. 输入 **Policy name**。确保每个 DRPolicy 都有一个唯一名称（例如： **ocp4bos1-ocp4bos2-5m**）。
4. 从与此新策略关联的受管集群列表中选择两个集群。



注意

如果您在选择集群后收到错误消息 "OSD not migrate"，请按照 将现有 OSD 迁移到 OpenShift Data Foundation for Regional-DR 集群中优化 OSD 的知识库文章进行操作，然后再继续下一步。

5. **复制策略**根据所选的 OpenShift 集群自动设置为 **Asynchronous**(async)，一个 **Sync schedule** 选项变为可用。
6. 设置**同步调度**。



重要

对于每个需要的复制间隔，必须使用唯一名称（如 **ocp4bos1-ocp4bos2-10m**）创建新的 **DRPolicy**。可以选择同一集群，但 **Sync 调度** 可以用分钟/小时/天内使用不同的复制间隔配置。最小值为一分钟。

7. 点 **Create**。
8. 验证 **DRPolicy** 是否已成功创建。在 **Hub 集群**中，为创建的每个 DRPolicy 资源运行这个命令，使用您的唯一名称替换 *<drpolicy_name>*。

```
$ oc get drpolicy <drpolicy_name> -o jsonpath='{.status.conditions[].reason}{"\n"}'
```

输出示例：

```
Succeeded
```

创建 DRPolicy 时，也会创建两个 DRCluster 资源。验证所有三个资源并且状态显示为 **Succeeded** 最多需要 10 分钟。

注意

DRPolicy 不支持编辑 **SchedulingInterval**、**ReplicationClassSelector**、**VolumeSnapshotClassSelector** 和 **DRClusters** 字段值。

9. 验证对象存储桶可以从 **Hub 集群**、**Primary 受管集群**和 **Secondary 受管集群**访问。

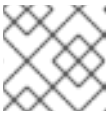
   a. 获取 Hub 集群上的 **DRClusters** 的名称。

   ```
   $ oc get drclusters
   ```

   输出示例：

   ```
   NAME       AGE
   ocp4bos1   4m42s
   ocp4bos2   4m42s
   ```

   b. 检查 S3 访问每个受管集群上创建的每个存储桶。使用 DRCluster 验证命令，使用您的唯一名称替换 *<drcluster_name>*。

   注意

   DRClusters 不支持编辑 **Region** 和 **S3ProfileName** 字段值。

   ```
   $ oc get drcluster <drcluster_name> -o jsonpath='{.status.conditions[2].reason}{"\n"}'
   ```

   输出示例：

   ```
   Succeeded
   ```

   注意

   确保为在 **Hub 集群**中的两个 **DRClusters** 运行命令。

10. 验证 **OpenShift DR Cluster operator** 已成功在 **Primary 受管集群**和 **Secondary 受管集群**上成功安装。

    ```
    $ oc get csv,pod -n openshift-dr-system
    ```

    输出示例：

    ```
    NAME                                                                DISPLAY                       VERSION
    REPLACES   PHASE
    clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.15.0        Openshift DR
    Cluster Operator   4.15.0                    Succeeded
    clusterserviceversion.operators.coreos.com/volsync-product.v0.8.0              VolSync
    0.8.0                     Succeeded
    ```

```
NAME                                    READY   STATUS    RESTARTS   AGE
pod/ramen-dr-cluster-operator-6467cf5d4c-cc8kz   2/2     Running   0          3d12h
```

您还可以验证 **OpenShift DR Cluster Operator** 是否在每个受管集群的 **OperatorHub** 上成功安装。

注意

在初始运行时，会自动安装 VolSync operator。VolSync 用于在两个集群之间设置卷复制，以保护基于 CephF 的 PVC。复制功能默认为启用。

11. 验证 OpenShift Data Foundation 镜像 **Primary 受管集群**和 **Secondary 受管集群**中的 **daemon** 健康的状态。

```
$ oc get cephblockpool ocs-storagecluster-cephblockpool -n openshift-storage -o jsonpath='{.status.mirroringStatus.summary}{"\n"}'
```

输出示例：

```
{"daemon_health":"OK","health":"OK","image_health":"OK","states":{}}
```

小心

daemon_health 和 **health** 从 **Warning** 变为 **OK** 可能需要最多 10 分钟。如果状态最终未变为 **OK**，则使用 RHACM 控制台验证受管集群之间的 Submariner 连接是否仍然处于健康状态。在所有值都变为 **OK** 之前不要继续。

4.8. 为测试灾难恢复解决方案创建示例应用程序

OpenShift Data Foundation 灾难恢复 (DR) 解决方案支持针对由 RHACM 管理的基于 Subscription 和 ApplicationSet 的应用程序的灾难恢复。如需了解更多详细信息，请参阅 订阅和 ApplicationSet 文档。

以下小节详细介绍了如何创建应用程序并将 DRPolicy 应用到应用程序。

- 基于订阅的应用程序
  没有 cluster-admin 权限的 OpenShift 用户，请参阅知识库文章 如何为应用程序用户分配必要的权限来执行灾难恢复操作。

- 基于 ApplicationSet 的应用程序
  没有 cluster-admin 权限的 OpenShift 用户无法创建基于 ApplicationSet 的应用程序。

4.8.1. 基于订阅的应用程序

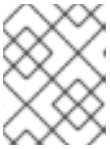
4.8.1.1. 创建基于订阅的应用程序示例

为了测试从 **Primary 受管集群**到 **Secondary 受管集群**的 **failover** 和 **relocate**，我们需要一个简单的应用程序。

先决条件

- 在创建用于常规消耗的应用程序时，请确保应用程序仅部署到一个集群。

- 使用名为 **busybox** 的示例应用作为示例。
- 当应用程序通过或重新定位失败时，确保使用全局流量管理器(GTM)或 Global Server Load Balancing (GLSB)服务配置应用程序的所有外部路由。
- 作为最佳实践，将属于 Red Hat Advanced Cluster Management (RHACM) 订阅的 Red Hat Advanced Cluster Management (RHACM) 订阅分组在一起，以指代单个 Placement Rule 到 DR 保护它们作为组。进一步将它们创建为一个应用程序，用于订阅的逻辑分组，以备将来 DR 操作，如故障转移和重新定位。



注意

如果不相关的订阅引用与放置操作相同的放置规则，它们也会受到 DR 保护，因为 DR 工作流控制引用放置规则的所有订阅。

流程

1. 在 Hub 集群中，进入到 **Applications**，再点 **Create application**。
2. 选择类型 **Subscription**。
3. 输入应用程序 **Name** （如 **busybox**）和 **Namespace** （如 **busybox-sample**）。
4. 在 Repository location for resources 部分中，选择 **Repository type Git**。
5. 输入示例应用程序的 Git 存储库 URL、github **Branch** 和 **Path**，在其中创建资源 **busybox** Pod 和 PVC。
   - 使用示例应用程序存储库作为 **https://github.com/red-hat-storage/ocm-ramen-samples**。
   - 选择 **Branch** 作为 **release-4.15**。
   - 选择以下 **路径之一** ：
     - **busybox-odr** 使用 RBD Regional-DR。
     - **busybox-odr-cephfs** 使用 CephFS Regional-DR。
6. 向下滚动，直到您看到 **Deploy application resources on cluster with all specified labels**。
   - 选择全局 **Cluster set** 或包含环境正确受管集群的全局集群。
   - 添加一个 *<name>* 标签，将其值设置为**受管集群**的名称。
7. 点位于右上角的 **Create**。
   在接下来的屏幕中，进入 **Topology** 选项卡。您可以看到应用程序拓扑上有所有的绿色对勾图标。

注意

要获取更多信息，请点击任何拓扑元素，拓扑视图右侧会显示一个窗口。

8. 验证示例应用程序部署。
   现在，**busybox** 应用程序已部署到首选集群，现在可以验证部署。

   登录到您的受管集群，其中 **busybox** 由 RHACM 部署。

```
$ oc get pods,pvc -n busybox-sample
```

输出示例：

```
NAME                     READY   STATUS    RESTARTS   AGE
pod/busybox-67bf494b9-zl5tr   1/1     Running   0          77s


NAME                         STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS              AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-c732e5fe-daaf-4c4d-99dd-462e04c18412
5Gi        RWO            ocs-storagecluster-ceph-rbd   77s
```

4.8.1.2. 将数据策略应用到示例应用程序

先决条件

- 确保 Data 策略中引用的两个受管集群都可以访问。如果没有，在两个集群都在线前，应用程序不会受到灾难恢复的影响。

流程

1. 在 Hub 集群中，进入到 **All Clusters → Applications**。
2. 点应用程序末尾的 Actions 菜单，以查看可用操作的列表。
3. 点 **Manage data policy→ Assign data policy**。
4. 选择 **Policy**，再单击 **Next**。
5. 选择 **Application resource**，然后使用 **PVC 标签选择器** 为所选应用程序资源选择 **PVC 标签**。

注意

您可以为所选应用程序资源选择多个 PVC 标签。您还可以使用 **Add application resource** 选项添加多个资源。

6. 添加所有应用程序资源后，点 **Next**。
7. 检查 **Policy 配置详情**并点 **Assign**。新分配的 Data 策略显示在 **Manage data policy** modal 列表视图中。
8. 验证您可以在 Applications 页面中查看分配的策略详情。
   a. 在 Applications 页面上，进入到 **Data policy** 列，再单击 **策略链接** 以展开视图。
   b. 验证您可以看到分配的策略数量以及故障转移和重新定位状态。
   c. 单击 **View more details**，以查看与应用程序一起使用的策略持续活动的状态。
9. 可选：在主集群中验证 RADOS 块设备(RBD) **volumereplication** 和 **volumereplicationgroup**。

```
$ oc get volumereplications.replication.storage.openshift.io -A
```

输出示例：

```
NAME          AGE    VOLUMEREPLICATIONCLASS                PVCNAME
DESIREDSTATE   CURRENTSTATE
busybox-pvc    2d16h   rbd-volumereplicationclass-1625360775   busybox-pvc     primary
Primary
```

```
$ oc get volumereplicationgroups.ramendr.openshift.io -A
```

输出示例：

```
NAME          DESIREDSTATE   CURRENTSTATE
busybox-drpc   primary        Primary
```

10. 可选：验证 CephFS volsync 复制源已在主集群中成功设置，在故障切换集群中设置了 VolSync ReplicationDestination。

```
$ oc get replicationsource -n busybox-sample
```

输出示例：

```
NAME          SOURCE         LAST SYNC              DURATION         NEXT SYNC
busybox-pvc    busybox-pvc    2022-12-20T08:46:07Z   1m7.794661104s   2022-12-
20T08:50:00Z
```

```
$ oc get replicationdestination -n busybox-sample
```

输出示例：

```
NAME          LAST SYNC              DURATION         NEXT SYNC
busybox-pvc    2022-12-20T08:46:32Z   4m39.52261108s
```

4.8.2. 基于 ApplicationSet 的应用程序

4.8.2.1. 创建基于 ApplicationSet 的应用程序

前提条件

- 确保 Hub 集群上安装了 Red Hat OpenShift GitOps Operator。具体步骤请查看 RHACM 文档。
- 确保主和从受管集群都注册到 GitOps。有关注册说明，请参阅 将受管集群注册到 GitOps。然后，检查 **GitOpsCluster** 资源使用的放置是否用于注册两个受管集群，具有处理集群的容限。您可以使用 **oc get placement <placement-name> -n openshift-gitops -o yaml** 命令来验证以下容限是否已添加到放置中。

```
tolerations:
- key: cluster.open-cluster-management.io/unreachable
  operator: Exists
- key: cluster.open-cluster-management.io/unavailable
  operator: Exists
```

如果没有添加容限，请参阅为 Red Hat Advanced Cluster Management 和 OpenShift GitOps 配置应用程序放置容限。

流程

1. 在 Hub 集群中，进入到 **All Clusters → Applications** 并点 **Create application**。
2. 选择应用程序类型作为 **Argo CD ApplicationSet - Push model**
3. 在常规步骤 1 中，输入您的**应用程序集名称**。
4. 选择 **Argo server openshift-gitops** 和 **Requeue time** 为 **180** 秒。
5. 点击 **Next**。
6. 在 Repository location for resources 部分中，选择 **Repository type Git**。
7. 输入示例应用程序的 Git 存储库 URL、github Branch 和 Path，在其中创建资源 busybox Pod 和 PVC。
   a. 使用示例应用程序存储库作为 https://github.com/red-hat-storage/ocm-ramen-samples
   b. 选择 **Revision** 作为 **release-4.15**
   c. 选择以下路径之一：
      - **busybox-odr** 使用 RBD Regional-DR。
      - **busybox-odr-cephfs** 使用 CephFS Regional-DR。
8. 输入 **Remote namespace** 值。(例如 busybox-sample)，然后点 **Next**。
9. 选择 **Sync policy** 设置并点 **Next**。
   您可以选择一个或多个选项。
10. 添加一个 *<name>* 标签，将其值设置为**受管集群**的名称。
11. 点击 **Next**。
12. 检查设置详情并点 **Submit**。

4.8.2.2. 将 Data policy 应用到基于 ApplicationSet 的应用程序示例

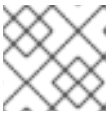
先决条件

- 确保 Data 策略中引用的两个受管集群都可以访问。如果没有，在两个集群都在线前，应用程序不会受到灾难恢复的影响。

流程

1. 在 Hub 集群中，进入到 **All Clusters → Applications**。
2. 点应用程序末尾的 Actions 菜单，以查看可用操作的列表。
3. 点 **Manage data policy→ Assign data policy**。
4. 选择 **Policy**，再单击 **Next**。

5. 选择 **Application resource**，然后使用 **PVC 标签选择器** 为所选应用程序资源选择 **PVC 标签**。



注意

您可以为所选应用程序资源选择多个 PVC 标签。

6. 添加所有应用程序资源后，点 **Next**。

7. 检查 **Policy 配置详情**并点 **Assign**。新分配的 Data 策略显示在 **Manage data policy** modal 列表视图中。

8. 验证您可以在 Applications 页面中查看分配的策略详情。

   a. 在 Applications 页面上，进入到 **Data policy** 列，再单击 **策略链接** 以展开视图。

   b. 验证您可以看到分配的策略数量以及故障转移和重新定位状态。

9. 可选：在主集群中验证 Rados 块设备(RBD) **volumereplication** 和 **volumereplicationgroup**。

```
$ oc get volumereplications.replication.storage.openshift.io -A
```

输出示例：

```
NAME           AGE     VOLUMEREPLICATIONCLASS                PVCNAME
DESIREDSTATE   CURRENTSTATE
busybox-pvc    2d16h   rbd-volumereplicationclass-1625360775   busybox-pvc      primary
Primary
```

```
$ oc get volumereplicationgroups.ramendr.openshift.io -A
```

输出示例：

```
NAME           DESIREDSTATE   CURRENTSTATE
busybox-drpc   primary        Primary
```

10. [可选] 验证 CephFS volsync 复制源已在主集群中成功设置，在故障转移集群中已设置了 VolSync ReplicationDestination。

```
$ oc get replicationsource -n busybox-sample
```

输出示例：

```
NAME           SOURCE         LAST SYNC              DURATION          NEXT SYNC
busybox-pvc    busybox-pvc    2022-12-20T08:46:07Z   1m7.794661104s    2022-12-
20T08:50:00Z
```

```
$ oc get replicationdestination -n busybox-sample
```

输出示例：

```
NAME           LAST SYNC              DURATION          NEXT SYNC
busybox-pvc    2022-12-20T08:46:32Z   4m39.52261108s
```

4.8.3. 删除示例应用程序

本节提供了使用 RHACM 控制台删除示例应用程序 **busybox** 的说明。

重要

当删除 DR 保护的应用程序时，需要可以访问属于 DRPolicy 的这两个集群。这是为了确保根据删除 DR 保护的一部分清理相应 S3 存储中的所有受保护的 API 资源和资源。如果访问其中一个集群不是健康，删除应用程序的 **DRPlacementControl** 资源（在 hub 上）将处于 Deleting 状态。

先决条件

- 这些说明在故障转移和重新定位测试完成前不应执行示例应用程序，且应用程序已准备好从 RHACM 和受管集群中删除。

流程

1. 在 RHACM 控制台上，前往 **Applications**。
2. 搜索要删除的示例应用程序（例如 **busybox**）。
3. 点击您要删除的应用程序旁边的 Action Menu **(⋮)**。
4. 点击 **Delete application**。
   选择了 **Delete application** 时，将显示一个新屏幕，询问是否还应删除与应用相关的资源。
5. 选中 **Remove application 相关资源**复选框，以删除 Subscription 和 PlacementRule。
6. 单击 **Delete**。这将删除主受管集群（或应用程序所运行的任何群集）上的 busybox 应用程序。
7. 除了使用 RHACM 控制台删除的资源外，如果删除 **busybox** 应用后没有自动删除，则删除 **DRPlacementControl**。
   a. 登录到 Hub 集群的 OpenShift Web 控制台，再进入到 项目 **busybox-sample** 的 Installed Operators。
      对于 ApplicationSet 应用程序，将项目选为 **openshift-gitops**。
   b. 单击 **OpenShift DR Hub Operator**，然后单击 **DRPlacementControl** 选项卡。
   c. 点击您要删除的 **busybox** 应用程序 DRPlacementControl 旁边的 Action Menu **(⋮)**。
   d. 单击 **Delete DRPlacementControl**。
   e. 单击 **Delete**。

注意

此过程可用于使用 **DRPlacementControl** 资源删除任何应用。

4.9. 受管集群间基于订阅的应用程序故障切换

故障转移是在主集群失败时将应用程序从主集群转换到二级集群的过程。虽然故障转移提供了应用程序以最小中断在二级集群中运行的功能，但做出不正确的故障转移决策可能会导致意外的故障转移决定，如发生从主到次要集群未验证复制失败时的完整数据丢失。如果自上次成功复制以来有大量的时间，最好等待

失败的主完成。

LastGroupSyncTime 是一个 关键指标，反映了自与应用程序关联的所有 PVC 最后一次成功复制以来的时间。本质上，它会测量主集群和次要集群之间的同步健康状况。因此，在从一个集群启动故障转移前，请检查此指标，并仅在 LastGroupSyncTime 在过去的合理时间内启动故障转移。

注意

在故障转移集群中故障转移 Ceph-RBD 镜像部署期间，会缩减故障转移，以确保 Ceph-RBD 支持的卷的干净故障转移作为存储置备程序。

先决条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为它可能需要一些时间才能更新。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行故障转移操作前，请检查两个受管集群的状态。
     但是，当您切换到的集群处于 *Ready* 状态时，故障转移操作仍然可以运行。
- 与当前时间相比，在 Hub Cluster 上运行以下命令检查 **lastGroupSyncTime** 是否在可接受的数据丢失窗口中。

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```

输出示例：

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

流程

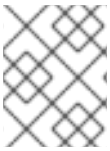
1. 在 Hub 集群中，进入到 **Applications**。
2. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
3. 点 **Failover application**。
4. 在显示了 **Failover 应用程序** 模态后，选择**策略**和**目标集群**，相关的应用程序将在出现灾难时故障转移。
5. 点**选择订阅组**下拉菜单，验证默认选择或修改此设置。
   默认情况下，选择为应用程序资源复制的订阅组。
6. 检查 **Failover readiness** 的状态。
   - 如果状态是 **Ready** 且带有一个绿色勾号，这表示目标集群已就绪，可启动故障转移。继续执行第 7 步。
   - 如果状态是 **Unknown** 或 **Not ready**，请等待到状态变为 **Ready**。

7. 点 **Initiate**。busybox 应用程序现在故障转移到 **Secondary-managed 集群**。
8. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
9. 验证活动状态是否为应用的 **FailedOver**。
    a. 进入 **Applications → Overview** 选项卡。
    b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
    c. 在 **Data policy** 弹出窗口中，点 **View more details** 链接。
    d. 验证您能否看到一个或多个策略名称以及与应用程序中使用的策略关联的持续活动（同步时间和活动状态）。

4.10. 受管集群之间基于 APPLICATIONSET 的应用程序故障切换

故障转移是在主集群失败时将应用程序从主集群转换到二级集群的过程。虽然故障转移提供了应用程序以最小中断在二级集群中运行的功能，但做出不正确的故障转移决策可能会导致意外的故障转移决定，如发生从主到次要集群未验证复制失败时的完整数据丢失。如果自上次成功复制以来有大量的时间，最好等待失败的主完成。

LastGroupSyncTime 是一个 关键指标，反映了自与应用程序关联的所有 PVC 最后一次成功复制以来的时间。本质上，它会测量主集群和次要集群之间的同步健康状况。因此，在从一个集群启动故障转移前，请检查此指标，并仅在 LastGroupSyncTime 在过去的合理时间内启动故障转移。



注意

在故障转移集群中故障转移 Ceph-RBD 镜像部署期间，会缩减故障转移，以确保 Ceph-RBD 支持的卷的干净故障转移作为存储置备程序。

先决条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为它可能需要一些时间才能更新。
    1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
    2. 在执行故障转移操作前，请检查两个受管集群的状态。
    但是，当您切换到的集群处于 *Ready* 状态时，故障转移操作仍然可以运行。
- 与当前时间相比，在 Hub Cluster 上运行以下命令检查 **lastGroupSyncTime** 是否在可接受的数据丢失窗口中。

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```

输出示例：

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

流程

1. 在 Hub 集群中，进入到 **Applications**。
2. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
3. 点 **Failover application**。
4. 显示 **Failover 应用程序** 模态时，验证显示的详细信息是否正确并检查 Failover **readiness** 的状态。如果状态是 **Ready** 且带有绿色勾号，这表示目标集群已准备好故障转移启动。
5. 点 **Initiate**。busybox 资源现在在目标集群上创建。
6. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
7. 验证活动状态是否为应用的 **FailedOver**。
   a. 进入 **Applications → Overview** 选项卡。
   b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   c. 在 **Data policy** 弹出窗口中，验证您可以看到一个或多个策略名称以及与应用程序中使用的策略关联的持续活动。

4.11. 在受管集群间重新定位基于订阅的应用程序

当所有受管集群都可用时，将应用程序重新定位到首选位置。

前提条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为可能需要一些时间来更新。只有在主集群和首选集群启动并运行时才可执行重新定位。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行重新定位操作前，请单独检查两个受管集群的状态。
- 与当前时间相比，当 **lastGroupSyncTime** 位于复制间隔（例如 5 分钟）时，执行重定位。建议为任何单个应用程序最小化恢复时间目标(RTO)。
  在 Hub 集群中运行这个命令：

  ```
  $ oc get drpc -o yaml -A | grep lastGroupSyncTime
  ```

  输出示例：

  ```
  [...]
  lastGroupSyncTime: "2023-07-10T12:40:10Z"
  ```

  将输出时间(UTC)与当前时间进行比较，以验证所有 **lastGroupSyncTime** 值是否在应用程序复制间隔内。如果没有，请等待 Relocate，直到所有 **lastGroupSyncTime** 值都为 true。

流程

1. 在 Hub 集群中，进入到 **Applications**。

2. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
3. 点 **Relocate application**。
4. 当 **Relocate application** 模态显示时，选择 **policy** 和 **target cluster**，在出现灾难时相关的应用程序将重新定位到其中。
5. 默认情况下，选择部署应用程序资源的订阅组。**点选择订阅组**下拉菜单，验证默认选择或修改此设置。
6. 检查 **Relocation readiness** 的状态。
   - 如果状态是 **Ready** 且带有一个绿色勾号，这表示目标集群已准备好重定位来进行启动。继续执行第 7 步。
   - 如果状态是 **Unknown** 或 **Not ready**，请等待到状态变为 **Ready**。
7. 点 **Initiate**。busybox 资源现在在目标集群上创建。
8. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
9. 验证应用程序的活动状态是否显示为 **Relocated**。
   a. 进入 **Applications → Overview** 选项卡。
   b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   c. 在 **Data policy** 弹出窗口中，点 **View more details** 链接。
   d. 验证您能否看到一个或多个策略名称以及与应用程序中使用的策略关联的持续活动（同步时间和活动状态）。

4.12. 在受管集群间重新定位基于 APPLICATIONSET 的应用程序

当所有受管集群都可用时，将应用程序重新定位到首选位置。

前提条件

- 如果您的设置具有主动和被动 RHACM hub 集群，请参阅使用 Red Hat Advanced Cluster Management 进行 Hub 恢复。
- 当主集群处于 **Ready** 以外的状态时，请检查集群的实际状态，因为可能需要一些时间来更新。只有在主集群和首选集群启动并运行时才可执行重新定位。
  1. 进入到 **RHACM 控制台 → Infrastructure → Clusters → Cluster list**选项卡。
  2. 在执行重新定位操作前，请单独检查两个受管集群的状态。
- 与当前时间相比，当 **lastGroupSyncTime** 位于复制间隔（例如 5 分钟）时，执行重定位。建议为任何单个应用程序最小化恢复时间目标(RTO)。
  在 Hub 集群中运行这个命令：

  ```
  $ oc get drpc -o yaml -A | grep lastGroupSyncTime
  ```

  输出示例：

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

将输出时间(UTC)与当前时间进行比较，以验证所有 **lastGroupSyncTime** 值是否在应用程序复制间隔内。如果没有，请等待 Relocate，直到所有 **lastGroupSyncTime** 值都为 true。

流程

1. 在 Hub 集群中，进入到 **Applications**。
2. 点应用程序行末尾的 **Actions** 菜单，以查看可用操作的列表。
3. 点 **Relocate application**。
4. 当 **Relocate application** 模态显示时，选择 **policy** 和 **target cluster**，在出现灾难时相关的应用程序将重新定位到其中。
5. 点 **Initiate**。busybox 资源现在在目标集群上创建。
6. 关闭模态窗口，并使用 Applications 页面中的 **Data policies** 列跟踪状态。
7. 验证应用程序的活动状态是否显示为 **Relocated**。
   a. 进入 **Applications → Overview** 选项卡。
   b. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   c. 在 **Data policy** 弹出窗口中，验证您可以看到一个或多个策略名称以及与应用程序一起使用的策略关联的重新定位状态。

4.13. 查看启用灾难恢复应用程序的恢复点目标值

恢复点目标 (RPO) 值是应用程序当前有效的集群中的持久数据的最新同步时间。此同步时间有助于确定在故障切换过程中丢失的数据持续时间。

注意

此 RPO 值仅适用于故障切换期间的 Regional-DR。重新定位可确保操作过程中没有数据丢失，因为所有对等集群都可用。

您可以在 Hub 集群中查看所有受保护的卷的恢复点目标 (RPO) 值。

流程

1. 在 Hub 集群中，进入到 **Applications → Overview** 选项卡。
2. 在 **Data policy** 列中，点您要将策略应用到的应用程序的**策略**链接。
   Data policies modal 页面会出现，每个应用程序所应用的灾难恢复策略数量以及故障转移和重新定位状态。
3. 在 **Data policies** 模态页面中，点 **View more details** 链接。
   此时会显示一个详细的 **数据策略** 模态页面，其中显示策略名称和与应用到应用程序的策略关联的持续活动(Last sync、Activity 状态)。

在模态页面中报告的 **Last sync 时间** 代表，表示所有针对应用程序保护的 DR 的卷的最新同步时间。

4.14. 使用 RED HAT ADVANCED CLUSTER MANAGEMENT 进行 HUB 恢复 [技术预览]

当您的设置有主动和被动 Red Hat Advanced Cluster Management for Kubernetes (RHACM) hub 集群时，如果活跃 hub 停机，您可以使用被动 hub 故障切换或重新定位灾难恢复保护的工作负载。

重要

hub 恢复是一个技术预览功能，受技术预览支持限制。技术预览功能不受红帽产品服务等级协议（SLA）支持，且功能可能并不完整。红帽不推荐在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需更多信息，请参阅技术预览功能支持范围。

4.14.1. 配置被动 hub 集群

在活跃 hub 停机或无法访问时执行 hub 恢复，请按照本节中的步骤配置被动 hub 集群，然后故障转移或重新定位灾难恢复受保护的工作负载。

流程

1. 确保在被动 hub 集群上安装 RHACM operator 和 **MultiClusterHub**。具体步骤请查看 RHACM 安装指南。
   成功安装 Operator 后，用户界面中会显示一个带有 Web 控制台更新可用消息的弹出窗口。点此弹出窗口中的 **Refresh Web 控制台**来反映控制台更改。

2. 在 hub 恢复前，配置备份和恢复。请参阅 *RHACM 业务连续性* 指南的 备份和恢复 主题。

3. 在恢复前，在被动 RHACM hub 上安装多集群编配器(MCO) Operator 和 Red Hat OpenShift GitOps operator。有关恢复 RHACM hub 的说明，请参阅安装 OpenShift Data Foundation Multicluster Orchestrator operator。

4. 确保 **Restore.cluster.open-cluster-management.io** 资源将 **.spec.cleanupBeforeRestore** 设置为 **None**。详情请参阅 RHACM 文档的检查备份一章时恢复被动资源。

5. 如果在设置过程中手动配置 SSL 访问，则在集群中重新配置 SSL 访问。具体步骤请参阅 在集群中配置 SSL 访问章节。

6. 在被动 hub 上，将标签添加到 **openshift-operators** 命名空间，以便使用以下命令启用 **VolumeSyncronizationDelay** 警报的基本监控。有关警报详情，请参阅 灾难恢复警报 章节。

   ```
   $ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
   ```

4.14.2. 切换到被动 hub 集群

当活跃 hub 停机或无法访问时，请使用这个步骤。

流程

1. 在被动 hub 集群上恢复备份。如需更多信息，请参阅 从备份中恢复 hub 集群。

重要

将失败的 hub 恢复到其被动实例，仅将应用程序及其 DR 保护状态恢复到其上次调度的备份。在最后一次调度的备份后，任何受 DR 保护的应用程序都需要在新 hub 上再次保护。

2. 在被动 hub 上导入受管集群后，Submariner 会被自动安装。

3. 验证 Primary 和 Seconday 受管集群是否已成功导入到 RHACM 控制台，并可以访问它们。如果任何受管集群停机或无法访问，则不会成功导入它们。

4. 等待 DRPolicy 验证成功。

5. 验证 **DRPolicy** 是否已成功创建。对于创建的每个 DRPolicy 资源，在 **Hub 集群** 中运行这个命令，其中 <*drpolicy_name* > 替换为唯一名称。

```
$ oc get drpolicy <drpolicy_name> -o jsonpath='{.status.conditions[].reason}{"\n"}'
```

输出示例：

```
Succeeded
```

6. 刷新 RHACM 控制台，以便在 Active hub 集群中启用 DR 监控仪表板标签页访问它。

7. 如果只有活跃 hub 集群停机，请通过执行 hub 恢复来恢复 hub，并在被动 hub 中恢复备份。如果受管集群仍然可以访问，则不需要进一步的操作。

8. 如果主受管集群与活跃 hub 集群一起停机，则需要将工作负载从主受管集群切换到二级受管集群。有关故障转移说明，根据您的工作负载类型，请参阅 基于订阅的应用程序或基于 ApplicationSet 的应用程序。

9. 验证故障转移是否成功。当主受管集群停机时，工作负载的 PROGRESSION 状态将处于 **Cleaning Up** 阶段，直到 down 受管集群恢复在线并成功导入到 RHACM 控制台中。
在被动 hub 集群中，运行以下命令来检查 PROGRESSION 状态。

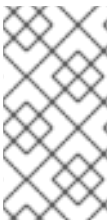
```
$ oc get drpc -o wide -A
```

输出示例：

```
NAMESPACE NAME AGE PREFERREDCLUSTER
FAILOVERCLUSTER DESIREDSTATE CURRENTSTATE PROGRESSION START
TIME DURATION PEER READY
[...]
busybox cephfs-busybox-placement-1-drpc 103m cluster-1 cluster-2
Failover FailedOver Cleaning Up 2024-04-15T09:12:23Z False
busybox cephfs-busybox-placement-1-drpc 102m cluster-1
Deployed Completed 2024-04-15T07:40:09Z 37.200569819s True
[...]
```

第 5 章 使用 OPENSHIFT DATA FOUNDATION 的扩展集群进行灾难恢复

Red Hat OpenShift Data Foundation 部署可以在两个不同的地理位置扩展，为存储基础架构提供灾难恢复功能。当遇到灾难时，如两个位置中的一个部分或完全不可用，OpenShift Container Platform 部署中的 OpenShift Data FoundationOpenShift 必须能够存活。此解决方案仅适用于在基础架构服务器之间具有特定延迟要求的 edorpolitan 跨数据中心。

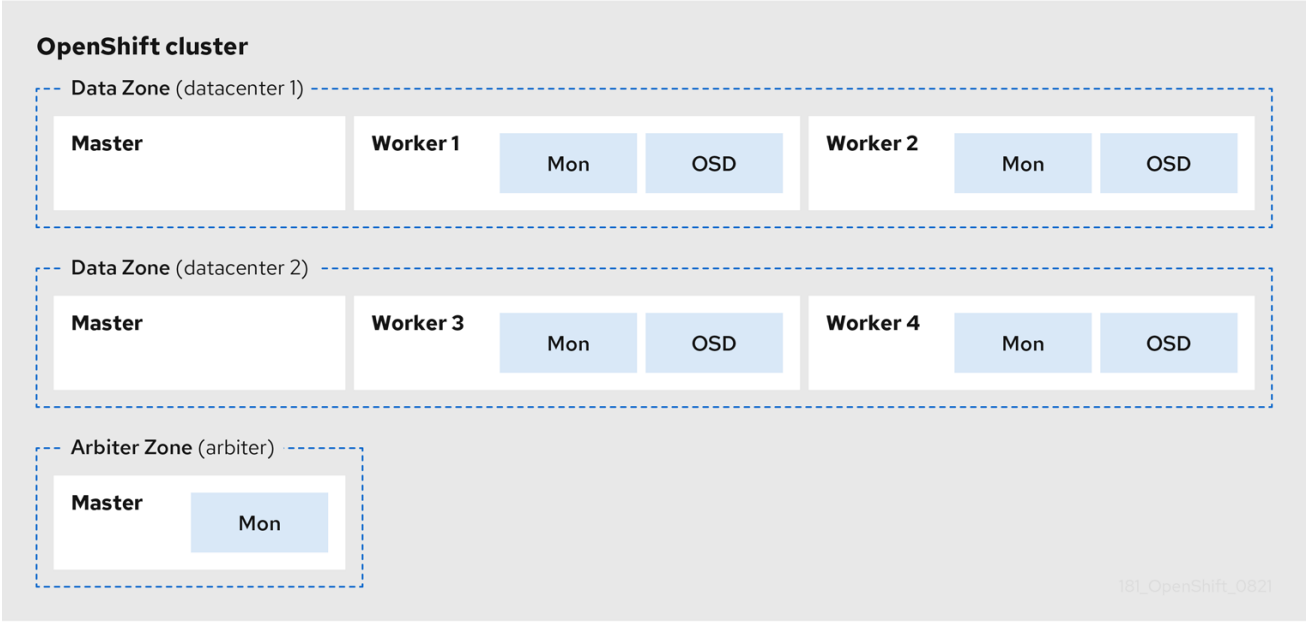


注意

扩展集群解决方案是为包含数据卷的区域之间延迟不超过 10 ms 的最大往返用时(RTT)的部署而设计。对于 Arbiter 节点遵循为 etcd 指定的延迟要求，请参阅 Red Hat OpenShift Container Platform 集群的指南 - Deployments Spanning Multiple Sites (Data Centers/Regions)。如果您计划以更高的延迟进行部署，请联系红帽客户支持。

下图显示了扩展集群的最简单的部署：

OpenShift 节点和 OpenShift Data Foundation 守护进程



在上图中，Arbiter 区域中部署的 OpenShift Data Foundation 监控 pod 具有 master 节点内置的容错能力。图中显示了每个 Data Zone 中的 master 节点，它们是高可用性 OpenShift Container Platform control plane 所需的 master 节点。另外，重要的一点是，其中一个区的 OpenShift Container Platform 节点与另外两个区的 OpenShift Container Platform 节点需要具有网络连接。

5.1. 启用扩展集群的要求

- 确保已解决了跨越多个站点的部署的 OpenShift Container Platform 要求。如需更多信息，请参阅有关跨多个站点的集群部署的知识库文章。
- 确保您在三个不同的区中至少有三个 OpenShift Container Platform master 节点。三个区域中的每一区域中有一个主节点。
- 确保至少四个 OpenShift Container Platform worker 节点分布在两个数据区中。

- 对于裸机上的扩展集群，请使用 SSD 驱动器作为 OpenShift Container Platform master 节点的 root 驱动器。
- 确保每个节点已预先标记其 zone 标签。如需更多信息，请参阅将拓扑区标签应用到 OpenShift Container Platform 节点部分。
- 扩展集群解决方案是为在不同区域之间延迟不超过 10 毫秒的部署而设计。如果您计划以更高的延迟进行部署，请联系红帽客户支持。

注意

灵活的缩放和仲裁程序不能与缩放逻辑冲突同时启用。通过灵活扩展，您可以一次向 OpenShift Data Foundation 集群添加一个节点。而在 Arbiter 集群中，您需要在每个数据区中添加至少一个节点。

5.2. 将拓扑区标签应用到 OPENSHIFT CONTAINER PLATFORM 节点

在一个站点停机时，具有仲裁器功能的区域使用仲裁器标签。这些标签是任意的，对于这三个位置来说必须是唯一的。

例如，您可以按如下方式标记节点：

```
topology.kubernetes.io/zone=arbiter for Master0

topology.kubernetes.io/zone=datacenter1 for Master1, Worker1, Worker2

topology.kubernetes.io/zone=datacenter2 for Master2, Worker3, Worker4
```

- 将标签应用到节点：

  ```
  $ oc label node <NODENAME> topology.kubernetes.io/zone=<LABEL>
  ```

  <NODENAME>
  是节点的名称

  <LABEL>
  是拓扑区标签

- 使用三个区的示例标签来验证标签：

  ```
  $ oc get nodes -l topology.kubernetes.io/zone=<LABEL> -o name
  ```

  <LABEL>
  是拓扑区标签
  或者，您可以运行单个命令来查看带有其区域的所有节点。

  ```
  $ oc get nodes -L topology.kubernetes.io/zone
  ```

扩展集群拓扑区域标签现在应用于适当的 OpenShift Container Platform 节点来定义三个位置。

后续步骤

- 从 OpenShift Container Platform Web 控制台安装本地存储 Operator。

5.3. 安装 LOCAL STORAGE OPERATOR

在本地存储设备上创建 Red Hat OpenShift Data Foundation 集群前，请先从 Operator Hub 安装 Local Storage Operator。

流程

1. 登录 OpenShift Web 控制台。
2. 点 **Operators → OperatorHub**。
3. 在 **Filter by keyword** 框中键入 **local storage**，从操作器列表中搜索 **Local Storage operator** 并单击它。
4. 在 **Install Operator** 页面中设置以下选项：
   a. 将频道更新为 **stable**。
   b. 安装模式是 **A specific namespace on the cluster**。
   c. Installed Namespace 为 **Operator recommended namespace openshift-local-storage**。
   d. 将批准更新为 **Automatic**。
5. 点 **Install**。

验证步骤

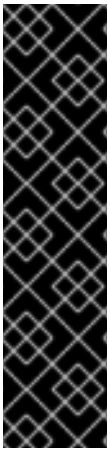
- 验证 Local Storage Operator 是否显示绿色勾号，代表安装成功。

5.4. 安装 RED HAT OPENSHIFT DATA FOUNDATION OPERATOR

您可以使用 Red Hat OpenShift Container Platform Operator Hub 安装 Red Hat OpenShift Data Foundation Operator。

先决条件

- 使用具有 cluster-admin 和 Operator 安装权限的账户访问 OpenShift Container Platform 集群。
- 您必须至少在 Red Hat OpenShift Container Platform 集群的两个数据中心中平均分配四个 worker 节点。
- 有关其他资源要求，请参阅规划您的部署。



重要

- 当您需要覆盖 OpenShift Data Foundation 的集群范围默认节点选择器时，您可以在命令行界面中使用以下命令为 **openshift-storage** 命名空间指定空白节点选择器（在这种情况下创建 openshift-storage 命名空间）：

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

- 将节点作为 **infra** 污点，以确保只在该节点上调度 Red Hat OpenShift Data Foundation 资源。这有助于您节省订阅成本。如需更多信息，请参阅 *管理和分配存储资源指南*中的如何将专用 worker 节点用于 Red Hat OpenShift Data Foundation 章节。

流程

1. 登录 OpenShift Web 控制台。
2. 点 **Operators → OperatorHub**。
3. 在 **Filter by keyword** 框中滚动或键入 **OpenShift Data Foundation**，以查找 **OpenShift Data Foundation** Operator。
4. 点 **Install**。
5. 在 **Install Operator** 页面中设置以下选项：
   a. 将频道更新为 **stable-4.15**。
   b. 安装模式是 **A specific namespace on the cluster**。
   c. Installed Namespace 为 **Operator recommended namespace openshift-storage**。如果命名空间 **openshift-storage** 不存在，它会在 Operator 安装过程中创建。
   d. 将 **Approval Strategy** 选为 **Automatic** 或 **Manual**。
   如果选择 **Automatic** 更新，Operator Lifecycle Manager(OLM)将自动升级 Operator 的运行实例，而无需任何干预。

   如果选择了 **手动** 更新，则 OLM 会创建一个更新请求。作为集群管理员，您必须手动批准该更新请求，才能将 Operator 更新至更新的版本。
6. 确保为 **Console 插件** 选择了 **Enable** 选项。
7. 点 **Install**。

验证步骤

- 成功安装 Operator 后，用户界面中会显示一个带有 **Web console update is available** 信息的弹出窗口。点这个弹出窗口中的 **Refresh web console** 来反映控制台的更改。
- 在 Web 控制台中：
  - 进入到 Installed Operators，再验证 **OpenShift Data Foundation** Operator 是否显示绿色勾号，指示安装成功。
  - 进入到 **Storage**，再验证 **Data Foundation** 仪表板是否可用。

后续步骤

- 创建 OpenShift Data Foundation 集群。

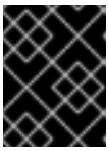
5.5. 创建 OPENSHIFT DATA FOUNDATION 集群

先决条件

- 确保您已满足 启用扩展集群部分的要求。

流程

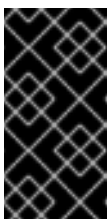
1. 在 OpenShift Web 控制台中，点 **Operators → Installed Operators** 查看所有已安装的 Operator。
   确保所选 **项目** 为 **openshift-storage**。
2. 单击 **OpenShift Data Foundation** 操作器，然后单击 **Create StorageSystem**。
3. 在 Backing storage 页面中，选择 **Create a new StorageClass using the local storage devices** 选项。
4. 点击 **Next**。



重要

如果还没有安装，系统会提示您安装 Local Storage Operator。点 **Install** 并按照 Installing Local Storage Operator 中所述的步骤进行操作。

5. 在 **Create local volume set** 页面中，提供以下信息：
   a. 为 **LocalVolumeSet** 和 **StorageClass** 输入一个名称。
   默认情况下，存储类名称会出现本地卷集名称。您可以更改名称。
   b. 选择以下任意一项：
   - **所有节点上的磁盘**
     使用与所有节点上所选过滤器匹配的可用磁盘。
   - **所选节点上的磁盘**
     仅在所选节点上使用与所选过滤器匹配的可用磁盘。



重要

如果选择的节点与 OpenShift Data Foundation 的一个聚合的 30 个 CPU 和 72 GiB RAM 的要求不匹配，则会部署一个最小的集群。

如需最低起始节点要求，请参阅 *规划指南* 中的资源要求部分。

   c. 选择 **SSD** 或 **NVMe** 来构建受支持的配置。您可以为不支持的测试安装选择 **HDD**。
   d. 展开 **Advanced** 部分并设置以下选项：

卷模式	默认会选择块。

设备类型	从下拉列表中选择一个或多个设备类型。
磁盘大小	为设备设置最小 100GB 大小，以及需要包含的设备的最大可用大小。
磁盘限制上限	这表示节点上可以创建的 PV 数量上限。如果此字段留空，则为匹配节点上的所有可用磁盘创建 PV。

e. 点击 **Next**。
此时会显示一个用于确认创建 LocalVolumeSet 的弹出窗口。

f. 单击 **Yes** 以继续。

6. 在 **Capacity 和 nodes** 页面中，配置以下内容：

a. **可用的原始容量**会根据与存储类关联的所有附加磁盘填充容量值。这将需要一些时间才能出现。
Selected nodes 列表根据存储类显示节点。

b. 如果要使用扩展集群，选择 **Enable arbiter** 复选框。这个选项只有在满足仲裁程序的所有先决条件并且填充了所选节点时才可用。如需更多信息，请参阅启用扩展集群 的要求中的 Arbiter 扩展集群 要求。
从下拉列表中选择 **arbiter 区域**。

7. 为 **配置性能选择性能配置集**。
您还可以使用 **StorageSystems** 选项卡的选项菜单中的 **Configure performance** 选项，在部署后配置性能配置集。

重要

在选择资源配置集前，请确保检查集群中资源的当前可用性。在资源不足的集群中选择较高的资源配置集可能会导致安装失败。有关资源要求的更多信息，请参阅性能配置集的资源要求。

a. 点击 **Next**。

8. 可选：在 Security and network 页面中，根据您的要求进行配置：

a. 若要启用加密，可选择为**块存储和文件存储启用数据加密。**

b. 选择以下**加密级别**之一：

- **Cluster-wide encryption** 来加密整个集群（块存储和文件存储）。
- **Storage class encryption** 以使用加密启用的存储类创建加密的持久性卷（仅限块）。

c. 可选：选择 **Connect to an external key management service**复选框。这是集群范围加密的可选选项。

i. 从 **Key Management Service Provider** 下拉列表中，选择 **Vault** 或 **Thales CipherTrust Manager (using KMIP)**。如果选择了 **Vault**，请进入下一步。如果您选择了 **Thales CipherTrust Manager （using KMIP）**，请转到步骤 iii。

ii. 选择**身份验证方法**。

使用令牌验证方法

- 输入唯一的**连接名称** , Vault 服务器的主机**地址**（'https://<hostname 或 ip>'），**端口**号和**令牌**。
- 展开 **Advanced Settings**，以根据您的 **Vault** 配置输入其他设置和证书详情：
  - 在**后端路径**中输入为 OpenShift Data Foundation 专用且唯一的 Key Value secret 路径。
  - （可选）输入 **TLS 服务器名称**和 **Vault Enterprise 命名空间**。
  - 上传对应的 PEM 编码证书文件，以提供 **CA 证书**、**客户端证书**和**客户端私钥**。
  - 点 **Save** 并跳过步骤 iv。

使用 Kubernetes 验证方法

- 输入唯一的 Vault **Connection Name**, Vault 服务器的主机**地址**（'https://<hostname 或 ip>'）、**端口**号和**角色**名称。
- 展开 **Advanced Settings**，以根据您的 **Vault** 配置输入其他设置和证书详情：
  - 在**后端路径**中输入为 OpenShift Data Foundation 专用且唯一的 Key Value secret 路径。
  - 可选：输入 **TLS Server Name** 和 **Authentication Path**（如果适用）。
  - 上传对应的 PEM 编码证书文件，以提供 **CA 证书**、**客户端证书**和**客户端私钥**。
  - 点 **Save** 并跳过步骤 iv。

iii. 要使用 **Thales CipherTrust Manager (using KMIP)** 作为 KMS 供应商，请按照以下步骤执行：

A. 在项目中输入密钥管理服务的唯一**连接名称**。

B. 在 **Address** 和 **Port** 部分中，输入 Thales CipherTrust Manager 的 IP 以及在其中启用了 KMIP 接口的端口。例如：

- **地址**: 123.34.3.2
- **端口** ：5696

C. 上传 **客户端证书**、**CA 证书**和 **客户端私钥**。

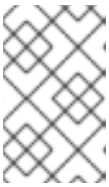
D. 如果启用了 StorageClass 加密，请输入用于加密和解密的唯一标识符。

E. **TLS Server** 字段是可选的，并在没有 KMIP 端点的 DNS 条目时使用。例如，**kmip_all_<port>.ciphertrustmanager.local**。

d. **如果您使用** 单个网络，则 network 设置为 Default (OVN)。
如果您使用多个网络接口，您可以切换到 Custom (Multus)，然后选择以下任一网络接口：

i. 从下拉菜单中选择公共网络接口。

ii. 从下拉菜单中选择集群网络接口。



注意

如果您只使用一个额外网络接口，请选择单个 **NetworkAttachementDefinition**，即 **ocs-public-cluster** 用于公共网络接口，并将 Cluster Network Interface 留空。

a. 点击 **Next**。

9. 在 **Data Protection** 页面中，点 **Next**。

10. 在 **Review and create** 页面中，检查配置详情。
若要修改任何配置设置，请单击 **Back** 以返回到上一配置页面。

11. 单击 **Create StorageSystem**。

验证步骤

- 验证已安装存储集群的最终状态：

  a. 在 OpenShift Web 控制台中，导航到 **Installed Operators → OpenShift Data Foundation → Storage System → ocs-storagecluster-storagesystem → Resources**。

  b. 验证 **StorageCluster** 的 **Status** 是否为 **Ready**，并在其旁边有一个绿色勾号标记。

- 对于部署的仲裁模式：

  a. 在 OpenShift Web 控制台中，导航到 **Installed Operators → OpenShift Data Foundation → Storage System → ocs-storagecluster-storagesystem → Resources → ocs-storagecluster**。

  b. 在 YAML 选项卡中，在 **spec** 部分搜索 **arbiter** 键，并确保 **enable** 设置为 **true**。

```
spec:
    arbiter:
      enable: true
    [..]
    nodeTopologies:
      arbiterLocation: arbiter #arbiter zone
    storageDeviceSets:
    - config: {}
      count: 1
        [..]
      replica: 4
status:
    conditions:
    [..]
    failureDomain: zone
```

- 要验证 OpenShift Data Foundation 的所有组件是否已成功安装，请参阅验证您的 OpenShift 数据基础安装。

5.6. 验证 OPENSHIFT DATA FOUNDATION

验证 OpenShift Data Foundation 是否已正确部署：

- 验证容器集的状态。
- 验证 OpenShift Data Foundation 集群是否健康。
- 验证 Multicloud 对象网关是否健康。
- 验证 OpenShift Data Foundation 特定的存储类是否存在。

5.6.1. 验证 pod 的状态

流程

1. 从 OpenShift Web 控制台点 **Workloads → Pods**。
2. 从 **Project** 下拉列表中选择 **openshift-storage**。

注意

如果禁用 **Show default projects** 选项，请使用切换按钮列出所有默认项目。

有关每个组件预期的 pod 数量及其变化取决于节点数量的更多信息，请参阅 表 5.1 "对应 OpenShift Data Foundation 集群的 Pod"。

3. 点 **Running** 和 **Completed** 标签页验证以下 pod 是否处于 **Running** 和 **Completed** 状态：

表 5.1. 对应 OpenShift Data Foundation 集群的 Pod

组件	对应的 pod
OpenShift Data Foundation Operator	• **OCS-operator-***（在任何 worker 节点上有 1 个 pod） • **ocS-metrics-exporter-***（任何 worker 节点上 1 个 pod） • **odF-operator-controller-manager-***（任何 worker 节点上 1 个 pod） • **odf-console-*** (任何 worker 节点上 1 个 pod) • **csi-addons-controller-manager-***（任何 worker 节点上 1 个 pod）
Rook-ceph Operator	**rook-ceph-operator-*** （任何 worker 节点上有 1 个 pod）

组件	对应的 pod
多云对象网关	• **noobaa-operator-*** (任何 worker 节点上 1 个 pod) • **noobaa-core-*** (任何存储节点上 1 个 pod) • **noobaa-db-pg-*** (任何存储节点上 1 个 pod) • **noobaa-endpoint-*** (任何存储节点上 1 个 pod)
MON	**rook-ceph-mon-*** （5 个 pod 分布到 3 个区域，每个数据中心区域 2 个，1 个在仲裁区域内）
MGR	**rook-ceph-mgr-*** （任何存储节点 2 个 pod）
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** （2 个 pod 分布到 2 个数据中心区域）
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** （2 个 pod 分布到 2 个数据中心区域）

CSI	• **cephfs**
	  ◦ **csi-cephfsplugin-*** (每个 worker 节点上 1 个 pod)
	  ◦ **csi-cephfsplugin-provisioner-*** （2 个 pod 在不同的 worker 节点上分布）
	• **rbd**
	  ◦ **csi-rbdplugin-*** (每个 worker 节点上 1 个 pod)
	  ◦ **csi-rbdplugin-provisioner-*** （2 个 pod 在不同的 worker 节点上分步）

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** （每个存储节点 1 个 pod，1 个 pod 在仲裁区域上）

组件	对应的 pod
OSD	• **rook-ceph-osd-*** (每个设备 1 个 pod) • **rook-ceph-osd-prepare-ocs-deviceset-*** （每个设备 1 个 pod）

5.6.2. 验证 OpenShift Data Foundation 集群是否健康

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
3. 在 **Block and File** 选项卡的 **Status** 卡中，验证 *Storage Cluster* 是否具有绿色勾号。
4. 在 **Details** 卡中，验证是否显示集群信息。

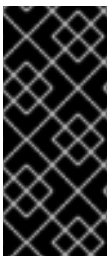
如需有关使用 **Block and File** 仪表板的 OpenShift Data Foundation 集群健康的更多信息，请参阅 监控 OpenShift Data Foundation。

5.6.3. 验证 Multicloud 对象网关是否健康

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
   a. 在 **Object** 选项卡的 **Status** 卡 中，验证 *Object Service* 和 *数据弹性* 都具有绿色勾号。
   b. 在 **Details** 卡中，验证是否显示了 MCG 信息。

如需有关使用对象服务仪表板的 OpenShift Data Foundation 集群健康的更多信息，请参阅监控 OpenShift Data Foundation。



重要

Multicloud 对象网关只有一个数据库副本(NooBaa DB)。这意味着，如果 NooBaa DB PVC 被破坏，且我们无法恢复它，可能会导致位于 Multicloud 对象网关上的应用程序数据总数丢失。因此，红帽建议定期备份 NooBaa DB PVC。如果 NooBaa DB 失败且无法恢复，您可以恢复到最新的备份版本。有关备份 NooBaa DB 的说明，请按照 本了解的gabase 文章中的步骤操作。

5.6.4. 验证特定的存储类是否存在

流程

1. 从 OpenShift Web 控制台左侧窗格中，点击 **Storage → Storage Classes**。
2. 验证是否在创建 OpenShift Data Foundation 集群时创建了以下存储类：

- **ocs-storagecluster-ceph-rbd**
- **ocs-storagecluster-cephfs**
- **openshift-storage.noobaa.io**
- **ocs-storagecluster-ceph-rgw**

5.7. 安装区示例应用程序

部署区域感知示例应用，以验证 OpenShift Data Foundation，扩展集群设置是否正确配置。

重要

随着数据区域之间的延迟，可以预期与节点和区域之间具有低延迟的 OpenShift 集群（例如，同一位置的所有节点）相比，可以看到性能下降。性能下降的速度取决于区域之间的延迟和使用存储（如高写流量）的应用程序行为。请确保使用扩展集群配置测试关键应用程序，以确保应用程序性能达到所需的服务水平。

ReadWriteMany(RWX)持久性卷声明(PVC)是使用 **ocs-storagecluster-cephfs** 存储类创建的。多个 pod 同时使用新创建的 RWX PVC。使用的应用称为文件上传程序。

演示应用程序如何在拓扑区间分布，以便在站点中断时仍然可用：

注意

此演示是可行的，因为此应用共享同一个 RWX 卷来存储文件。这也适用于持久数据访问，因为 Red Hat OpenShift Data Foundation 被配置为具有区感知和高可用性的扩展集群。

1. 创建一个新项目。

```
$ oc new-project my-shared-storage
```

2. 部署名为 file-uploader 的示例 PHP 应用。

```
$ oc new-app openshift/php:latest~https://github.com/mashetty330/openshift-php-upload-demo --name=file-uploader
```

输出示例：

```
Found image 4f2dcc0 (9 days old) in image stream "openshift/php" under tag "7.2-ubi8" for "openshift/php:7.2-
ubi8"

Apache 2.4 with PHP 7.2
-----------------------
PHP 7.2 available as container is a base platform for building and running various PHP 7.2 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts to make it easy for developers to write dynamically generated web pages. PHP also offers built-in database integration for several commercial and non-commercial database management systems, so writing a database-enabled webpage with PHP is fairly simple. The most common
use of PHP coding is probably as a replacement for CGI scripts.
```

```
Tags: builder, php, php72, php-72

* A source build using source code from https://github.com/christianh814/openshift-php-upload-demo will be cr
eated
* The resulting image will be pushed to image stream tag "file-uploader:latest"
* Use 'oc start-build' to trigger a new build

--> Creating resources ...
    imagestream.image.openshift.io "file-uploader" created
    buildconfig.build.openshift.io "file-uploader" created
    deployment.apps "file-uploader" created
    service "file-uploader" created
--> Success
    Build scheduled, use 'oc logs -f buildconfig/file-uploader' to track its progress.

    Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
     'oc expose service/file-uploader'

    Run 'oc status' to view your app.
```

3. 查看构建日志并等待应用部署好。

```
$ oc logs -f bc/file-uploader -n my-shared-storage
```

输出示例：

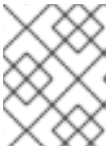
```
Cloning "https://github.com/christianh814/openshift-php-upload-demo" ...

    [...]
    Generating dockerfile with builder image image-registry.openshift-image-regis
try.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c
0e05b593844b41d5494ea
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/openshift/php@s
ha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea
STEP 2: LABEL "io.openshift.build.commit.author"="Christian Hernandez <christ
ian.hernandez@yahoo.com>"       "io.openshift.build.commit.date"="Sun Oct 1 1
7:15:09 2017 -0700"       "io.openshift.build.commit.id"="288eda3dff43b02f7f7
b6b6b6f93396ffdf34cb2"       "io.openshift.build.commit.ref"="master"       "
io.openshift.build.commit.message"="trying to modularize"       "io.openshift
.build.source-location"="https://github.com/christianh814/openshift-php-uploa
d-demo"       "io.openshift.build.image"="image-registry.openshift-image-regi
stry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610
c0e05b593844b41d5494ea"
STEP 3: ENV OPENSHIFT_BUILD_NAME="file-uploader-1"
OPENSHIFT_BUILD_NAMESP
ACE="my-shared-storage"     OPENSHIFT_BUILD_SOURCE="https://github.com/christ
ianh814/openshift-php-upload-demo"     OPENSHIFT_BUILD_COMMIT="288eda3dff43b0
2f7f7b6b6b6f93396ffdf34cb2"
STEP 4: USER root
STEP 5: COPY upload/src /tmp/src
STEP 6: RUN chown -R 1001:0 /tmp/src
STEP 7: USER 1001
```

```
STEP 8: RUN /usr/libexec/s2i/assemble
---> Installing application source...
=> sourcing 20-copy-config.sh ...
---> 17:24:39     Processing additional arbitrary httpd configuration provide
d by s2i ...
=> sourcing 00-documentroot.conf ...
=> sourcing 50-mpm-tuning.conf ...
=> sourcing 40-ssl-certs.sh ...
STEP 9: CMD /usr/libexec/s2i/run
STEP 10: COMMIT temp.builder.openshift.io/my-shared-storage/file-uploader-1:3
b83e447
Getting image source signatures

[...]
```

当您看到 **Push successful** 后，命令提示符会从 tail 模式返回。



注意

new-app 命令直接从 git 存储库部署应用，而不使用 OpenShift 模板，因此默认情况下不创建 OpenShift 路由资源。您需要手动创建路线。

扩展应用程序

1. 将应用缩放为四个副本，并公开其服务，使应用区域了解和可用。

```
$ oc expose svc/file-uploader -n my-shared-storage
```

```
$ oc scale --replicas=4 deploy/file-uploader -n my-shared-storage
```

```
$ oc get pods -o wide -n my-shared-storage
```

几分钟后，您应该有四个 file-uploader pod。重复上述命令，直到有 4 个文件加载程序容器集处于 **Running** 状态。

2. 创建 PVC 并将其附加到应用程序中。

```
$ oc set volume deploy/file-uploader --add --name=my-shared-storage \
-t pvc --claim-mode=ReadWriteMany --claim-size=10Gi \
--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \
--mount-path=/opt/app-root/src/uploaded \
-n my-shared-storage
```

这个命令：

- 创建 PVC。
- 更新应用部署，以包含卷定义。
- 更新应用部署，将卷挂载附加到指定的 mount-path 中。
- 使用四个应用 pod 创建一个新部署。

3. 检查添加卷的结果。

```
$ oc get pvc -n my-shared-storage
```

输出示例：

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS                 AGE
my-shared-storage   Bound    pvc-5402cc8a-e874-4d7e-af76-1eb05bd2e7c7   10Gi       RWX
ocs-storagecluster-cephfs    52s
```

注意 **ACCESS MODE** 已设置为 RWX。

所有四个 **file-uploader** pod 都使用相同的 RWX 卷。如果没有这种访问模式，OpenShift 不会尝试可靠地将多个容器集附加到同一持久卷(PV)。如果您试图扩展正在使用 ReadWriteOnce(RWO) PV 的部署，则 pod 可能会在同一节点上在一起。

5.7.1. 安装后扩展应用程序

流程

1. 将应用缩放为四个副本，并公开其服务，使应用区域了解和可用。

```
$ oc expose svc/file-uploader -n my-shared-storage
```

```
$ oc scale --replicas=4 deploy/file-uploader -n my-shared-storage
```

```
$ oc get pods -o wide -n my-shared-storage
```

几分钟后，您应该有四个 file-uploader pod。重复上述命令，直到有 4 个文件加载程序容器集处于 **Running** 状态。

2. 创建 PVC 并将其附加到应用程序中。

```
$ oc set volume deploy/file-uploader --add --name=my-shared-storage \
-t pvc --claim-mode=ReadWriteMany --claim-size=10Gi \
--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \
--mount-path=/opt/app-root/src/uploaded \
-n my-shared-storage
```

这个命令：

- 创建 PVC。
- 更新应用部署，以包含卷定义。
- 更新应用部署，将卷挂载附加到指定的 mount-path 中。
- 使用四个应用 pod 创建一个新部署。

3. 检查添加卷的结果。

```
$ oc get pvc -n my-shared-storage
```

输出示例：

```
NAME                STATUS   VOLUME                                     CAPACITY   ACCESS MODES
STORAGECLASS                 AGE
my-shared-storage   Bound    pvc-5402cc8a-e874-4d7e-af76-1eb05bd2e7c7   10Gi       RWX
ocs-storagecluster-cephfs    52s
```

注意 **ACCESS MODE** 已设置为 RWX。

所有四个 **file-uploader** pod 都使用相同的 RWX 卷。如果没有这种访问模式，OpenShift 不会尝试可靠地将多个容器集附加到同一持久卷(PV)。如果您试图扩展正在使用 ReadWriteOnce(RWO) PV 的部署，则 pod 可能会在同一节点上在一起。

5.7.2. 将 Deployment 修改为区域 Aware

目前，**file-uploader** Deployment 不是区域感知型，它可以调度同一区域中的所有 pod。在这种情况下，如果站点中断，则应用不可用。如需更多信息，请参阅使用 pod 拓扑分布限制控制 pod 放置。

1. 在应用部署配置中添加容器集放置规则，使应用区域感知。

   a. 运行以下命令并查看输出：

   ```
   $ oc get deployment file-uploader -o yaml -n my-shared-storage | less
   ```

   输出示例：

   ```
   [...]
   spec:
     progressDeadlineSeconds: 600
     replicas: 4
     revisionHistoryLimit: 10
     selector:
       matchLabels:
         deployment: file-uploader
     strategy:
       rollingUpdate:
         maxSurge: 25%
         maxUnavailable: 25%
       type: RollingUpdate
     template:
       metadata:
         annotations:
           openshift.io/generated-by: OpenShiftNewApp
         creationTimestamp: null
         labels:
           deployment: file-uploader
         spec: # <-- Start inserted lines after here
           containers: # <-- End inserted lines before here
           - image: image-registry.openshift-image-registry.svc:5000/my-shared-storage/file-
   uploader@sha256:a458ea62f990e431ad7d5f84c89e2fa27bdebdd5e29c5418c70c56eb81f
   0a26b
             imagePullPolicy: IfNotPresent
             name: file-uploader
   [...]
   ```

   b. 编辑部署以使用拓扑区标签。

```
$ oc edit deployment file-uploader -n my-shared-storage
```

在 **Start** 和 **End** （上一步中的输出中显示）之间添加以下新行：

```
[...]
   spec:
     topologySpreadConstraints:
       - labelSelector:
           matchLabels:
             deployment: file-uploader
         maxSkew: 1
         topologyKey: topology.kubernetes.io/zone
         whenUnsatisfiable: DoNotSchedule
       - labelSelector:
           matchLabels:
             deployment: file-uploader
         maxSkew: 1
         topologyKey: kubernetes.io/hostname
         whenUnsatisfiable: ScheduleAnyway
     nodeSelector:
       node-role.kubernetes.io/worker: ""
     containers:
[...]
```

输出示例：

```
deployment.apps/file-uploader edited
```

2. 将部署缩减为**零**个容器集，然后重新扩展到**四个**容器集。这是必要的，因为部署在 pod 的放置方面发生了变化。

   缩减至零个 pod

   ```
   $ oc scale deployment file-uploader --replicas=0 -n my-shared-storage
   ```

   输出示例：

   ```
   deployment.apps/file-uploader scaled
   ```

   扩展到四个 pod

   ```
   $ oc scale deployment file-uploader --replicas=4 -n my-shared-storage
   ```

   输出示例：

   ```
   deployment.apps/file-uploader scaled
   ```

3. 验证这四个容器集分散到 datacenter1 和 datacenter2 区域中的四个节点上。

   ```
   $ oc get pods -o wide -n my-shared-storage | egrep '^file-uploader'| grep -v build | awk '{print $7}' | sort | uniq -c
   ```

输出示例：

```
1 perf1-mz8bt-worker-d2hdm
1 perf1-mz8bt-worker-k68rv
1 perf1-mz8bt-worker-ntkp8
1 perf1-mz8bt-worker-qpwsr
```

搜索所用的区域标签。

```
$ oc get nodes -L topology.kubernetes.io/zone | grep datacenter | grep -v master
```

输出示例：

```
perf1-mz8bt-worker-d2hdm   Ready   worker   35d   v1.20.0+5fbfd19   datacenter1
perf1-mz8bt-worker-k68rv   Ready   worker   35d   v1.20.0+5fbfd19   datacenter1
perf1-mz8bt-worker-ntkp8   Ready   worker   35d   v1.20.0+5fbfd19   datacenter2
perf1-mz8bt-worker-qpwsr   Ready   worker   35d   v1.20.0+5fbfd19   datacenter2
```

4. 使用浏览器中的 file-uploader Web 应用上传新文件。

   a. 查找创建的路由。

      ```
      $ oc get route file-uploader -n my-shared-storage -o jsonpath --
      template="http://{.spec.host}{'\n'}"
      ```

      输出示例：

      ```
      http://file-uploader-my-shared-storage.apps.cluster-ocs4-abdf.ocs4-
      abdf.sandbox744.opentlc.com
      ```

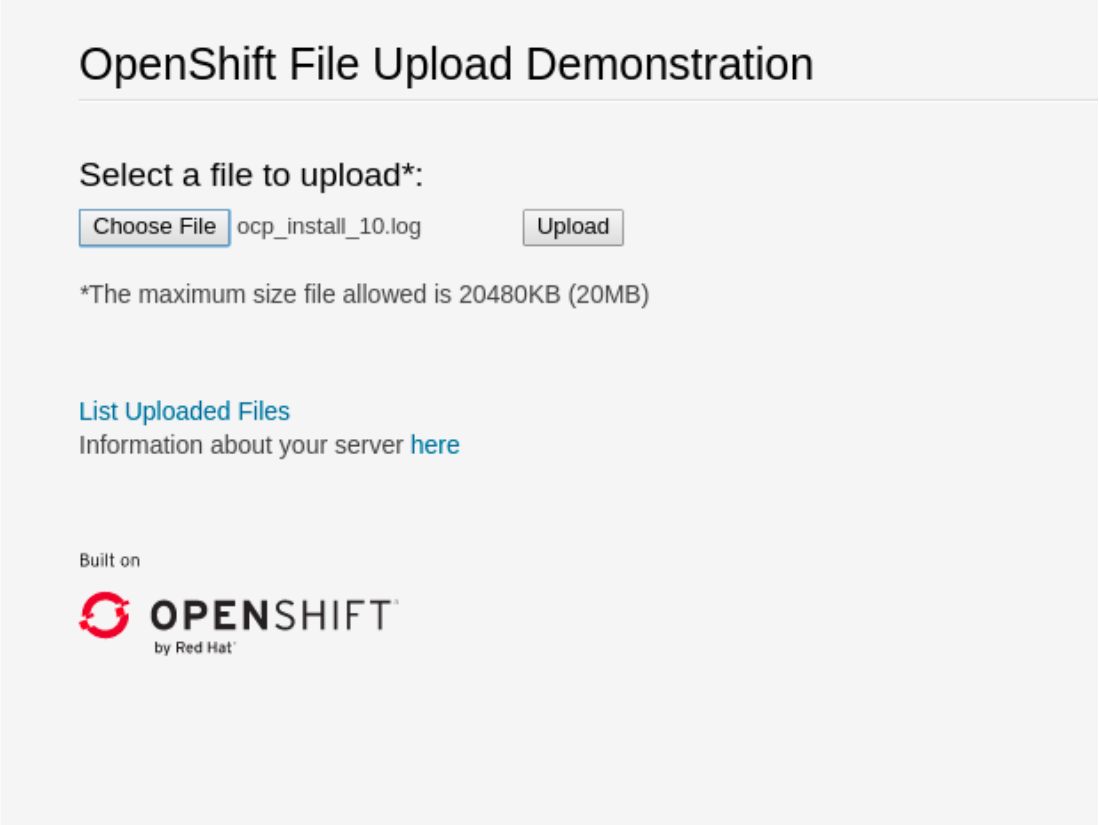
   b. 使用上一步中的路由将浏览器指向 Web 应用。
      Web 应用会列出所有上传的文件，并可以上传新文件并下载现有数据。现在，并没有任何内容。

   c. 从本地计算机中选择一个任意文件，并将它上传到应用程序。

      i. 单击 **Choose file** 以选择任意文件。

      ii. 点 **Upload**。

图 5.1. 基于 PHP 的简单文件上传工具



d. 单击 **已上传文件列表**，以查看所有当前上传的文件的列表。

注意

OpenShift Container Platform 镜像 registry、入口路由和监控服务不被区识别

5.8. 恢复 OPENSHIFT DATA FOUNDATION 扩展集群

鉴于具有扩展集群灾难恢复解决方案是为环境出现完全或部分故障时为环境提供抗压性，因此了解应用程序及其存储的不同恢复方法非常重要。

应用程序的机构决定了它在一个活动区域中再次可用的速度。

当区域出现问题时，有不同的方法来恢复应用及其存储。恢复时间取决于应用程序的架构。不同的恢复方法如下：

- 使用 RWX 存储恢复区域感知 HA 应用程序。
- 使用 RWX 存储恢复 HA 应用程序.
- 使用 RWO 存储恢复应用程序.
- 恢复 StatefulSet pod.

5.8.1. 了解区失败

在本小节中，区失败代表区中的所有 OpenShift Container Platform、master 和 worker 节点不再与第二个数据区中的资源进行通信（例如关闭节点）。如果数据区域之间的通信仍处于可以部分工作的状态（不时出现启动或关闭的情况），集群、存储和网络管理员需要断开数据区之间的通信路径，才能成功恢复。

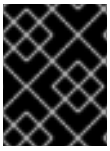
重要

安装示例应用程序时，请关闭 OpenShift Container Platform 节点（至少使用 OpenShift Data Foundation 设备的节点），以测试数据区的故障，以验证您的 file-uploader 应用程序是否可用，您可以上传新文件。

5.8.2. 使用 RWX 存储恢复区感知 HA 应用程序

使用 **topologyKey: topology.kubernetes.io/zone** 部署的应用程序，在每个数据区中调度一个或多个副本，并使用共享存储，即 ReadWriteMany (RWX) CephFS 卷，在几分钟后，在几分钟后，新 pod 会被推出部署并处于待处理状态，直到区域恢复为止。

此类型的应用示例在 Install Zone Aware Sample Application 部分中进行了详细说明。



重要

在区域恢复期间，如果应用容器集在挂载 CephFS 卷时进入 CrashLoopBackOff (CLBO) 状态，然后重启调度 Pod 的节点。等待一段时间，然后检查 pod 是否再次运行。

5.8.3. 使用 RWX 存储恢复 HA 应用程序

使用 **topologyKey: kubernetes.io/hostname** 或没有拓扑配置的应用程序无法防止同一区域中的所有应用副本。

注意

即使 **Pod** spec 中有 *podAntiAffinity* 和 **topologyKey: kubernetes.io/hostname** 也会发生，因为此反关联性规则基于主机且不是基于区域的规则。

如果发生这种情况，且所有副本都位于失败的区域中，则使用 ReadWriteMany(RWX)存储的应用程序需要 6-8 分钟才可以在活跃区域中恢复。此暂停时间用于故障区中的 OpenShift Container Platform 节点变为 **NotReady**（60 秒），然后让默认 pod 驱除超时过期（300 秒）。

5.8.4. 使用 RWO 存储恢复应用程序

使用 ReadWriteOnce(RWO)存储的应用程序具有在此 Kubernetes 问题中描述的已知行为。因此，如果数据区失败，该区中的任何应用程序 pod 都挂载 RWO 卷（例如，基于 **cephrbd** 的卷）都会在 6-8 分钟后处于 **Terminating** 状态，且不会在没有手动干预的情况下在活跃区中重新创建。

检查 OpenShift Container Platform 节点，其状态为 **NotReady**。可能会出现阻止节点与 OpenShift 控制平面通信的问题。但是，节点可能仍然在对持久卷(PV)执行 I/O 操作。

如果两个 pod 同时写入同一个 RWO 卷，则存在数据崩溃的风险。确保 **NotReady** 节点上的进程被终止或阻止，直到它们终止为止。

解决方案示例：

- 使用带外管理系统在确认的情况下关闭节点，以确保进程终止。
- 撤回节点在故障站点使用的网络路由与存储通信。

注意

在将服务恢复到失败的区域或节点前，请确认所有带有 PV 的 pod 已被成功终止。

要让 **Terminating** pod 在活跃区中重新创建，您可以强制删除 pod 或删除关联的 PV 上的终结器。完成这两个操作之一后，应用程序 Pod 应在 active 区域中重新创建，并成功挂载其 RWO 存储。

强制删除 pod

强制删除不会等待来自 kubelet 的确认，但不会等待 pod 已终止。

```
$ oc delete pod <PODNAME> --grace-period=0 --force --namespace <NAMESPACE>
```

<PODNAME>

是 pod 的名称

<NAMESPACE>

是项目的命名空间

删除关联的 PV 上的终结器

找到由 Terminating pod 挂载的持久性卷声明(PVC)关联的 PV，并使用 **oc patch** 命令删除终结器。

```
$ oc patch -n openshift-storage pv/<PV_NAME> -p '{"metadata":{"finalizers":[]}}' --type=merge
```

<PV_NAME>

是 PV 的名称

查找关联的 PV 的一种简单方法是描述 Terminating pod。如果您看到多附件警告，在警告中应具有 PV 名称（例如： **pvc-0595a8d2-683f-443b-aee0-6e547f5f5a7c**）。

```
$ oc describe pod <PODNAME> --namespace <NAMESPACE>
```

<PODNAME>

是 pod 的名称

<NAMESPACE>

是项目的命名空间

输出示例：

```
[...]
Events:
  Type     Reason              Age   From                     Message
  ----     ------              ----  ----                     -------
  Normal   Scheduled           4m5s  default-scheduler        Successfully assigned openshift-
storage/noobaa-db-pg-0 to perf1-mz8bt-worker-d2hdm
  Warning  FailedAttachVolume  4m5s  attachdetach-controller  Multi-Attach error for volume
"pvc-0595a8d2-683f-443b-aee0-6e547f5f5a7c" Volume is already exclusively attached to one
node and can't be attached to another
```

5.8.5. 恢复 StatefulSet pod

作为 StatefulSet 一部分的 Pod 的问题与 pod 挂载 ReadWriteOnce(RWO)卷相似。Kubernetes 资源 StatefulSet 注意事项中会引用更多信息。

要让 StatefulSet 的 pod 部分在 6-8 分钟后在活跃区中重新创建，您需要强制删除具有与 RWO 卷的 pod 相同的要求（即，关闭或断开通信）的 pod。

第 6 章 监控灾难恢复健康状况

6.1. 为灾难恢复启用监控

使用这个流程为您的灾难恢复设置启用基本监控。

流程

1. 在 Hub 集群中，打开一个终端窗口
2. 将以下标签添加到 **openshift-operator** 命名空间。

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

6.2. 在 HUB 集群上启用灾难恢复仪表板

本节介绍了在 Hub 集群上为高级监控启用灾难恢复仪表板。

对于 Regional-DR，仪表板会显示 operator 健康、集群运行状况、指标、警报和应用程序计数的监控状态卡。

对于 Metro-DR，您可以将仪表板配置为仅监控帧设置健康和应用程序计数。

先决条件

- 确保已安装以下内容
  - OpenShift Container Platform 版本 4.15 并具有管理员特权。
  - 启用控制台插件的 ODF Multicluster Orchestrator。
  - 来自 Operator Hub 的 Red Hat Advanced Cluster Management for Kubernetes 2.10 (RHACM)。有关如何安装的步骤，请参阅安装 RHACM。
- 确保已在 RHACM 上启用了可观察性。请参阅 启用可观察性指南。

流程

1. 在 Hub 集群中，打开终端窗口并执行以下步骤。
2. 创建名为 **observability-metrics-custom-allowlist.yaml** 的 configmap 文件。
   您可以使用以下 YAML 列出 Hub 集群上的灾难恢复指标。详情请参阅 添加自定义指标。要了解有关帧指标的更多信息，请参阅 灾难恢复指标。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: observability-metrics-custom-allowlist
  namespace: open-cluster-management-observability
data:
  metrics_list.yaml: |
    names:
      - ceph_rbd_mirror_snapshot_sync_bytes
      - ceph_rbd_mirror_snapshot_snapshots
```

```
    matches:
      - __name__="csv_succeeded",exported_namespace="openshift-dr-system",name=~"odr-cluster-operator.*"
      - __name__="csv_succeeded",exported_namespace="openshift-operators",name=~"volsync.*"
```

3. 在 **open-cluster-management-observability** 命名空间中运行以下命令：

```
$ oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml
```

4. 创建 **observability-metrics-custom-allowlist** yaml 后，RHACM 将开始从所有受管集群收集列出的 OpenShift Data Foundation 指标。
要排除特定受管集群来收集可观察性数据，请在集群中添加以下集群标签 **clusters: observability: disabled**。

6.3. 查看灾难恢复复制关系的健康状态

先决条件

确保您为监控启用了灾难恢复仪表板。具体步骤，请参阅在 Hub 集群上启用灾难恢复仪表板。

流程

1. 在 Hub 集群中，确保选择了 **All Clusters** 选项。
2. 刷新控制台，使 DR 监控仪表板标签页可以访问。
3. 进入 **Data Services** 并点 **Data policies**。
4. 在 Overview 选项卡中，您可以查看 operator、集群和应用程序的健康状态。绿色勾号表示操作器正在运行且可用。
5. 点 **Disaster recovery** 选项卡查看 DR 策略详情和连接的应用程序列表。

6.4. 灾难恢复指标

这些是 prometheus scrapped 的框架指标。

- ramen_last_sync_timestamp_seconds
- ramen_policy_schedule_interval_seconds
- ramen_last_sync_duration_seconds
- ramen_last_sync_data_bytes

从安装了 Red Hat Advanced Cluster Management for Kubernetes (RHACM operator)的 Hub 集群运行这些指标。

最后同步时间戳（以秒为单位）

这是以秒为单位，可让每个应用程序最近成功同步所有 PVC 的时间（以秒为单位）。

指标名称

ramen_last_sync_timestamp_seconds

指标类型

量表

标签

- **ObjType** ：对象的类型，这里的 DPPC
- **ObjName**: 对象的名称，这里是 DRPC-Name
- **ObjNamespace**: DRPC 命名空间
- **policyName**: DRPolicy 的名称
- **schedulingInterval** ：来自 DRPolicy 的调度间隔值

指标值

该值被设置为 Unix 秒，从 DRPC 状态获取 **lastGroupSyncTime**。

策略调度间隔（以秒为单位）

这提供了 DRPolicy 的调度间隔（以秒为单位）。

指标名称

ramen_policy_schedule_interval_seconds

指标类型

量表

标签

- **policyName**: DRPolicy 的名称

指标值

这设置为从 DRPolicy 进行的调度间隔（以秒为单位）。

最后同步持续时间（以秒为单位）

这代表了从每个应用程序所有 PVC 的最新成功同步的最长时间。

指标名称

ramen_last_sync_duration_seconds

指标类型

量表

标签

- **obj_type** ：对象的类型，这里是 DPPC
- **obj_name**: 对象的名称，这里是 DRPC-Name
- **obj_namespace**: DRPC 命名空间
- **scheduling_interval** ：来自 DRPolicy 的调度间隔值

指标值

该值取自 DRPC 状态的 **lastGroupSyncDuration**。

从最新同步传输的总字节数

这个值代表从每个应用程序的所有 PVC 的最新成功同步传输的总字节数。

指标名称

ramen_last_sync_data_bytes

指标类型

量表

标签

- **obj_type** ：对象的类型，这里是 DPPC
- **obj_name**: 对象的名称，这里是 DRPC-Name
- **obj_namespace**: DRPC 命名空间
- **scheduling_interval** ：来自 DRPolicy 的调度间隔值

指标值

该值取自 DRPC 状态中的 **lastGroupSyncBytes**。

6.5. 灾难恢复警报

本节提供了在灾难恢复环境中与 Red Hat OpenShift Data Foundation 关联的所有支持警报的列表。

记录规则

- 记录：**ramen_sync_duration_seconds**

  表达式

  ```
  sum by (obj_name, obj_namespace, obj_type, job, policyname)(time() -
  (ramen_last_sync_timestamp_seconds > 0))
  ```

  用途

  卷组最后一次同步时间和时间（以秒为单位）之间的时间间隔。

- 记录：**ramen_rpo_difference**

  表达式

  ```
  ramen_sync_duration_seconds{job="ramen-hub-operator-metrics-service"} /
  on(policyname, job) group_left() (ramen_policy_schedule_interval_seconds{job="ramen-
  hub-operator-metrics-service"})
  ```

  用途

  预期同步延迟和卷复制组所使用的实际同步延迟之间的差别。

- 记录：**count_persistentvolumeclaim_total**

  表达式

```
count(kube_persistentvolumeclaim_info)
```

用途

来自受管集群的所有 PVC 的总和。

警报

- 警报：**VolumeSynchronizationDelay**

  影响

  Critical

  用途

  卷复制组占用的实际同步延迟是延迟预期同步延迟。

  YAML

```
alert: VolumeSynchronizationDela
expr: ramen_rpo_difference >= 3
for: 5s
labels:
  cluster: '{{ $labels.cluster }}'
  severity: critical
annotations:
  description: >-
    Syncing of volumes (DRPC: {{ $labels.obj_name }}, Namespace: {{
    $labels.obj_namespace }}) is taking more than thrice the scheduled
    snapshot interval. This may cause data loss and a backlog of replication
    requests.
  alert_type: DisasterRecovery
```

- 警报：**VolumeSynchronizationDelay**

  影响

  Warning

  用途

  卷复制组占用的实际同步延迟是预期的同步延迟的两倍。

  YAML

```
alert: VolumeSynchronizationDela
expr: ramen_rpo_difference > 2 and ramen_rpo_difference < 3
for: 5s
labels:
  cluster: '{{ $labels.cluster }}'
  severity: critical
annotations:
  description: >-
    Syncing of volumes (DRPC: {{ $labels.obj_name }}, Namespace: {{
    $labels.obj_namespace }}) is taking more than twice the scheduled
    snapshot interval. This may cause data loss and a backlog of replication
    requests.
  alert_type: DisasterRecovery
```

第 7 章 灾难恢复故障排除

7.1. 对 METRO-DR 进行故障排除

7.1.1. 在故障切换后，有状态集应用程序会卡住

问题

在重新定位到首选集群时，DRPlacementControl 会一直报告 PROGRESSION 为 "MovingToSecondary"。
在以前的版本中，在 Kubernetes v1.23 之前，Kubernetes control plane 不会清理为 StatefulSets 创建的 PVC。此活动由集群管理员或管理 StatefulSets 的软件操作员保留。因此，当 Pod 被删除时，StatefulSet 的 PVC 会保持不变。这可防止 Ramen 将应用程序重新定位到首选集群。

解决方案

1. 如果工作负载使用 StatefulSets，且重新定位会卡住 PROGRESSION 为 "MovingToSecondary"，则运行：

```
$ oc get pvc -n <namespace>
```

2. 对于属于该 StatefulSet 的命名空间的每个绑定的 PVC，请运行

```
$ oc delete pvc <pvcname> -n namespace
```

删除所有 PVC 后，卷组复制组 (VRG) 过渡到 secondary，然后会被删除。

3. 运行以下命令

```
$ oc get drpc -n <namespace> -o wide
```

几分钟后，PROGRESSION 报告 "Completed" 并完成重新定位。

结果

工作负载重新定位到首选集群

BZ reference: [2118270]

7.1.2. DR 策略可保护同一命名空间中的所有应用程序

问题

虽然只有单个应用程序被选择供 DR 策略使用，但同一命名空间中的所有应用程序都会受到保护。这会导致 PVC，与多个工作负载的 **DRPlacementControl spec.pvcSelector** 匹配；或者，如果所有工作负载中没有选择器，复制管理可能会多次管理每个 PVC，并根据单独的 **DRPlacementControl** 操作造成数据崩溃或无效操作。

解决方案

标签 PVC 属于唯一工作负载，并使用所选标签作为 DRPlacementControl **spec.pvcSelector**，以忽略哪个 DRPlacementControl 保护和管理命名空间中的哪些 PVC 子集。无法使用用户界面为 DRPlacementControl 指定 **spec.pvcSelector** 字段，因此必须使用命令行删除并创建此类应用程序的 DRPlacementControl。

BZ 参考：[2128860]

7.1.3. 在应用程序故障时处于 Relocating 状态

问题

此问题可能会在执行故障转移和应用程序的故障后发生（所有节点或集群）。当执行故障恢复时，应用程序会一直处于 **Relocating** 状态，并显示**等待** PV 恢复完成。

解决方案

使用 S3 客户端或等同从 s3 存储清理重复的 PV 对象。仅保持接近故障转移或重新定位时间的时间戳的那一个。

BZ 参考： [2120201]

7.1.4. 重新定位或故障恢复可能会处于**初始**状态

问题

当主集群停机并回到在线时，**重新定位** 或 **故障恢复** 可能会处于 **Initiating** 状态。

解决方案

要避免这种情况，请关闭从旧的活跃 hub 到受管集群的所有访问。
另外，您可以在旧的活跃 hub 集群上缩减 ApplicationSet 控制器，然后再移动工作负载或处于清理阶段。

在旧的活跃 hub 中，使用以下命令缩减两个部署：

```
$ oc scale deploy -n openshift-gitops-operator openshift-gitops-operator-controller-manager --replicas=0

$ oc scale statefulset -n openshift-gitops openshift-gitops-application-controller --replicas=0
```

BZ 参考： [2243804]

7.2. 对 REGIONAL-DR 进行故障排除

7.2.1. rbd-mirror 守护进程健康状态

问题

当 mirror 服务 **::get_mirror_service_status** 调用 **Ceph** monitor 来获取 **rbd-mirror** 的服务状态，会有多种情况会报告 WARNING。
在网络连接网络连接后，**rbd-mirror** 守护进程健康状态在 **警告** 状态，同时两个受管集群间的连接都正常。

解决方案

在 toolbox 中运行以下命令并查找 **leader:false**

```
rbd mirror pool status --verbose ocs-storagecluster-cephblockpool | grep 'leader:'
```

如果您在输出中看到以下内容：

leader: false	这表示存在守护进程启动问题，最有可能的根本原因是由于问题可靠地连接到 secondary 集群。 临时解决方案：通过删除 pod 并验证它已重新调度到另一节点上，将 **rbd-mirror** pod 移到另一个节点。
leader: true 或没有输出	联系红帽支持。

BZ 参考： [2118627]

7.2.2. volsync-rsync-src pod 处于错误状态，因为它无法解析目标主机名

问题

VolSync 源 pod 无法解析 VolSync 目标 pod 的主机名。VolSync Pod 的日志在延长时间内显示错误消息，类似于以下日志片断。

```
$ oc logs -n busybox-workloads-3-2 volsync-rsync-src-dd-io-pvc-1-p25rz
```

输出示例

```
VolSync rsync container version: ACM-0.6.0-ce9a280
Syncing data to volsync-rsync-dst-dd-io-pvc-1.busybox-workloads-3-2.svc.clusterset.local:22 ...
ssh: Could not resolve hostname volsync-rsync-dst-dd-io-pvc-1.busybox-workloads-3-
2.svc.clusterset.local: Name or service not known
```

解决方案

在这两个节点上，重启 **submariner-lighthouse-agent**。

```
$ oc delete pod -l app=submariner-lighthouse-agent -n submariner-operator
```

7.2.3. 在恢复旧的主受管集群后，对 ApplicationSet 工作负载的清理和数据同步会卡住

问题

当 hub 集群失败时，基于 ApplicationSet 的工作负载部署到受管集群不会收集垃圾回收。它被恢复到一个备用的 hub 集群，而工作负载已切换到存活的受管集群。工作负载失败的集群，重新加入新的恢复的待机 hub。
使用区域 DRPolicy 进行 DR 保护的 ApplicationSets 开始触发 VolumeSynchronizationDelay 警报。另外，这种 DR 保护的工作负载无法切换到对等集群，或重新定位到对等集群，因为数据在两个集群之间没有同步。

解决方案

这个临时解决方案要求 **openshift-gitops** operator 可以拥有受管集群上孤立的工作负载资源，这些资源会从新的恢复的 hub 故障转移后重新加入 hub。要实现此目的，可以执行以下步骤：

1. 确定由 **openshift-gitops** 命名空间中的 hub 集群上的 ArgoCD ApplicationSet 资源使用的放置。

2. 在此字段中检查 ApplicationSet 的放置标签值：
spec.generators.clusterDecisionResource.labelSelector.matchLabels
这将是放置资源 < *placement-name> 的名称*

3. 确保 ApplicationSet 引用的 **Placement** 存在 **PlacemenDecision**。

```
$ oc get placementdecision -n openshift-gitops --selector cluster.open-cluster-
management.io/placement=<placement-name>
```

这会导致单个 **PlacementDecision** 将工作负载放在当前所需的故障转移集群中。

4. 为 ApplicationSet 创建新的 **PlacementDecision**，指向应清理的集群。
例如：

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/decision-group-index: "1" # Typically one higher
than the same value in the esisting PlacementDecision determined at step (2)
    cluster.open-cluster-management.io/decision-group-name: ""
    cluster.open-cluster-management.io/placement: cephfs-appset-busybox10-placement
  name: <placemen-name>-decision-<n> # <n> should be one higher than the existing
PlacementDecision as determined in step (2)
  namespace: openshift-gitops
```

5. 使用 status **子资源** 更新新创建的 **PlacementDecision**。

```
decision-status.yaml:
status:
  decisions:
  - clusterName: <managedcluster-name-to-clean-up> # This would be the cluster from
where the workload was failed over, NOT the current workload cluster
    reason: FailoverCleanup
```

```
$ oc patch placementdecision -n openshift-gitops <placemen-name>-decision-<n> --
patch-file=decision-status.yaml --subresource=status --type=merge
```

6. 监控并确保 ApplicationSet 的 Application 资源已放在所需的集群中

```
$ oc get application -n openshift-gitops  <applicationset-name>-<managedcluster-name-
to-clean-up>
```

在输出中，检查 SYNC STATUS 是否显示为 **Synced**，并且 HEALTH STATUS 显示为 **Healthy**。

7. 删除在第 3 步中创建的 PlacementDecision，以便 ArgoCD 可以垃圾回收 <managedcluster-name-to-clean-up> 上的工作负载资源

```
$ oc delete placementdecision -n openshift-gitops <placemen-name>-decision-<n>
```

受 DR 保护的 ApplicationSets （带有区域 DRPolicy）停止触发 **VolumeSynchronizationDelay** 警报。

BZ 参考： [2268594]

7.3. 使用 ARBITER 对 2 站点扩展集群进行故障排除

7.3.1. 在区恢复后恢复工作负载 pod 处于 ContainerCreating 状态

问题

执行完区失败和恢复后，工作负载 pod 有时会处于 **ContainerCreating** 状态，并显示以下错误：

- MountDevice 无法创建 newCsiDriverClient: 驱动程序名称 openshift-storage.rbd.csi.ceph.com，在注册的 CSI 驱动程序列表中找不到
- 卷 *<volume_name>* MountDevice 失败：rpc error: code = Aborted desc = an operation with the given Volume ID *<volume_id>* already exists
- 卷 *<volume_name>* MountVolume.SetUp 失败：rpc error: code = Internal desc = staging path *<path>* for volume *<volume_id>* is not a mountpoint

解决方案

如果工作负载 pod 遇到上述任何错误，请执行以下临时解决方案：

- 对于 **ceph-fs** 工作负载，处于 **ContainerCreating** 中：
  1. 重启调度卡住 pod 的节点
  2. 删除这些卡住的 pod
  3. 验证新 pod 是否正在运行
- 对于 **ceph-rbd** 工作负载，在 **ContainerCreating** 中，在一段时间后不会进行自助恢复
  1. 在调度卡住 pod 的节点中重启 csi-rbd 插件 pod
  2. 验证新 pod 是否正在运行