Red Hat OpenShift Data Foundation 4.10

使用 Advanced Cluster Management 为 OpenShift Data Foundation 配置 Metro-DR

开发者预览：设置带有 Metro-DR 功能的 OpenShift Data Foundation 的功能此解决方案是一个开发者技术预览功能，它不应该在生产环境中运行。

Last Updated: 2024-01-17

Red Hat OpenShift Data Foundation 4.10 使用 Advanced Cluster Management 为 OpenShift Data Foundation 配置 Metro-DR

开发者预览：设置带有 Metro-DR 功能的 OpenShift Data Foundation 的功能此解决方案是一个开发者技术预览功能，它不应该在生产环境中运行。

法律通告

摘要

本指南旨在详细说明使用 Advanced Cluster Management 部署 OpenShift Data Foundation 进行灾难恢复所需的步骤，以实现高度可用的存储基础架构。 Configuring OpenShift Data Foundation for Metro-DR with Advanced Cluster Management is a Developer Preview feature and is subject to Developer Preview support limitations. Developer Preview releases are not intended to be run in production environments and are not supported through the Red Hat Customer Portal case management system. If you need assistance with Developer Preview features, reach out to the ocs-devpreview@redhat.com mailing list and a member of the Red Hat Development Team will assist you as quickly as possible based on their availability and work schedules.

目录

让开源更具包容性

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。提供反馈：

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要提交更复杂的反馈，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 METRO-DR 简介

灾难恢复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。

Metro-DR 功能可以在同一地理区域内的站点间提供卷持久性数据和元数据复制。在公共云中，它们类似于防止可用性区域失败。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。这通常通过 Recovery Point Objective (RPO) 和 Recovery Time Objective (RTO) 代表。

- 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。Metro-DR 解决方案可确保您的 RPO 为零，因为数据将以同步的方式复制。
- 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？"

本指南的目的是为了详细详细说明自治区恢复(Metro-DR)的步骤和必要的命令，以便能够从一个 Red Hat OpenShift Container Platform 集群故障切换应用程序到另一个集群，然后将相同的应用程序恢复到原始主集群。在这种情况下，将使用 Red Hat Advanced Cluster Management (RHACM)创建或导入 RHOCP 集群，并且 ***RHOCP 集群小于 10 ms RTT 延迟之间存在距离限制***。

应用的持久存储将由外部 **红帽 Ceph 存储集群** 提供，该群集将在连接此存储群集的 RHOCP 实例在两个位置之间扩展。当网站中断时，需要具有存储监控器服务的仲裁节点（与部署 RHOCP 实例不同的位置）为 Red Hat Ceph Storage 集群建立仲裁。第三个位置具有 relaxed 延迟要求，它支持从连接到 RHOCP 实例的存储集群中最多 100 毫秒 RTT 延迟的值。

1.1. 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，用于在 OpenShift Container Platform 集群中提供应用程序和数据移动功能。

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，适用于公共云或私有云部署。

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 Data Foundation 堆栈通过提供 **csi-addons** 增强功能，以管理每个持久性卷声明镜像。

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 集群的应用状态关系
- 在应用程序状态变为对等集群时失败
- 将应用的状态重新定位到之前部署的集群

OpenShift DR 分为两个组件：

- **OpenShift DR Hub Operator** ：在 hub 集群上安装，以管理应用程序的故障转移和重新定位。
- **OpenShift DR Cluster Operator** ：安装在每个受管集群上，以管理应用程序的所有 PVC 的生命周期。

1.2. METRO-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 OpenShift Data Foundation 版本 4.10、RHCS 5 和 RHACM 最新版本配置和部署区域 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. 确保您满足每个 Metro-DR 要求，包括 RHACM operator（操作器）安装、创建或导入 OpenShift Container Platform 到 RHACM hub 和网络配置。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. 配置红帽 Ceph 存储扩展群集模式。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，请参阅配置 Red Hat Ceph Storage 扩展集群 。
4. 在 Primary 和 Secondary 受管集群中安装 OpenShift Data Foundation 4.10。请参阅在受管集群中安装 OpenShift Data Foundation。
5. 在 Hub 集群上安装 Openshift DR Hub Operator。请参阅在 Hub 集群上安装 OpenShift DR Hub Operator。
6. 配置受管和 Hub 集群。请参阅配置受管和 hub 集群。
7. 在 hub 集群上创建 DRPolicy 资源，用于在受管集群间部署、故障转移和重新定位工作负载。请参阅在 Hub 集群上创建灾难恢复策略。
8. 启用 OpenShift DR Cluster Operator 自动安装，并在受管集群中自动传输 S3 secret。具体步骤请参阅 启用 OpenShift DR 集群 Operator 的自动安装，并 在受管集群中启用 S3 secret 自动传输。
9. 使用 RHACM 控制台创建示例应用程序，用于测试故障转移和重定位测试。具体步骤，请参阅 创建示例应用程序、应用程序故障切换并在受管集群间重新定位应用程序。

第 2 章 启用 METRO-DR 的要求

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：

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

  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。
- 您必须有三个在它们之间具有网络可访问性的 OpenShift 集群：
  - 安装了 Advanced Cluster Management for Kubernetes(RHACM operator)和 OpenShift DR Hub 控制器的 **hub 集群**。
  - 安装 **OpenShift Data Foundation、OpenShift DR 集群控制器和应用的主要受管集群**。
  - 安装 **OpenShift Data Foundation、OpenShift DR 集群控制器和应用的辅助受管集群**。
- 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。
  - 部署完成后，使用 OpenShift 凭证登录到 RHACM 控制台。
  - 查找为 Advanced Cluster Manager 控制台创建的路由：

    ```
    $ oc get route multicloud-console -n open-cluster-management -o jsonpath --
    template="https://{.spec.host}/multicloud/clusters{'\n'}"
    ```

    输出示例：

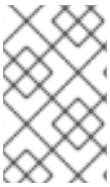
    ```
    https://multicloud-console.apps.perf3.example.com/multicloud/clusters
    ```

    使用 OpenShift 凭据登录后，您应该会看到导入的本地集群。
- 使用 RHACM 控制台导入或创建**主（Primary）受管集群**和**二级（Secondary）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。

第 3 章 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。有关更复杂的部署，请参阅 RHCS 5 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。

3.1. 硬件要求

如需有关部署 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
ceph7	DC3	MON

3.2. 软件要求

使用 **Red Hat Ceph Storage 5**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 中的知识库文章。

3.3. 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 **仲裁** 数据中心，这已测试的值为 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。

3.4. 节点部署前的要求

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

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：
   - **rhel-8-for-x86_64-baseos-rpms**
   - **rhel-8-for-x86_64-appstream-rpms**

```
subscription-manager repos --disable="*" --enable="rhel-8-for-x86_64-baseos-rpms" --
enable="rhel-8-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-8-x86_64-rpms** 和 **rhceph-5-tools-for-rhel-8-x86_64-rpms** 存储库：

```
subscription-manager repos --enable="ansible-2.9-for-rhel-8-x86_64-rpms" --
enable="rhceph-5-tools-for-rhel-8-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 验证用于部署红帽 Ceph 存储的主机名配置。

```
$ 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：

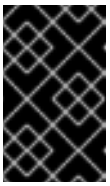
```
ceph1.example.domain.com
```

注：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

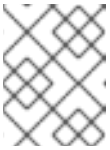
要运行 ansible playbook，您必须有 **ssh** 免密码访问配置 Red Hat Ceph Storage 集群的所有节点。确保配置的用户（如 **deployment-user**）具有可调用 **sudo** 命令的 root 特权，而无需输入密码。

   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
EOF
```



注意

作为清单文件的 [admin] 组配置的主机将 **cephadm** 将标记为 **_admin**，因此它们会在 bootstrap 过程中收到 admin ceph 密钥环。

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"
}
```

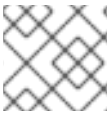
e. 运行以下 ansible playbook。

```
$ 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 Red Hat Ceph Storage **dnf** 存储库，并准备存储集群以进行引导。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**。

3.5. 使用 CEPHADM 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。



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 **cluster-spec.yaml**，将节点添加到 RHCS 集群，并为该服务应该运行下表 3.1 设置特定的标签。

```
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: fs_name
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. 检索从 bootstrap 节点配置的 RHCS 公共网络的 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：

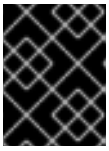
```
10.0.40.78
```

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。

```
$ 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：

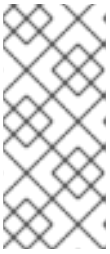
```
You can access the Ceph CLI with:

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

Please 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 osds 时出现一个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 rbdpool 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 服务处于活动状态，并且每个数据中心上具有一个服务。

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

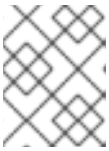
输出示例：

```
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。如需更多信息，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
```

第 4 章 配置 RED HAT CEPH STORAGE 扩展集群

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两个站点情况。

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，请参阅配置监控选举策略。

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，请参见 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 DC1
        step chooseleaf firstn 2 type host
        step emit
        step take DC2
        step chooseleaf firstn 2 type host
        step emit
}

# end crush map
```

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：

- **规则名称** ：
  - 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。
  - Value: **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。
  - Value: **10**
- **step take DC1**
  - Description：获取存储桶名称(DC1)，并开始迭代树。
- **step chooseleaf firstn 2 type host**
  - Description：选择给定类型的存储桶数量，本例中为位于 DC1 中的两个不同的主机。
- **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。
- **step take DC2**
  - Description：获取存储桶名称(DC2)，并开始迭代树。
- **step chooseleaf firstn 2 type host**
  - Description：选择给定类型的存储桶数量，本例中为两个位于 DC2 的不同主机。
- **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：

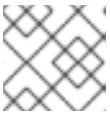
```
$ crushtool -c /etc/ceph/crushmap.txt -o /etc/ceph/crushmap2.bin
```

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。

第 5 章 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 必须首先安装在每个受管集群中，如下所示：

1. 在每个受管集群上安装最新的 OpenShift Data Foundation。
2. 安装 Operator 后，使用选项 **Connect with external storage platform** 创建 StorageSystem。具体步骤请参考以外部模式部署 OpenShift Data foundation。
3. 验证 OpenShift Data foundation 的部署是否成功：
   a. 使用以下命令在每个受管集群中：

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

   b. 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。

注意

另外，OpenShift Container Platform Web 控制台中还可在 OpenShift Container Platform Web 控制台中验证成功安装 OpenShift Data，方法是进入到 **Storage**，再单击 **Data Foundation**。

第 6 章 在 HUB 集群上安装 OPENSHIFT DR HUB OPERATOR

步骤

1. 在 Hub 集群中，导航到 OperatorHub 并使用 **OpenShift DR Hub Operator** 的搜索过滤器。
2. 按照屏幕说明，将操作器安装到 **openshift-dr-system** 项目中。
3. 使用以下命令验证 Operator Pod 是否处于 **Running** 状态：

```
$ oc get pods -n openshift-dr-system
```

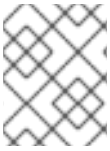
输出示例：

```
NAME                                READY   STATUS    RESTARTS   AGE
ramen-hub-operator-898c5989b-96k65  2/2     Running   0          4m14s
```

第 7 章 配置受管和 HUB 集群

7.1. 配置 S3 端点之间的 SSL 访问

配置 **s3 端点**之间的网络(SSL)访问，以便元数据可以在 **MCG 对象存储桶**中存储，使用安全传输协议并在 **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. 在**主受管集群**, **二级受管集群**, 和 **Hub cluster** 上创建一个新的 **ConfigMap** 用于保存远程集群的证书捆绑，其文件名为 **cm-clusters-crt.yaml**。

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-----
```

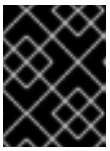
```
    <copy contents of cert2 from secondary.crt here>
    -----END CERTIFICATE-----

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

4. 在**主受管集群**、**二级受管集群**和 **Hub 集群**上创建 ConfigMap 文件。

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

输出示例：

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



重要

对于 Hub 集群，使用 **DRPolicy** 资源验证对象存储桶的访问权限，必须在 Hub 集群上创建相同的 **ConfigMap cm-clusters-crt.yaml**。

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
```

7.2. 创建对象存储桶和 S3STOREPROFILE

OpenShift DR S3 存储来存储受管集群中工作负载的相关集群数据，并在故障转移或重新定位操作期间编配工作负载的恢复。这些说明可用于使用多云对象网关(MCG)创建所需的对象存储桶。安装 OpenShift Data Foundation 后，MCG 应已被安装。

步骤

1. 创建 MCG 对象存储桶或 OBC，以便在 Primary 和 Secondary 受管集群上存储持久性卷元数据。

   a. 将以下 YAML 文件复制到名为 **odrbucket.yaml** 的文件。

```
apiVersion: objectbucket.io/v1alpha1
kind: ObjectBucketClaim
metadata:
  name: odrbucket
  namespace: openshift-storage
spec:
  generateBucketName: "odrbucket"
  storageClassName: openshift-storage.noobaa.io
```

b. 在**主受管集群**和**次受管集群**上创建 MCG bucket **odrbucket**。

```
$ oc create -f odrbucket.yaml
```

输出示例：

```
objectbucketclaim.objectbucket.io/odrbucket created
```

2. 使用以下命令，提取每个受管集群的 **odrbucket** OBC 访问密钥作为其 *base-64 编码* 值。

```
$ oc get secret odrbucket -n openshift-storage -o jsonpath='{.data.AWS_ACCESS_KEY_ID}
{"\n"}'
```

输出示例：

```
cFpIYTZWN1NhemJjbEUyWlpwN1E=
```

3. 使用以下命令，提取每个受管集群的 **odrbucket** OBC secret 密钥作为其 *base-64 编码* 值。

```
$ oc get secret odrbucket -n openshift-storage -o
jsonpath='{.data.AWS_SECRET_ACCESS_KEY}{"\n"}'
```

输出示例：

```
V1hUSnMzZUoxMHRRTXdGMU9jQXRmUlAyMmd5bGwwYjNvMHprZVhtNw==
```

重要

必须针对 **主受管集群**和**二级受管集群**上的 **odrbucket** OBC 检索 access key 和 secret key。

7.3. 为 MULTICLOUD OBJECT GATEWAY 对象存储桶创建 S3 SECRET

现在，在上一节中为对象存储桶提取了必要的信息，必须在 **Hub 集群**上创建新的 **Secret**。这些新 Secret 将为 Hub 集群上的两个受管集群存储 MCG 对象存储桶访问密钥和 secret 密钥。

流程

1. 将**主受管集群**的以下 S3 secret YAML 格式复制到名为 **odr-s3secret-primary.yaml** 的文件。

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <primary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <primary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-primary
  namespace: openshift-dr-system
```

2. 在 **Hub 集群**上创建此 secret。

```
$ oc create -f odr-s3secret-primary.yaml
```

输出示例：

```
secret/odr-s3secret-primary created
```

3. 将**二级受管集群**的以下 S3 secret YAML 格式复制到名为 **odr-s3secret-secondary.yaml** 的文件中。

```
apiVersion: v1
data:
  AWS_ACCESS_KEY_ID: <secondary cluster base-64 encoded access key>
  AWS_SECRET_ACCESS_KEY: <secondary cluster base-64 encoded secret access key>
kind: Secret
metadata:
  name: odr-s3secret-secondary
  namespace: openshift-dr-system
```

4. 在 **Hub 集群**上创建此 secret。

```
$ oc create -f odr-s3secret-secondary.yaml
```

输出示例：

```
secret/odr-s3secret-secondary created
```

重要

access key 和 secret key 的值必须是**以 base-64 编码**。密钥编码的值在 prior 部分中检索。

7.4. 配置 OPENSHIFT DR HUB OPERATOR S3STOREPROFILES

要查找 MCG 的 s3CompatibleEndpoint 或路由，请在主受管集群和二级受管集群中执行以下命令：

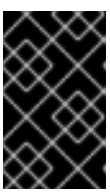
步骤

a. 使用以下命令，搜索外部 S3 端点 **s3CompatibleEndpoint** 或路由来为每个受管集群上的 MCG 进行 MCG。

```
$ oc get route s3 -n openshift-storage -o jsonpath --template="https://{.spec.host}{'\n'}"
```

输出示例：

```
https://s3-openshift-storage.apps.perf1.example.com
```



重要

唯一的 s3CompatibleEndpoint 路由或 **s3-openshift-storage.apps.<primary clusterID>.<baseDomain>** 和 **s3-openshift-storage.apps.<secondary clusterID>.<baseDomain>** 必须分别为**主受管集群**和**次受管集群**获取。

b. 搜索 **odrbucket** OBC 准确存储桶名称。

```
$ oc get configmap odrbucket -n openshift-storage -o jsonpath='{.data.BUCKET_NAME}
{"\n"}'
```

输出示例：

```
odrbucket-2f2d44e4-59cb-4577-b303-7219be809dcd
```

重要

唯一的 **s3Bucket** 名称 *odrbucket-<your value1>* 和 *odrbucket-<your value2>* 必须分别在**主受管集群**和**次受管集群**上获得。

c. 修改 Hub 集群上的 ConfigMap **ramen-hub-operator-config** 以添加新内容。

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

d. 将以下新内容从 **s3StoreProfiles** 开始添加到 **Hub 集群**的 ConfigMap 中。

```
[...]
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    kind: RamenConfig
[...]
    ramenControllerType: "dr-hub"
    ### Start of new content to be added
    s3StoreProfiles:
    - s3ProfileName: s3-primary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<primary clusterID>.
<baseDomain>
      s3Region: primary
      s3Bucket: odrbucket-<your value1>
      s3SecretRef:
        name: odr-s3secret-primary
        namespace: openshift-dr-system
    - s3ProfileName: s3-secondary
      s3CompatibleEndpoint: https://s3-openshift-storage.apps.<secondary clusterID>.
<baseDomain>
      s3Region: secondary
      s3Bucket: odrbucket-<your value2>
      s3SecretRef:
        name: odr-s3secret-secondary
        namespace: openshift-dr-system
[...]
```

第 8 章 在 HUB 集群上创建灾难恢复策略

OpenShift DR 使用 RHACM hub 集群上的 Disaster Recovery Policy(DRPolicy)资源（集群范围）来部署、故障转移和重新定位受管集群中的工作负载。

先决条件

- 确保有两个集群。
- 确保为策略中的每个集群分配 S3 配置文件名称，该名称使用 OpenShift DR 集群和 hub 操作器的 ConfigMap 配置。

流程

1. 在 **Hub 集群**中，进入 **openshift-dr-system** 项目中的 Installed Operators，然后点 **OpenShift DR Hub Operator**。您应该会看到两个可用的 API，即 **DRPolicy** 和 **DRPlacementControl**。
2. 为 DRPolicy 点 **Create instance** 并点击 **YAML 视图**。
3. 将 <cluster1> 和 <cluster2> 替换为 **RHACM** 中受管集群的正确名称后，将以下 YAML 保存到名为 **drpolicy.yaml** 的文件中。将 <string_value> 替换为任何值 (例如 metro)。

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: odr-policy
spec:
  drClusterSet:
  - name: <cluster1>
    region: <string_value>
    s3ProfileName: s3-primary
    clusterFence: Unfenced
  - name: <cluster2>
    region: <string_value>
    s3ProfileName: s3-secondary
    clusterFence: Unfenced
```

注意

不需要指定一个命名空间来创建此资源，因为 DRPolicy 是一个集群范围的资源。

4. 将唯一 **drpolicy.yaml** 文件的内容复制到 YAML 视图中。您必须完全替换原始的内容。
5. 在 YAML 视图屏幕上点 **Create**。
6. 要验证 **DRPolicy** 是否已成功创建，并且可以使用之前创建的 Secret 访问 MCG 对象存储桶，请在 **Hub 集群**上运行这个命令：

```
$ oc get drpolicy odr-policy -n openshift-dr-system -o jsonpath='{.status.conditions[].reason}
{"\n"}'
```

输出示例：

```
Succeeded
```

第 9 章 启用自动安装 OPENSHIFT DR 集群 OPERATOR

在成功创建 DRPolicy 后，**OpenShift DR Cluster operator** 可以安装到主受管集群，并在 **openshift-dr-system** 命名空间中安装二级受管集群。

流程

1. 编辑 Hub 集群上的 ConfigMag **ramen-hub-operator-config**，并将 **deploymentAutomationEnabled=false** 的值改为 **true**，如下所示：

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
  [...]
    drClusterOperator:
      deploymentAutomationEnabled: true  ## <-- Change value to "true" if it is set to "false"
      channelName: stable-4.10
      packageName: odr-cluster-operator
      namespaceName: openshift-dr-system
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
[...]
```

2. 验证在**主受管集群**和**二级受管集群**中是否安装否成功，使用以下命令：

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

输出示例：

```
NAME                                                                                  DISPLAY                       VERSION
REPLACES   PHASE
clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.10.0   Openshift DR
Cluster Operator   4.10.0              Succeeded

NAME                                         READY   STATUS    RESTARTS   AGE
pod/ramen-dr-cluster-operator-5564f9d669-f6lbc   2/2     Running   0          5m32s
```

您还可以进入每个受管集群的 OperatorHub，并确认是否安装了 **OpenShift DR Cluster Operator**。

第 10 章 启用自动向受管集群传输 S3SECRETS

按照以下步骤，将 s3Secrets 自动传送到所需的 OpenShift DR 集群组件。它使用访问 OpenShift DR 配置映射中 s3Profiles 所需的 s3Secrets 更新 OpenShift DR 集群命名空间。

步骤

1. 编辑 Hub 集群上的 ConfigMag **ramen-hub-operator-config** 以添加 **s3SecretDistributionEnabled=true**，如下所示：

```
$ oc edit configmap ramen-hub-operator-config -n openshift-dr-system
```

```
apiVersion: v1
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    drClusterOperator:
      deploymentAutomationEnabled: true
      s3SecretDistributionEnabled: true  ## <-- Add to enable automatic transfer of s3secrets
      catalogSourceName: redhat-operators
      catalogSourceNamespaceName: openshift-marketplace
      channelName: stable-4.10
      clusterServiceVersionName: odr-cluster-operator.v4.10.0
      namespaceName: openshift-dr-system
      packageName: odr-cluster-operator
[...]
```

2. 在两个受管集群中运行此命令，验证 secret 的传输是否成功。

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

输出示例：

```
8b3fb9ed90f66808d988c7edfa76eba35647092   Opaque       2      11m
af5f82f21f8f77faf3de2553e223b535002e480   Opaque       2      11m
```

第 11 章 创建示例应用程序

为了测试从主受管集群到二级受管集群的**故障切换**，我们需要一个简单的应用程序。使用名为 **busybox** 的示例应用作为示例。

步骤

1. 在 **Hub 集群**中为 **busybox** 示例应用程序创建一个**命名空间或项目**。

```
$ oc new-project busybox-sample
```

注意

如果需要，可以使用除 **busybox-sample** 以外的不同项目名称。在通过 Advanced Cluster Manager 控制台部署示例应用程序时，确保使用与此步骤中创建相同的项目名称。

2. 创建 **DRPlacementControl** 资源
DRPlacementControl 是一个 API，在 Hub 集群上安装 OpenShift DR Hub Operator 后可用。它基本上是一个 Advanced Cluster Manager PlacementRule 协调器，它根据作为 DRPolicy 一部分的集群中的数据可用性来编排放置决策。

   a. 在 Hub 集群中，导航到 **busybox-sample** 项目中的 Installed Operators，再单击 **OpenShift DR Hub Operator**。您应该会看到两个可用的 API，DRPolicy 和 DRPlacementControl。

   b. 为 **DRPlacementControl** 创建一个实例，然后进入 YAML 视图。确保已选中 **busybox-sample** 项目。

   c. 在将 *<cluster1>* 替换为 Advanced Cluster Manager 中的受管集群的正确名称后，将以下 YAML 复制到名为 **busybox-drpc.yaml** 的文件。

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPlacementControl
metadata:
  labels:
    app: busybox-sample
  name: busybox-drpc
spec:
  drPolicyRef:
    name: odr-policy
  placementRef:
    kind: PlacementRule
    name: busybox-placement
  preferredCluster: <cluster1>
  pvcSelector:
    matchLabels:
      appname: busybox
```

   d. 将唯一 **busybox-drpc.yaml** 文件的内容复制到 YAML 视图（完全替换原始内容）。

   e. 在 YAML 视图屏幕上点 **Create**。
   您还可以使用以下 CLI 命令创建此资源：

```
$ oc create -f busybox-drpc.yaml -n busybox-sample
```

输出示例：

```
drplacementcontrol.ramendr.openshift.io/busybox-drpc created
```

重要

此资源必须在 **busybox-sample** 命名空间（或您之前创建的任何命名空间）中创建。

3. 创建 **Placement Rule** 资源，以定义可以部署资源模板的目标集群。使用放置规则促进应用程序的多集群部署。

   a. 将以下 YAML 复制到名为 **busybox-placementrule.yaml** 的文件中。

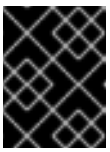
```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  labels:
    app: busybox-sample
  name: busybox-placement
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterReplicas: 1
  schedulerName: ramen
```

   b. 为 **busybox-sample** 应用创建 Placement Rule 资源。

```
$ oc create -f busybox-placementrule.yaml -n busybox-sample
```

输出示例：

```
placementrule.apps.open-cluster-management.io/busybox-placement created
```



重要

此资源必须在 **busybox-sample** 命名空间（或您之前创建的任何命名空间）中创建。

4. 使用 RHACM 控制台创建**示例应用程序**

   a. 如果尚未登录，请使用您的 OpenShift 凭据登录 RHACM 控制台。

```
$ oc get route multicloud-console -n open-cluster-management -o jsonpath --
template="https://{.spec.host}/multicloud/applications{'\n'}"
```

输出示例：

```
https://multicloud-console.apps.perf3.example.com/multicloud/applications
```

b. 导航到 **Applications**，再单击 **Create application**。

c. 选择类型 **Subscription**。

d. 输入应用程序 **Name** （如 **busybox**）和 **Namespace** （如 **busybox-sample**）。

e. 在 Repository location for resources 部分中，选择 **Repository type Git**。

f. 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/RamenDR/ocm-ramen-samples**，其中 **Branch** 是 **main**，**Path** 是 **busybox-odr-metro**。

g. 将表单向下滚动到 **Select clusters to deploy** 部分，然后点击 **Select an existing placement configuration**。

h. 从下拉列表中选择 **现有 Placement Rule** （如 **busybox-placement**）。

i. 点击 **Save**。
在下一屏幕上，滚动到底部。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。

5. 验证示例应用程序部署和复制。
现在 **busybox** 应用程序已部署到首选的集群（在 DRPlacementControl 中指定），可以验证部署。

a. 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   1/1     Running   0          6m

NAME                                  STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS                AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-a56c138a-a1a9-4465-927f-
af02afbbff37   1Gi        RWO            ocs-storagecluster-ceph-rbd   6m
```

b. 验证是否也为 **busybox** PVC 创建了复制资源。

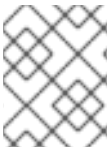
```
$ oc get volumereplicationgroup -n busybox-sample
```

输出示例：

```
NAME                                              AGE
volumereplicationgroup.ramendr.openshift.io/busybox-drpc   6m
```

11.1. 删除示例应用程序

您可以使用 RHACM 控制台删除示例 application **busybox**。

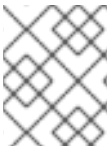


注意

在故障转移和故障恢复（重复）测试完成以及应用程序已可以从 RHACM 和受管集群中删除之前，不应执行示例应用程序的说明。

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 应用时，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 控制台删除资源外，在删除 **busybox** 应用后，还必须立即删除 **DRPlacementControl**。
   a. 登录到 Hub 集群的 OpenShift Web 控制台，再进入为 **busybox-sample** 项目安装的 Operators。
   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。也可以使用 CLI，将 **DRPlacementControl** 资源删除到应用命名空间中。

第 12 章 受管集群之间的应用程序故障切换

本节介绍如何故障转移 busybox 示例应用程序。Metro-DR 的故障转移方法是基于应用程序。以这种方式保护的每个应用都必须具有对应的 **DRPlacementControl** 资源，并在应用程序**命名空间**中创建 **PlacementRule** 资源，如 Create Sample Application for DR 测试部分所示。

流程

1. 创建 **NetworkFence** 资源并启用 **隔离**。
   指定执行网络隔离操作的 CIDR 块或 IP 地址列表。在我们的情形中，这将是集群中每个 OpenShift 节点的 EXTERNAL-IP，需要使用外部 RHCS 集群进行隔离。

   a. 执行此命令以获取**主受管集群**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
   ```

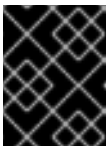
   注意

   在出现站点中断前，收集所有 OpenShift 节点的当前 IP 地址。最佳实践是创建 **NetworkFence** YAML 文件，并将其可用并最新以用于灾难恢复事件。

   所有节点的 IP 地址将添加到 **NetworkFence** 示例资源中，如下所示。这个示例是 6 个节点，但集群中可能存在多个节点。

   ```
   apiVersion: csiaddons.openshift.io/v1alpha1
   kind: NetworkFence
   metadata:
     name: network-fence-<cluster1>
   spec:
     driver: openshift-storage.rbd.csi.ceph.com
     cidrs:
       - <IP_Address1>/32
       - <IP_Address2>/32
       - <IP_Address3>/32
       - <IP_Address4>/32
       - <IP_Address5>/32
       - <IP_Address6>/32
       [...]
     secret:
       name: rook-csi-rbd-provisioner
       namespace: openshift-storage
     parameters:
       clusterID: openshift-storage
   ```

b. 对于以上 YAML 文件示例，修改 IP 地址，并将正确的 *<cluster1>* 提供给在 **RHACM** 中主受管集群的 RHACM 中的集群名称。把它保存到名为 **network-fence-<cluster1>.yaml** 的文件中。



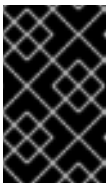
重要

在故障转移之前，必须从当前运行该应用的相对受管集群创建 **NetworkFence**。在本例中，这是**二级受管集群**。

```
$ oc create -f network-fence-<cluster1>.yaml
```

输出示例：

```
networkfences.csiaddons.openshift.io/network-fence-ocp4perf1 created
```



重要

创建 **NetworkFence** 后，所有应用程序与 OpenShift Data Foundation 存储的通信都将失败，一些 Pod 将处于不健康状态（例如：CreateContainerError, CrashLoopBackOff）。

c. 在与创建 **NetworkFence** 的同一集群中，验证其状态是否为 Succeeded。Modify <cluster1> to be correct.

```
export NETWORKFENCE=network-fence-<cluster1>
oc get networkfences.csiaddons.openshift.io/$NETWORKFENCE -n openshift-dr-system
-o jsonpath='{.status.result}{"\n"}'
```

输出示例：

```
Succeeded
```

2. 为**隔离**的集群修改 **DRPolicy**。

a. 编辑 Hub 集群上的 **DRPolicy**，并将 *<cluster1>*（例如： ocp4perf1）从 **Unfenced** 变为 **ManuallyFenced**。

```
$ oc edit drpolicy odr-policy
```

输出示例：

```
[...]
spec:
  drClusterSet:
  - clusterFence: ManuallyFenced  ## <-- Modify from Unfenced to ManuallyFenced
    name: ocp4perf1
    region: metro
    s3ProfileName: s3-primary
  - clusterFence: Unfenced
    name: ocp4perf2
```

```
    region: metro
    s3ProfileName: s3-secondary
[...]
```

输出示例：

```
drpolicy.ramendr.openshift.io/odr-policy edited
```

b. 验证 Hub 集群中的 **DRPolicy** 状态已针对 **Primary managed cluster** 改为 **Fenced**。

```
$ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
```

输出示例：

```
drClusters:
  ocp4perf1:
    status: Fenced
    string: ocp4perf1
  ocp4perf2:
    status: Unfenced
    string: ocp4perf2
```

3. 将 **DRPlacementControl** 修改为 **failover**

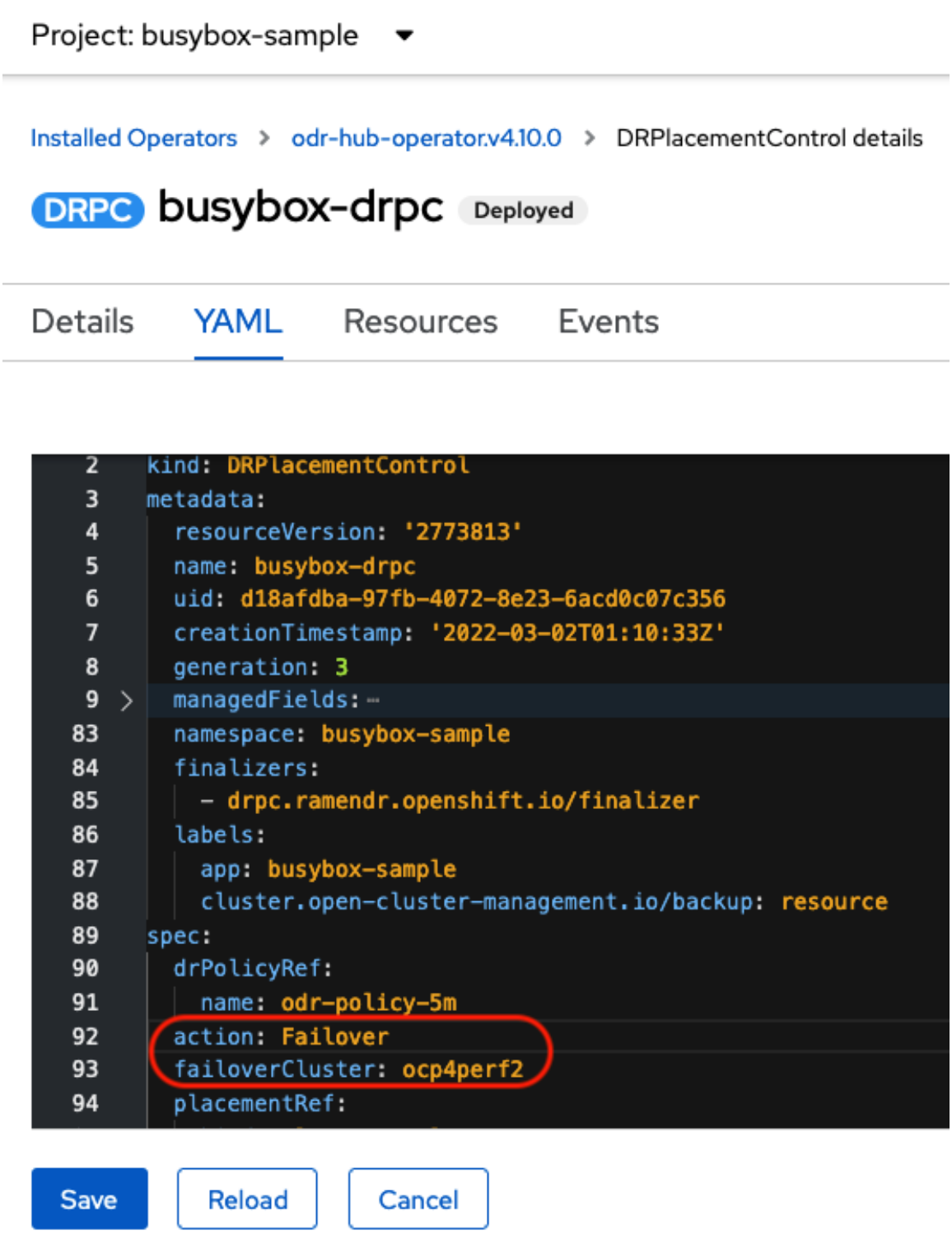
a. 在 Hub 集群中，导航到 Installed Operators，然后点 **Openshift DR Hub Operator**。

b. 单击 **DRPlacementControl** 选项卡。

c. 单击 DRPC **busybox-drpc**，然后单击 YAML 视图。

d. 添加 **action** 和 **failoverCluster** 详情，如下方屏幕截图所示。**failoverCluster** 应该是第二个受管集群的 ACM 集群名称。

DRPlacementControl 添加 Failover 操作



e. 点击 **Save**。

4. 验证 application **busybox** 是否现在在次受管集群中运行，即 YAML 文件中指定的故障转移集群 **ocp4perf2**。

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

输出示例：

```
NAME          READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          35s

NAME                                STATUS   VOLUME                                     CAPACITY   ACCESS
```

```
MODES   STORAGECLASS                 AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-666b74cfa1bb
5Gi        RWO            ocs-storagecluster-ceph-rbd   35s
```

5. 验证 **busybox** 不再在主受管集群上运行。

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

输出示例：

```
No resources found in busybox-sample namespace.
```

重要

请注意，在发行注记的已知问题一节中记录的 Metro-DR 问题。

第 13 章 在受管集群间重新定位应用程序

重定位操作与故障转移非常相似。重新定位基于应用程序，并使用 **DRPlacementControl** 触发重新定位。failback 的主要区别在于，应用程序在 failoverCluster 上缩减，因此不需要创建 **NetworkFence**。

流程

1. 删除 **NetworkFence** 资源并禁用**隔离** 。
   在故障恢复或重定位操作之前，必须成功删除主受管集群的 **NetworkFence**。

   a. 在 **二级受管集群**中执行此命令，并修改 *<cluster1>* 来使用上一节中创建的 **NetworkFence** YAML 的正确文件名。

   ```
   $ oc delete -f network-fence-<cluster1>.yaml
   ```

   输出示例：

   ```
   networkfence.csiaddons.openshift.io "network-fence-ocp4perf1" deleted
   ```

   b. 重新引导被**隔离**的 OpenShift Container Platform 节点。
   此步骤是必需的，因为以前的隔离集群上的一些应用程序 Pod（本例中为 Primary managed cluster）处于不健康状态（如 CreateContainerError, CrashLoopBackOff）。这可以通过一次重启所有 worker OpenShift 节点来轻松地修复。

   注意

   OpenShift Web 控制台仪表板和 **Overview** 页面也可用于评估应用程序和外部存储的健康状况。详细的 OpenShift Data Foundation 仪表板可通过 **Storage → Data Foundation** 找到。

   c. 在所有 OpenShift 节点重新引导并且处于 **Ready** 状态后，在**主受管集群**上运行此命令，以验证所有 Pod 都处于健康状态。此查询的输出应该为零个 Pod。

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

   输出示例：

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

   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。

2. 将 **DRPolicy** 修改为 **Unfenced** 状态。
   要让 ODR HUB operator 知道 **NetworkFence** 已为主受管集群删除，需要为新创建的 **Unfenced** 集群修改 **DRPolicy**。

   a. 编辑 Hub 集群上的 **DRPolicy**，并将 *<cluster1>*（例如 **ocp4perf1**）从 **manuallyFenced** 改为 **Unfenced**。

```
$ oc edit drpolicy odr-policy
```

输出示例：

```
[...]
spec:
  drClusterSet:
  - clusterFence: Unfenced  ## <-- Modify from ManuallyFenced to Unfenced
    name: ocp4perf1
    region: metro
    s3ProfileName: s3-primary
  - clusterFence: Unfenced
    name: ocp4perf2
    region: metro
    s3ProfileName: s3-secondary
[...]
```

输出示例：

```
drpolicy.ramendr.openshift.io/odr-policy edited
```

b. 验证 **Hub 集群**中的 **DRPolicy** 的状态是否已为**主受管集群**更改为 **Unfenced**。

```
$ oc get drpolicies.ramendr.openshift.io odr-policy -o yaml | grep -A 6 drClusters
```

输出示例：

```
  drClusters:
    ocp4perf1:
      status: Unfenced
      string: ocp4perf1
    ocp4perf2:
      status: Unfenced
      string: ocp4perf2
```

3. 将 **DRPlacementControl** 修改为 **failback**

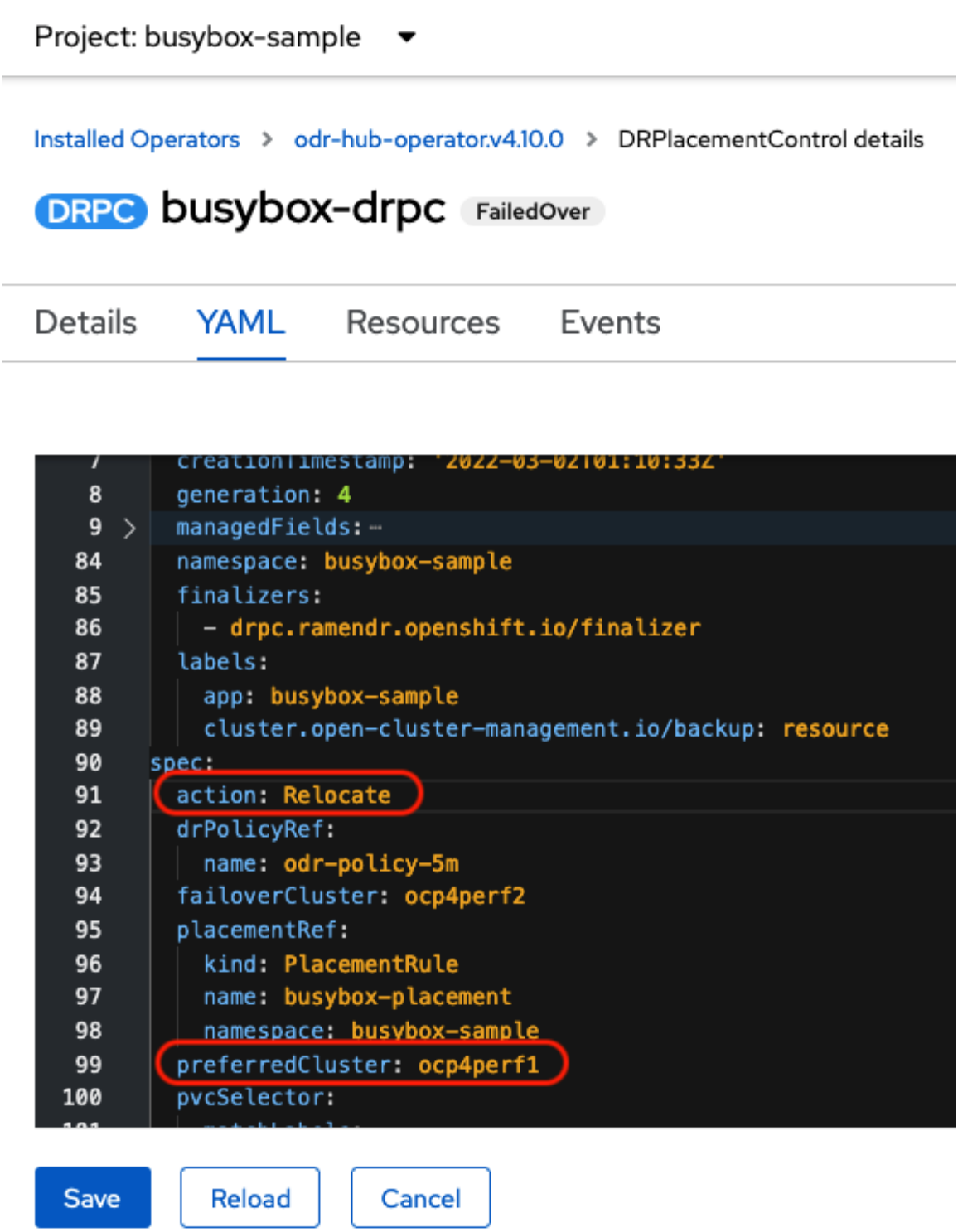
a. 在 Hub 集群中，导航到 Installed Operators，然后点 **Openshift DR Hub Operator**。

b. 单击 **DRPlacementControl** 选项卡。

c. 单击 DRPC **busybox-drpc**，然后单击 YAML 视图。

d. 将 **action** 改为 **Relocate**。

DRPlacementControl 修改重新分配的操作



e. 点击 **Save**。

f. 验证 **busybox** 应用程序已在主受管集群中运行。故障转移是在 YAML 文件中指定的 **preferredCluster ocp4perf1**，它是故障转移操作前应用程序运行的位置。

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

输出示例：

```
NAME        READY   STATUS    RESTARTS   AGE
pod/busybox   1/1     Running   0          60s

NAME                                STATUS   VOLUME                                     CAPACITY
```

```
ACCESS MODES   STORAGECLASS                 AGE
persistentvolumeclaim/busybox-pvc   Bound    pvc-79f2a74d-6e2c-48fb-9ed9-
666b74cfa1bb   5Gi        RWO           ocs-storagecluster-ceph-rbd   61s
```

g. 验证 **busybox** 是否在第二个受管集群中运行。busybox 应用程序不应在此受管集群上运行。

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

输出示例：

```
No resources found in busybox-sample namespace.
```

重要

请注意，在发行注记的已知问题一节中记录的 Metro-DR 问题。