Red Hat OpenShift Data Foundation 4.10

Configuring OpenShift Data Foundation for Metro-DR with Advanced Cluster Management

DEVELOPER PREVIEW: Instructions about setting up OpenShift Data Foundation with Metro-DR capabilities. This solution is a Developer Preview feature and is not intended to be run in production environments.

Last Updated: 2022-09-22

Red Hat OpenShift Data Foundation 4.10 Configuring OpenShift Data Foundation for Metro-DR with Advanced Cluster Management

DEVELOPER PREVIEW: Instructions about setting up OpenShift Data Foundation with Metro-DR capabilities. This solution is a Developer Preview feature and is not intended to be run in production environments.

Legal Notice

Abstract

The intent of this solution guide is to detail the steps necessary to deploy OpenShift Data Foundation for disaster recovery with Advanced Cluster Management to achieve a highly available storage infrastructure. 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.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message .

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Do let us know how we can make it better. To give feedback:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. In the **Component** section, choose **documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

CHAPTER 1. INTRODUCTION TO METRO-DR

Disaster recovery is the ability to recover and continue business critical applications from natural or human created disasters. It is a component of the overall business continuance strategy of any major organization as designed to preserve the continuity of business operations during major adverse events.

Metro-DR capability provides volume persistent data and metadata replication across sites that are in the same geographical area. In the public cloud these would be similar to protecting from an Availability Zone failure. Metro-DR ensures business continuity during the unavailability of a data center with no data loss. This is usually expressed at Recovery Point Objective (RPO) and Recovery Time Objective (RTO).

- RPO is a measure of how frequently you take backups or snapshots of persistent data. In practice, the RPO indicates the amount of data that will be lost or need to be reentered after an outage. Metro-DR solution ensures your RPO is zero because data is replicated in a synchronous fashion.
- RTO is the amount of downtime a business can tolerate. The RTO answers the question, "How long can it take for our system to recover after we were notified of a business disruption?"

The intent of this guide is to detail the Metro Disaster Recovery (Metro-DR) steps and commands necessary to be able to failover an application from one Red Hat OpenShift Container Platform cluster to another and then failback the same application to the original primary cluster. In this case the RHOCP clusters will be created or imported using Red Hat Advanced Cluster Management (RHACM) and have ***distance limitations between the RHOCP clusters of less than 10 ms RTT latency***

The persistent storage for applications will be provided by an external **Red Hat Ceph Storage**cluster stretched between the two locations with the RHOCP instances connected to this storage cluster. An arbiter node with a storage monitor service will be required at a third location (different location than where RHOCP instances are deployed) to establish quorum for the Red Hat Ceph Storage cluster in the case of a site outage. The third location has relaxed latency requirements, which supports values as high up to 100 ms RTT latency from the storage cluster connected to the RHOCP instances.

1.1. COMPONENTS OF METRO-DR SOLUTION

Metro-DR is composed of Red Hat Advanced Cluster Management for Kubernetes, Red Hat Ceph Storage and OpenShift Data Foundation components to provide application and data mobility across OpenShift Container Platform clusters.

Red Hat Advanced Cluster Management for Kubernetes

Red Hat Advanced Cluster Management (RHACM) provides the ability to manage multiple clusters and application lifecycles. Hence, it serves as a control plane in a multi-cluster environment.

RHACM is split into two parts:

- RHACM Hub: components that run on the multi-cluster control plane
- Managed clusters: components that run on the clusters that are managed

For more information about this product, see RHACM documentation and the RHACM "Manage Applications" documentation.

Red Hat Ceph Storage

Red Hat Ceph Storage is a massively scalable, open, software-defined storage platform that combines the most stable version of the Ceph storage system with a Ceph management platform, deployment utilities, and support services. It significantly lowers the cost of storing enterprise data and helps organizations manage exponential data growth. The software is a robust and modern petabyte-scale storage platform for public or private cloud deployments.

OpenShift Data Foundation

OpenShift Data Foundation provides the ability to provision and manage storage for stateful applications in an OpenShift Container Platform cluster. It is backed by Ceph as the storage provider, whose lifecycle is managed by Rook in the OpenShift Data Foundation component stack and Ceph-CSI provides the provisioning and management of Persistent Volumes for stateful applications.

OpenShift Data Foundation stack is enhanced with the ability to provide **csi-addons** to manage per Persistent Volume Claim mirroring.

OpenShift DR

OpenShift DR is a disaster recovery orchestrator for stateful applications across a set of peer OpenShift clusters which are deployed and managed using RHACM and provides cloud-native interfaces to orchestrate the life-cycle of an application's state on Persistent Volumes. These include:

- Protecting an application state relationship across OpenShift clusters
- Failing over an application's state to a peer cluster
- Relocate an application's state to the previously deployed cluster

OpenShift DR is split into two components:

- **OpenShift DR Hub Operator**: Installed on the hub cluster to manage failover and relocation for applications.
- **OpenShift DR Cluster Operator**: Installed on each managed cluster to manage the lifecycle of all PVCs of an application.

1.2. METRO-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Metro-DR capabilities using OpenShift Data Foundation version 4.10, RHCS 5 and RHACM latest version across two distinct OpenShift Container Platform clusters. In addition to two managed clusters, a third OpenShift Container Platform cluster will be required to deploy the Advanced Cluster Management.

To configure your infrastructure, perform the below steps in the order given:

1. Ensure you meet each of the Metro-DR requirements which includes RHACM operator installation, creation or importing of OpenShift Container Platform into RHACM hub and network configuration. See Requirements for enabling Metro-DR.
2. Ensure you meet the requirements for deploying Red Hat Ceph Storage stretch cluster with arbiter. See Requirements for deploying Red Hat Ceph Storage .
3. Configure Red Hat Ceph Storage stretch cluster mode. For instructions on enabling Ceph cluster on two different data centers using stretched mode functionality, see Configuring Red Hat Ceph Storage stretch cluster.
4. Install OpenShift Data Foundation 4.10 on Primary and Secondary managed clusters. See Installing OpenShift Data Foundation on managed clusters .

5. Install the Openshift DR Hub Operator on the Hub cluster. See Installing OpenShift DR Hub Operator on Hub cluster.

6. Configure the managed and Hub cluster. See Configuring managed and hub clusters .

7. Create the DRPolicy resource on the hub cluster which is used to deploy, failover, and relocate the workloads across managed clusters. See Creating Disaster Recovery Policy on Hub cluster .

8. Enable automatic installation of the OpenShift DR Cluster operator and automatic transfer of S3 secrets on the managed clusters. For instructions, see Enabling automatic install of OpenShift DR cluster operator and Enabling automatic transfer of S3 secrets on managed clusters.

9. Create a sample application using RHACM console for testing failover and relocation testing. For instructions, see Creating sample application, application failover and relocating an application between managed clusters.

CHAPTER 2. REQUIREMENTS FOR ENABLING METRO-DR

Disaster Recovery features supported by Red Hat OpenShift Data Foundation require all of the following prerequisites in order to successfully implement a Disaster Recovery solution:

- Subscription requirements
  - A valid Red Hat OpenShift Data Foundation Advanced entitlement
  - A valid Red Hat Advanced Cluster Management for Kubernetes subscription

  To know how subscriptions for OpenShift Data Foundation work, see knowledgebase article on OpenShift Data Foundation subscriptions.

- You must have three OpenShift clusters that have network reachability between them:
  - **Hub cluster** where Advanced Cluster Management for Kubernetes (RHACM operator) and OpenShift DR Hub controllers are installed.
  - **Primary managed cluster** where OpenShift Data Foundation, OpenShift DR Cluster controller, and applications are installed.
  - **Secondary managed cluster** where OpenShift Data Foundation, OpenShift DR Cluster controller, and applications are installed.

- Ensure that RHACM operator and MultiClusterHub is installed on the Hub cluster. See RHACM installation guide for instructions.
  - Once deployment is completed, login to the RHACM console using your OpenShift credentials.
  - Find the Route that has been created for the Advanced Cluster Manager console:

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

    Example Output:

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

    After logging in using your OpenShift credentials, you should see your local cluster imported.

- Ensure that you either import or create the **Primary managed cluster** and the **Secondary managed cluster** using the RHACM console. Choose the appropriate options for your environment. After the managed clusters are successfully created or imported, you can see the list of clusters that were imported or created on the console.

CHAPTER 3. REQUIREMENTS FOR DEPLOYING RED HAT CEPH STORAGE STRETCH CLUSTER WITH ARBITER

Red Hat Ceph Storage is an open-source enterprise platform that provides unified software-defined storage on standard, economical servers and disks. With block, object, and file storage combined into one platform, Red Hat Ceph Storage efficiently and automatically manages all your data, so you can focus on the applications and workloads that use it.

This section provides a basic overview of the Red Hat Ceph Storage deployment. For more complex deployment, refer to the official documentation guide for RHCS 5.

NOTE

Only Flash media is supported since it runs with **min_size=1** when degraded. Use stretch mode only with all-flash OSDs. Using all-flash OSDs minimizes the time needed to recover once connectivity is restored, thus minimizing the potential for data loss.

IMPORTANT

Erasure coded pools cannot be used with stretch mode.

3.1. HARDWARE REQUIREMENTS

For information on minimum hardware requirements for deploying Red Hat Ceph Storage, see Minimum hardware recommendations for containerized Ceph.

Table 3.1. Physical server locations and Ceph component layout for Red Hat Ceph Storage cluster deployment:

Node name	Datacenter	Ceph components
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. SOFTWARE REQUIREMENTS

Use the latest software version of **Red Hat Ceph Storage 5**

For more information on the supported Operating System versions for Red Hat Ceph Storage, see knowledgebase article on Red Hat Ceph Storage: Supported configurations .

3.3. NETWORK CONFIGURATION REQUIREMENTS

The recommended Red Hat Ceph Storage configuration is as follows:

- You must have two separate networks, one public network and one private network.
- You must have three different datacenters that support VLANS and subnets for Cephs private and public network for all datacenters.

NOTE

You can use different subnets for each of the datacenters.

- The latencies between the two datacenters running the Red Hat Ceph Storage Object Storage Devices (OSDs) cannot exceed 10 ms RTT. For the **arbiter** datacenter, this was tested with values as high as 100 ms RTT to the other two OSD datacenters.

Here is an example of a basic network configuration that we have used in this guide:

- **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

For more information on the required network environment, see Ceph network configuration .

3.4. NODE PRE-DEPLOYMENT REQUIREMENTS

Before installing the Red Hat Ceph Storage cluster, perform the following steps to fulfill all the requirements needed.

1. Register all the nodes to the Red Hat Network or Red Hat Satellite and subscribe to a valid pool:

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

2. Enable access for all the nodes in the Ceph cluster for the following repositories:

   - **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. Update the operating system RPMs to the latest version and reboot if needed:

```
dnf update -y
reboot
```

4. Select a node from the cluster to be your bootstrap node. **ceph1** is our bootstrap node in this example going forward.
   Only on the bootstrap node **ceph1**, enable the **ansible-2.9-for-rhel-8-x86_64-rpms** and **rhceph-5-tools-for-rhel-8-x86_64-rpms** repositories:

   ```
   subscription-manager repos --enable="ansible-2.9-for-rhel-8-x86_64-rpms" --
   enable="rhceph-5-tools-for-rhel-8-x86_64-rpms"
   ```

5. Configure the **hostname** using the bare/short hostname in all the hosts.

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

6. Verify the hostname configuration for deploying Red Hat Ceph Storage with cephadm.

   ```
   $ hostname
   ```

   Example output:

   ```
   ceph1
   ```

7. Modify /etc/hosts file and add the fqdn entry to the 127.0.0.1 IP by setting the DOMAIN variable with our DNS domain name.

   ```
   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. Check the long hostname with the **fqdn** using the **hostname -f** option.

   ```
   $ hostname -f
   ```

   Example output:

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

   Note: To know more about why these changes are required, see Fully Qualified Domain Names vs Bare Host Names.

9. Run the following steps on bootstrap node. In our example, the bootstrap node is **ceph1**.

   a. Install the **cephadm-ansible** RPM package:

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

IMPORTANT

To run the ansible playbooks, you must have **ssh** passwordless access to all the nodes that are configured to the Red Hat Ceph Storage cluster. Ensure that the configured user (for example, **deployment-user**) has root privileges to invoke the **sudo** command without needing a password.

b. To use a custom key, configure the selected user (for example, **deployment-user**) ssh config file to specify the id/key that will be used for connecting to the nodes via ssh:

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

c. Build the ansible inventory

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

NOTE

Hosts configured as part of the [admin] group on the inventory file will be tagged as **_admin** by **cephadm**, so they receive the admin ceph keyring during the bootstrap process.

d. Verify that **ansible** can access all nodes using ping module before running the pre-flight playbook.

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

Example output:

```
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. Run the following ansible playbook.

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

The preflight playbook Ansible playbook configures the Red Hat Ceph Storage **dnf** repository and prepares the storage cluster for bootstrapping. It also installs podman, lvm2, chronyd, and cephadm. The default location for **cephadm-ansible** and **cephadm-preflight.yml** is **/usr/share/cephadm-ansible**.

3.5. CLUSTER BOOTSTRAPPING AND SERVICE DEPLOYMENT WITH CEPHADM

The cephadm utility installs and starts a single Ceph Monitor daemon and a Ceph Manager daemon for a new Red Hat Ceph Storage cluster on the local node where the cephadm bootstrap command is run.

> **NOTE**
>
> For additional information on the bootstrapping process, see Bootstrapping a new storage cluster.

Procedure

1. Create json file to authenticate against the container registry using a json file as follows:

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

2. Create a **cluster-spec.yaml** that adds the nodes to the RHCS cluster and also sets specific labels for where the services should run following table 3.1.

```
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. Retrieve the IP for the NIC with the RHCS public network configured from the bootstrap node. After substituting **10.0.40.0** with the subnet that you have defined in your ceph public network, execute the following command.

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

Example output:

```
10.0.40.78
```

4. Run the **Cephadm** bootstrap command as the **root** user on the node that will be the initial Monitor node in the cluster. The **IP_ADDRESS** option is the node's IP address that you are using to run the **cephadm bootstrap** command.

NOTE

If you have configured a different user instead of **root** for passwordless SSH access, then use the **--ssh-user=** flag with the **cepadm bootstrap** command.

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

IMPORTANT

If the local node uses fully-qualified domain names (FQDN), then add the **--allow-fqdn-hostname** option to **cephadm bootstrap** on the command line.

Once the bootstrap finishes, you will see the following output from the previous cephadm bootstrap command:

```
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. Verify the status of Red Hat Ceph Storage cluster deployment using the Ceph CLI client from ceph1:

```
$ ceph -s
```

Example output:

```
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
```

NOTE

It may take several minutes for all the services to start.

It is normal to get a global recovery event while you don’t have any osds configured.

You can use **ceph orch ps** and **ceph orch ls** to further check the status of the services.

6. Verify if all the nodes are part of the **cephadm** cluster.

```
$ ceph orch host ls
```

Example output:

```
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
```

> **NOTE**
>
> You can run Ceph commands directly from the host because **ceph1** was configured in the **cephadm-ansible** inventory as part of the [admin] group. The Ceph admin keys were copied to the host during the **cephadm bootstrap** process.

7. Check the current placement of the Ceph monitor services on the datacenters.

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

Example output:

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

8. Check the current placement of the Ceph manager services on the datacenters.

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

Example output:

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

9. Check the ceph osd crush map layout to ensure that each host has one OSD configured and its status is **UP**. Also, double-check that each node is under the right datacenter bucket as specified in table 3.1

```
$ ceph osd tree
```

Example output:

```
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. Create and enable a new RDB block pool.

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

> **NOTE**
>
> The number 32 at the end of the command is the number of PGs assigned to this pool. The number of PGs can vary depending on several factors like the number of OSDs in the cluster, expected % used of the pool, etc. You can use the following calculator to determine the number of PGs needed: Ceph Placement Groups (PGs) per Pool Calculator.

11. Verify that the RBD pool has been created.

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

Example output:

```
 3 rbdpool
```

12. Verify that MDS services are active and has located one service on each datacenter.

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

Example output:

```
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. Create the CephFS volume.

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

> **NOTE**
>
> The **ceph fs volume create** command also creates the needed data and meta CephFS pools. For more information, see Configuring and Mounting Ceph File Systems.

14. Check the **Ceph** status to verify how the MDS daemons have been deployed. Ensure that the state is active where **ceph6** is the primary MDS for this filesystem and **ceph3** is the secondary MDS.

```
$ ceph fs status
```

Example output:

```
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. Verify that RGW services are active.

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

Example output:

```
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
```

CHAPTER 4. CONFIGURING RED HAT CEPH STORAGE STRETCH CLUSTER

Once the Red Hat Ceph Storage cluster is fully deployed using **cephadm**, use the following procedure to configure the stretch cluster mode. The new stretch mode is designed to handle the 2-site case.

Procedure

1. Check the current election strategy being used by the monitors with the ceph mon dump command. By default in a ceph cluster, the connectivity is set to classic.

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

Example output:

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

2. Change the monitor election to connectivity.

```
ceph mon set election_strategy connectivity
```

3. Run the previous ceph mon dump command again to verify the election_strategy value.

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

Example output:

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

To know more about the different election strategies, see Configuring monitor election strategy.

4. Set the location for all our Ceph monitors:

```
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. Verify that each monitor has its appropriate location.

```
$ ceph mon dump
```

Example output:

```
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. Create a CRUSH rule that makes use of this OSD crush topology by installing the **ceph-base** RPM package in order to use the **crushtool** command:

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

   To know more about CRUSH ruleset, see Ceph CRUSH ruleset .

7. Get the compiled CRUSH map from the cluster:

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

8. Decompile the CRUSH map and convert it to a text file in order to be able to edit it:

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

9. Add the following rule to the CRUSH map by editing the text file **/etc/ceph/crushmap.txt** at the end of the file.

   ```
   $ 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
   ```

   NOTE

   The rule **id** has to be unique. In the example, we only have one more crush rule with id 0 hence we are using id 1. If your deployment has more rules created, then use the next free id.

The CRUSH rule declared contains the following information:

- **Rule name**:
  - Description: A unique whole name for identifying the rule.
  - Value: **stretch_rule**
- **id**:
  - Description: A unique whole number for identifying the rule.
  - Value: **1**
- **type**:
  - Description: Describes a rule for either a storage drive replicated or erasure-coded.
  - Value: **replicated**
- **min_size**:
  - Description: If a pool makes fewer replicas than this number, CRUSH will not select this rule.
  - Value: **1**
- **max_size**:
  - Description: If a pool makes more replicas than this number, CRUSH will not select this rule.
  - Value: **10**
- **step take DC1**
  - Description: Takes a bucket name (DC1), and begins iterating down the tree.
- **step chooseleaf firstn 2 type host**
  - Description: Selects the number of buckets of the given type, in this case is two different hosts located in DC1.
- **step emit**
  - Description: Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to pick from different trees in the same rule.
- **step take DC2**
  - Description: Takes a bucket name (DC2), and begins iterating down the tree.
- **step chooseleaf firstn 2 type host**
  - Description: Selects the number of buckets of the given type, in this case, is two different hosts located in DC2.
- **step emit**

- Description: Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to pick from different trees in the same rule.

10. Compile the new CRUSH map from the file **/etc/ceph/crushmap.txt** and convert it to a binary file called **/etc/ceph/crushmap2.bin**:

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

11. Inject the new crushmap we created back into the cluster:

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

Example output:

```
17
```

NOTE

The number 17 is a counter and it will increase (18,19, and so on) depending on the changes you make to the crush map.

12. Verify that the stretched rule created is now available for use.

```
ceph osd crush rule ls
```

Example output:

```
replicated_rule
stretch_rule
```

13. Enable the stretch cluster mode.

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

In this example, **ceph7** is the arbiter node, **stretch_rule** is the crush rule we created in the previous step and **datacenter** is the dividing bucket.

14. Verify all our pools are using the **stretch_rule** CRUSH rule we have created in our Ceph cluster:

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

Example output:

```
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
```

This indicates that a working Red Hat Ceph Storage stretched cluster with arbiter mode is now available.

CHAPTER 5. INSTALLING OPENSHIFT DATA FOUNDATION ON MANAGED CLUSTERS

In order to configure storage replication between the two OpenShift Container Platform clusters, OpenShift Data Foundation must be installed first on each managed cluster as follows:

1. Install the latest OpenShift Data Foundation on each of the managed clusters.

2. After installing the operator, create StorageSystem using the option **Connect with external storage platform**.
   For detailed instructions, refer to Deploying OpenShift Data foundation in external mode .

3. Validate the successful deployment of OpenShift Data foundation:

   a. on each managed cluster with the following command:

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

   b. For the Multicloud Gateway (MCG):

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

   If the status result is **Ready** for both queries on the **Primary managed cluster** and the **Secondary managed cluster**, then continue with the next step.

NOTE

The successful installation of OpenShift Data Foundation can also be validated in the OpenShift Container Platform Web Console by navigating to **Storage** and then **Data Foundation**.

CHAPTER 6. INSTALLING OPENSHIFT DR HUB OPERATOR ON HUB CLUSTER

Procedure

1. On the Hub cluster, navigate to OperatorHub and use the search filter for **OpenShift DR Hub Operator**.

2. Follow the screen instructions to Install the operator into the project **openshift-dr-system**.

3. Verify that the operator Pod is in **Running** state using the following command:

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

Example output:

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

CHAPTER 7. CONFIGURING MANAGED AND HUB CLUSTERS

7.1. CONFIGURING SSL ACCESS BETWEEN S3 ENDPOINTS

Configure network (SSL) access between the **s3 endpoints** so that metadata can be stored on the alternate cluster in a **MCG object bucket** using a secure transport protocol and in addition, the **Hub cluster** needs to verify access to the object buckets.

> **NOTE**
>
> If all of your OpenShift clusters are deployed using a signed and valid set of certificates for your environment then this section can be skipped.

Procedure

1. Extract the ingress certificate for the Primary managed cluster and save the output to **primary.crt**.

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

2. Extract the ingress certificate for the Secondary managed cluster and save the output to **secondary.crt**.

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

3. Create a new **ConfigMap** to hold the remote cluster's certificate bundle with filename **cm-clusters-crt.yaml** on the **Primary managed cluster**, **Secondary managed cluster**, and the **Hub cluster**.

> **NOTE**
>
> There could be more or less than three certificates for each cluster as shown in this example file. Also, ensure that the certificate contents are correctly indented after you copy and paste from the **primary.crt** and **secondary.crt** files that were created before.

```
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. Create the ConfigMap file on the **Primary managed cluster**, **Secondary managed cluster**, and the **Hub cluster**.

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

Example output:

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

IMPORTANT

For the Hub cluster to verify access to the object buckets using the **DRPolicy** resource, the same **ConfigMap** **cm-clusters-crt.yaml** must also be created on the Hub cluster.

5. Patch the default proxy resource on the **Primary managed cluster**, **Secondary managed cluster**, and the **Hub cluster**.

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

Example output:

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

7.2. CREATING OBJECT BUCKETS AND S3STOREPROFILES

OpenShift DR requires S3 stores to store relevant cluster data of a workload from the managed clusters and to orchestrate a recovery of the workload during failover or relocate actions. These instructions are applicable for creating the necessary object bucket(s) using Multicloud Object Gateway (MCG). MCG should already be installed as a result of installing OpenShift Data Foundation.

Procedure

1. Create MCG object bucket or OBC to be used for storing persistent volume metadata on both the Primary and Secondary managed clusters.

   a. Copy the following YAML file to filename **odrbucket.yaml**.

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

   b. Create a MCG bucket **odrbucket** on both the **Primary managed cluster** and the **Secondary managed cluster**.

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

   Example output:

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

2. Extract the **odrbucket OBC** access key for each managed cluster as their *base-64 encoded* values by using the following command.

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

   Example output:


   ```
   cFpIYTZWN1NhemJjbEUyWlpwN1E=
   ```


3. Extract the **odrbucket OBC** secret key for each managed cluster as their *base-64 encoded* values by using the following command.

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

   Example output:


   ```
   V1hUSnMzZUoxMHRRTXdGMU9jQXRmUlAyMmd5bGwwYjNvMHprZVhtNw==
   ```


IMPORTANT

The access key and secret key must be retrieved for the **odrbucket OBC** on both the **Primary managed cluster** and **Secondary managed cluster**.

7.3. CREATING S3 SECRETS FOR MULTICLOUD OBJECT GATEWAY OBJECT BUCKETS

Now that the necessary information has been extracted for the object buckets in the previous section, there must be new **Secrets** created on the **Hub cluster**. These new Secrets will store the MCG object bucket access key and secret key for both managed clusters on the Hub cluster.

Procedure

1. Copy the following S3 secret YAML format for the **Primary managed cluster** to filename **odr-s3secret-primary.yaml**.

```
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. Create this secret on the **Hub cluster**.

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

Example output:

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

3. Copy the following S3 secret YAML format for the **Secondary managed cluster** to filename **odr-s3secret-secondary.yaml**.

```
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. Create this secret on the **Hub cluster**.

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

Example output:

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

IMPORTANT

The values for the access key and secret key must be **base-64 encoded**. The encoded values for the keys were retrieved in the prior section.

7.4. CONFIGURE OPENSHIFT DR HUB OPERATOR S3STOREPROFILES

To find the s3CompatibleEndpoint or route for MCG, execute the following command on the Primary managed cluster and the Secondary managed cluster:

Procedure

a. Search for the external S3 endpoint **s3CompatibleEndpoint** or route for MCG on each managed cluster by using the following command.

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

Example output:

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

> **IMPORTANT**
>
> The unique **s3CompatibleEndpoint** route or **s3-openshift-storage.apps.<primary clusterID>.<baseDomain>** and **s3-openshift-storage.apps.<secondary clusterID>.<baseDomain>** must be retrieved for both the **Primary managed cluster** and **Secondary managed cluster** respectively.

b. Search for the **odrbucket OBC** exact bucket name.

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

Example output:

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

> **IMPORTANT**
>
> The unique **s3Bucket** name *odrbucket-<your value1>* and *odrbucket-<your value2>* must be retrieved on both the **Primary managed cluster** and **Secondary managed cluster** respectively.

c. Modify the ConfigMap **ramen-hub-operator-config** on the Hub cluster to add the new content.

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

d. Add the following new content starting at **s3StoreProfiles** to the ConfigMap on the **Hub cluster** only.

```
[...]
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
[...]
```

CHAPTER 8. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER

OpenShift DR uses Disaster Recovery Policy (DRPolicy) resources (cluster scoped) on the RHACM hub cluster to deploy, failover, and relocate workloads across managed clusters.

Prerequisites

- Ensure that there is a set of two clusters.
- Ensure that each cluster in the policy is assigned a S3 profile name, which is configured using the ConfigMap of the OpenShift DR cluster and hub operators.

Procedure

1. On the **Hub cluster**, navigate to Installed Operators in the **openshift-dr-system** project and click on **OpenShift DR Hub Operator**. You should see two available APIs, **DRPolicy** and **DRPlacementControl**.
2. Click **Create instance** for DRPolicy and click **YAML view**.
3. Save the following YAML to filename **drpolicy.yaml** after replacing <cluster1> and <cluster2> with the correct names of your managed clusters in **RHACM**. Replace <string_value> with any value (i.e. 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
```

NOTE

There is no need to specify a namespace to create this resource because DRPolicy is a cluster-scoped resource.

4. Copy the contents of your unique **drpolicy.yaml** file into the YAML view. You must completely replace the original content.
5. Click **Create** on the YAML view screen.
6. To validate that the **DRPolicy** is created successfully and that the MCG object buckets can be accessed using the Secrets created earlier, run this command on the **Hub cluster**:

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

Example output:

```
Succeeded
```

CHAPTER 9. ENABLING AUTOMATIC INSTALL OF OPENSHIFT DR CLUSTER OPERATOR

Once the DRPolicy is created successfully, the **OpenShift DR Cluster operator** can be installed on the Primary managed cluster and Secondary managed cluster in the **openshift-dr-system** namespace.

Procedure

1. Edit the ConfigMag **ramen-hub-operator-config** on the Hub cluster and modify the value of **deploymentAutomationEnabled=false** to **true** as follows:

```
$ 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. Verify that the installation was successful in the **Primary managed cluster** and the **Secondary managed cluster** do the following command:

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

Example output:

```
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
```

You can also go to OperatorHub on each of the managed clusters and verify if the **OpenShift DR Cluster Operator** is installed.

CHAPTER 10. ENABLING AUTOMATIC TRANSFER OF S3SECRETS TO MANAGED CLUSTERS

Follow this procedure to enable auto transfer of s3Secrets to the required OpenShift DR cluster components. It updates the OpenShift DR cluster namespace with the s3Secrets that are required to access the s3Profiles in the OpenShift DR config map.

Procedure

1. Edit the ConfigMag **ramen-hub-operator-config** on the Hub cluster to add **s3SecretDistributionEnabled=true** as follows:

```
$ 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. Verify that transfer of secrets was successful by running this command in both managed clusters.

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

Example output:

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

CHAPTER 11. CREATING A SAMPLE APPLICATION

In order to test **failover** from the Primary managed cluster to the Secondary managed cluster and back again we need a simple application. Use the sample application called **busybox** as an example.

Procedure

1. Create a **namespace** or **project** on the **Hub cluster** for a **busybox** sample application.

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

> **NOTE**
>
> A different project name other than **busybox-sample** can be used if desired. Make sure when deploying the sample application via the Advanced Cluster Manager console to use the same project name as what is created in this step.

2. Create **DRPlacementControl** resource
   DRPlacementControl is an API available after the OpenShift DR Hub Operator is installed on the Hub cluster. It is broadly an Advanced Cluster Manager PlacementRule reconciler that orchestrates placement decisions based on data availability across clusters that are part of a DRPolicy.

   a. On the Hub cluster, navigate to Installed Operators in the **busybox-sample** project and click on **OpenShift DR Hub Operator**. You should see two available APIs, DRPolicy and DRPlacementControl.

   b. Create an instance for **DRPlacementControl** and then go to the YAML view. Make sure the **busybox-sample** project is selected.

   c. Copy and save the following YAML to filename **busybox-drpc.yaml** after replacing *<cluster1>* with the correct name of your managed cluster in Advanced Cluster Manager.

```
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. Copy the contents of your unique **busybox-drpc.yaml** file into the YAML view (completely replacing original content).

   e. Click **Create** on the YAML view screen.

You can also create this resource using the following CLI command:

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

Example output:

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

> **IMPORTANT**
>
> This resource must be created in the **busybox-sample** namespace (or whatever namespace you created earlier).

3. Create **Placement Rule** resource that defines the target clusters where resource templates can be deployed. Use placement rules to facilitate the multicluster deployment of your applications.

   a. Copy and save the following YAML to filename **busybox-placementrule.yaml**.

```
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. Create the Placement Rule resource for the **busybox-sample** application.

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

Example output:

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

> **IMPORTANT**
>
> This resource must be created in the **busybox-sample** namespace (or whatever namespace you created earlier).

4. Create **sample application** using RHACM console

   a. Log in to the RHACM console using your OpenShift credentials if not already logged in.

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

Example Output:

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

b. Navigate to **Applications** and click **Create application**.

c. Select type as **Subscription**.

d. Enter your application **Name** (for example, **busybox**) and **Namespace** (for example, **busybox-sample**).

e. In Repository location for resources section, select **Repository type Git**.

f. Enter the Git repository URL for the sample application, the github **Branch** and **Path** where the resources **busybox** Pod and PVC will be created.
Use the sample application repository as **https://github.com/RamenDR/ocm-ramen-samples** where the **Branch** is **main** and **Path** is **busybox-odr-metro**.

g. Scroll down the form to the section **Select clusters to deploy to** and click **Select an existing placement configuration**.

h. Select an **Existing Placement Rule** (for example, **busybox-placement**) from the drop-down list.

i. Click **Save**.
On the follow-on screen scroll to the bottom. You should see that there are all Green checkmarks on the application topology.

> **NOTE**
>
> To get more information, click on any of the topology elements and a window will appear on the right of the topology view.

5. Validating the sample application deployment and replication.
Now that the **busybox** application has been deployed to your preferred Cluster (specified in the DRPlacementControl) the deployment can be validated.

a. Login to your managed cluster where **busybox** was deployed by RHACM.

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

Example output:

```
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. Verify that the replication resources are also created for the **busybox** PVC.

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

Example output:

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

11.1. DELETING SAMPLE APPLICATION

You can delete the sample application **busybox** using the RHACM console.

> **NOTE**
>
> The instructions to delete the sample application should not be executed until the failover and failback (relocate) testing is completed and the application is ready to be removed from RHACM and the managed clusters.

Procedure

1. On the RHACM console, navigate to **Applications**.
2. Search for the sample application to be deleted (for example, **busybox**).
3. Click the Action Menu **(⋮)** next to the application you want to delete.
4. Click **Delete application**.
   When Delete application is selected a new screen will appear asking if the application related resources should also be deleted.
5. Select **Remove application related resources** checkbox to delete the Subscription and PlacementRule.
6. Click **Delete**. This will delete the busybox application on the Primary managed cluster (or whatever cluster the application was running on).
7. In addition to the resources deleted using the RHACM console, the **DRPlacementControl** must also be deleted immediately after deleting the **busybox** application.
   a. Login to the OpenShift Web console for the Hub cluster and navigate to Installed Operators for the project **busybox-sample**.
   b. Click **OpenShift DR Hub Operator** and then click **DRPlacementControl** tab.
   c. Click the Action Menu **(⋮)** next to the **busybox** application DRPlacementControl that you want to delete.
   d. Click **Delete DRPlacementControl**.
   e. Click **Delete**.

> **NOTE**
>
> This process can be used to delete any application with a **DRPlacementControl** resource. The **DRPlacementControl** resource can also be deleted in the application namespace using CLI.

CHAPTER 12. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

This section provides instructions on how to failover the busybox sample application. The failover method for Metro-DR is application based. Each application that is to be protected in this manner must have a corresponding **DRPlacementControl** resource and a **PlacementRule** resource created in the application **namespace** as shown in the Create Sample Application for DR testing section.

Procedure

1. Create **NetworkFence** resource and enable **Fencing**.
   Specify the list of CIDR blocks or IP addresses on which network fencing operation will be performed. In our case, this will be the EXTERNAL-IP of every OpenShift node in the cluster that needs to be fenced from using the external RHCS cluster.

   a. Execute this command to get the IP addresses for the **Primary managed cluster**.

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

      Example output:

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

      NOTE

      Collect the current IP addresses of all OpenShift nodes before there is a site outage. Best practice would be to create the **NetworkFence** YAML file and have it available and up-to-date for a disaster recovery event.

      The IP addresses for all nodes will be added to the **NetworkFence** example resource as shown below. This example is for six nodes but there could be more nodes in your cluster.

      ```
      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. For the YAML file example above, modify the IP addresses and provide the correct *<cluster1>* to be the cluster name found in **RHACM** for the Primary managed cluster. Save this to filename **network-fence-<cluster1>.yaml**.

> **IMPORTANT**
>
> The **NetworkFence** must be created from the opposite managed cluster where the application is currently running prior to failover. In this case, that is the **Secondary managed cluster**.

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

Example output:

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

> **IMPORTANT**
>
> After the **NetworkFence** is created, all communication from applications to the OpenShift Data Foundation storage will fail and some Pods will be in an unhealthy state (For example: CreateContainerError, CrashLoopBackOff) on the cluster that is now fenced.

c. In the same cluster as where the **NetworkFence** was created, verify that the status is 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"}'
```

Example output:

```
Succeeded
```

2. Modify **DRPolicy** for the **fenced** cluster.

a. Edit the **DRPolicy** on the Hub cluster and change *<cluster1>* (for example: ocp4perf1) from **Unfenced** to **ManuallyFenced**.

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

Example output:

```
[...]
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
[...]
```

Example output:

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

b. Validate the **DRPolicy** status in the Hub cluster has changed to **Fenced** for the **Primary managed cluster**.

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

Example output:

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

3. Modify **DRPlacementControl** to **failover**

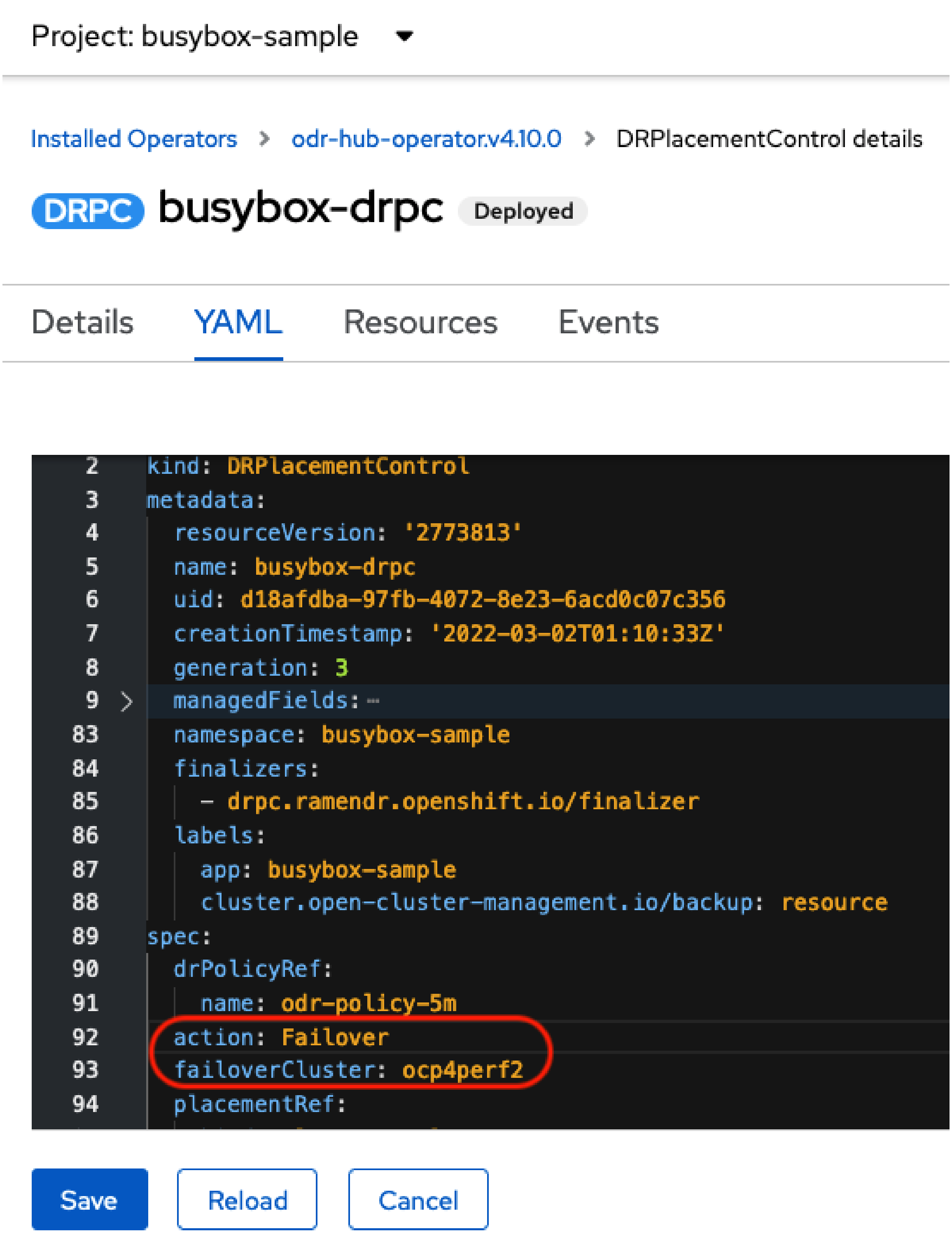
   a. On the Hub cluster navigate to Installed Operators and then click **Openshift DR Hub Operator**.

   b. Click **DRPlacementControl** tab.

   c. Click DRPC **busybox-drpc** and then the YAML view.

   d. Add the **action** and **failoverCluster** details as shown in below screenshot. The **failoverCluster** should be the ACM cluster name for the Secondary managed cluster.

DRPlacementControl add action Failover



e. Click **Save**.

4. Verify that the application **busybox** is now running in the Secondary managed cluster, the failover cluster **ocp4perf2** specified in the YAML file.

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

Example output:

```
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. Verify that **busybox** is no longer running on the Primary managed cluster.

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

Example output:

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

IMPORTANT

Be aware of known Metro-DR issues as documented in Known Issues section of Release Notes.

CHAPTER 13. RELOCATING AN APPLICATION BETWEEN MANAGED CLUSTERS

A relocation operation is very similar to failover. Relocate is application based and uses the **DRPlacementControl** to trigger the relocation. The main difference for failback is that the application is scaled down on the failoverCluster and therefore creating a **NetworkFence** is not required.

Procedure

1. Remove **NetworkFence** resource and disable **Fencing**.
   Before a failback or relocate action can be successful the **NetworkFence** for the Primary managed cluster must be deleted.

   a. Execute this command in the **Secondary managed cluster** and modify *<cluster1>* to be correct for the **NetworkFence** YAML filename created in the prior section.

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

   Example output:

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

   b. Reboot OpenShift Container Platform nodes that were **Fenced**.
   This step is required because some application Pods on the prior fenced cluster, in this case the Primary managed cluster, are in an unhealthy state (For example: CreateContainerError, CrashLoopBackOff). This can be most easily fixed by rebooting all worker OpenShift nodes one at a time.

   > **NOTE**
   >
   > The OpenShift Web Console dashboards and **Overview** page can also be used to assess the health of applications and the external storage. The detailed OpenShift Data Foundation dashboard is found by navigating to **Storage → Data Foundation**

   c. Verify all Pods are in a healthy state by running this command on the **Primary managed cluster** after all OpenShift nodes have rebooted and are in a **Ready** status. The output for this query should be zero Pods.

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

   Example output:

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

   > **IMPORTANT**
   >
   > If there are Pods still in an unhealthy status because of severed storage communication, troubleshoot and resolve before continuing. Because the storage cluster is external to OpenShift, it also has to be properly recovered after a site outage for OpenShift applications to be healthy.

2. Modify **DRPolicy** to **Unfenced** status.
   In order for the ODR HUB operator to know the **NetworkFence** has been removed for the Primary managed cluster the **DRPolicy** must be modified for the newly **Unfenced** cluster.

   a. Edit the **DRPolicy** on the Hub cluster and change *<cluster1>* (example **ocp4perf1**) from **ManuallyFenced** to **Unfenced**.

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

   Example output:

   ```
   [...]
   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
   [...]
   ```

   Example output:

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

   b. Verify that the status of **DRPolicy** in the **Hub cluster** has changed to **Unfenced** for the **Primary managed cluster**.

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

   Example output:

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

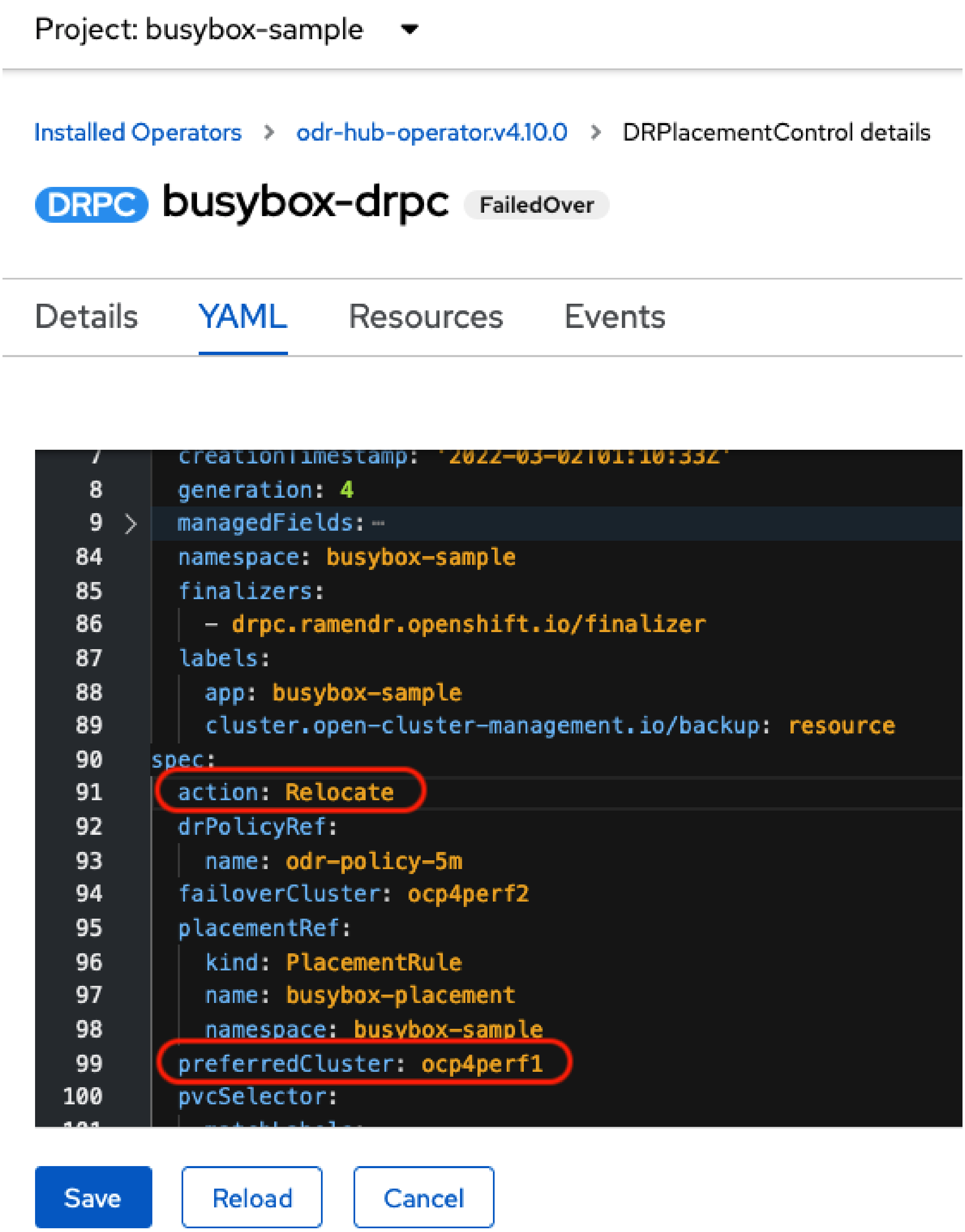
3. Modify **DRPlacementControl** to **failback**

   a. On the Hub cluster navigate to Installed Operators and then click **Openshift DR Hub Operator**.

   b. Click **DRPlacementControl** tab.

   c. Click DRPC **busybox-drpc** and then the YAML view.

   d. Modify **action** to **Relocate**.



DRPlacementControl modify action to Relocate

e. Click **Save**.

f. Verify if the application **busybox** is now running in the Primary managed cluster.The failback is to the **preferredCluster ocp4perf1** as specified in the YAML file, which is where the application was running before the failover operation.

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

Example output:

```
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. Verify if **busybox** is running in the Secondary managed cluster. The busybox application should no longer be running on this managed cluster.

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

Example output:

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

IMPORTANT

Be aware of known Metro-DR issues as documented in Known Issues section of Release Notes.