Red Hat OpenShift Data Foundation 4.9

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

Instructions about setting up OpenShift Data Foundation between two different geographical locations for providing storage infrastructure with disaster recovery capabilities.

Last Updated: 2022-05-16

Red Hat OpenShift Data Foundation 4.9 Configuring OpenShift Data Foundation for Regional-DR with Advanced Cluster Management

Instructions about setting up OpenShift Data Foundation between two different geographical locations for providing storage infrastructure with disaster recovery capabilities.

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 Regional-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 REGIONAL-DR

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

Regional-DR capability provides volume persistent data and metadata replication across sites that are geographically dispersed. In the public cloud these would be akin to protecting from a region failure. Regional-DR ensures business continuity during the unavailability of a geographical region, accepting some loss of data in a predictable amount. 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.
- 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 steps and commands necessary for configuring your infrastructure for enabling disaster recovery.

1.1. COMPONENTS OF REGIONAL-DR SOLUTION

Regional-DR is composed of Red Hat Advanced Cluster Management for Kubernetes (RHACM) and OpenShift Data Foundation components to provide application and data mobility across OpenShift Container Platform clusters.

Red Hat Advanced Cluster Management for Kubernetes (RHACM)

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.

OpenShift Data Foundation

OpenShift Data Foundation provides the ability to provision and manage storage for stateful applications in an OpenShift Container Platform cluster.

OpenShift Data Foundation is backed by Ceph as the storage provider, whose lifecycle is managed by Rook in the OpenShift Data Foundation component stack. Ceph-CSI provides the provisioning and management of Persistent Volumes for stateful applications.

OpenShift Data Foundation stack is enhanced with the ability to:

- Enable pools for mirroring
- Automatically mirror images across RBD pools

- Provides 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 on unavailability of the currently deployed cluster
- Relocate an application's state to the previously deployed cluster

OpenShift DR is split into three components:

- **ODF Multicluster Orchestrator**: Installed on the multi-cluster control plane (RHACM Hub), creates a bootstrap token and exchanges this token between the managed clusters.
- **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. REGIONAL-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Regional-DR capabilities using OpenShift Data Foundation version 4.9 and RHACM version 2.4 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 hub solution.

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

1. Ensure you meet each of the Regional-DR requirements. See Requirements for enabling Regional-DR.
2. Configure multisite storage replication by creating the mirroring relationship between two OpenShift Data Foundation managed clusters. See Configuring multisite storage replication .
3. Create a VolumeReplicationClass resource on each managed cluster to configure the replication schedule (for example: replicate between peers every 5 minutes). See Creating VolumeReplicationClass resource.
4. Create a mirroring StorageClass resource on each managed cluster that supports new **imageFeatures** for block volumes that have mirroring enabled. See Creating mirroring StorageClass resource.
5. Install the OpenShift DR Cluster Operator on the managed clusters and create the required object buckets, secrets and configmaps. See Installing OpenShift DR Cluster Operator on Managed clusters.
6. Install the OpenShift DR Hub Operator on the Hub cluster and create the required object buckets, secrets and configmap. See Installing OpenShift DR Hub Operator on Hub cluster .

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 .

CHAPTER 2. REQUIREMENTS FOR ENABLING REGIONAL-DR

- You must have three OpenShift clusters that have network reachability between them:
  - **Hub cluster** where Advanced Cluster Management for Kubernetes (RHACM operator), ODF Multicluster Orchestrator 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 you have installed RHACM operator and MultiClusterHub on the Hub cluster and logged on to the RHACM console using your OpenShift credentials. See RHACM installation guide for instructions.
  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 have either imported or created the **Primary managed cluster** and the **Secondary managed clusters** using the RHACM console.
  To connect the managed OpenShift cluster and service networks using the Submariner add-ons, you need to validate that the two clusters have non-overlapping networks by running the following commands for each of the managed clusters.

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

  Example output for **cluster1** (for example, **ocp4perf1**):

  ```
  {
    "clusterNetwork": [
      {
        "cidr": "10.5.0.0/16",
        "hostPrefix": 23
      }
    ],
    "externalIP": {
      "policy": {}
    },
    "networkType": "OpenShiftSDN",
    "serviceNetwork": [
      "10.15.0.0/16"
    ]
  }
  ```

  Example output for **cluster2** (for example, **ocp4perf2**):

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

For more information, see Submariner add-ons documentation.

- Ensure that the Managed clusters can connect using **Submariner add-ons**. After identifying and ensuring that the cluster and service networks have non-overlapping ranges, install the **Submariner add-ons** for each managed cluster using the RHACM console and **Cluster sets**. For instructions, see Submariner documentation.

- Ensure OpenShift Data Foundation 4.9 or greater is installed on each of the managed clusters.

  - For information about the OpenShift Data Foundation deployment, refer to your infrastructure specific deployment guides (for example, AWS, VMware, Bare metal, Azure).

  - Validate the successful deployment on each managed cluster with the following command:

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

  If the status result is **Ready** on the **Primary managed cluster** and the **Secondary managed cluster**, then continue with enabling mirroring on the managed clusters.

CHAPTER 3. CONFIGURING MULTISITE STORAGE REPLICATION

Mirroring or replication is enabled on a per **CephBlockPool** basis within peer managed clusters and can then be configured on a specific subset of images within the pool. The **rbd-mirror** daemon is responsible for replicating image updates from the local peer cluster to the same image in the remote cluster.

These instructions detail how to create the mirroring relationship between two OpenShift Data Foundation managed clusters.

3.1. ENABLING OMAP GENERATOR AND VOLUME REPLICATION ON MANAGED CLUSTERS

Execute the following steps on the **Primary managed cluster** and the **Secondary managed cluster** to enable the OMAP and Volume-Replication CSI sidecar containers in the **csi-rbdplugin-provisioner** Pods.

Procedure

1. Run the following **patch** command to set the value to **true** for **CSI_ENABLE_OMAP_GENERATOR** in the **rook-ceph-operator-config** ConfigMap.

   ```
   $ oc patch cm rook-ceph-operator-config -n openshift-storage --type json --patch  '[{ "op": "add", "path": "/data/CSI_ENABLE_OMAP_GENERATOR", "value": "true" }]'
   ```

   Example output:

   ```
   configmap/rook-ceph-operator-config patched
   ```

2. Run the following **patch** command to set the value to **true** for **CSI_ENABLE_VOLUME_REPLICATION** in the **rook-ceph-operator-config** ConfigMap.

   ```
   $ oc patch cm rook-ceph-operator-config -n openshift-storage --type json --patch  '[{ "op": "add", "path": "/data/CSI_ENABLE_VOLUME_REPLICATION", "value": "true" }]'
   ```

   Example output:

   ```
   configmap/rook-ceph-operator-config patched
   ```

3. Validate that the following two new CSI sidecar containers per **csi-rbdplugin-provisioner** pod are added.

   ```
   $ for l in $(oc get pods -n openshift-storage -l app=csi-rbdplugin-provisioner -o jsonpath={.items[*].spec.containers[*].name}) ; do echo $l ; done | egrep "csi-omap-generator|volume-replication"
   ```

   Example output:

   ```
   csi-omap-generator
   volume-replication
   csi-omap-generator
   volume-replication
   ```

NOTE

The new containers are repeated because there are two **csi-rbdplugin-provisioner** pods for redundancy.

3.2. INSTALLING OPENSHIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR

OpenShift Data Foundation Multicluster Orchestrator is a controller that is installed from OpenShift Container Platform's OperatorHub on the Hub cluster. This Multicluster Orchestrator controller, along with the MirrorPeer custom resource, creates a bootstrap token and exchanges this token between the managed clusters.

Procedure

1. Navigate to **OperatorHub** on the **Hub cluster** and use the keyword filter to search for **ODF Multicluster Orchestrator**.
2. Click **ODF Multicluster Orchestrator** tile.
3. Keep all default settings and click Install.
   The operator resources are installed in **openshift-operators** and available to all namespaces.
4. Verify that the **ODF Multicluster Orchestrator** shows a green tick indicating successful installation.

3.3. CREATING MIRROR PEER ON HUB CLUSTER

Mirror Peer is a cluster-scoped resource to hold information about the managed clusters that will have a peer-to-peer relationship.

Prerequisites

- Ensure that **ODF Multicluster Orchestrator** is installed on the **Hub cluster**.
- You must have only two clusters per Mirror Peer.
- Ensure that each cluster has uniquely identifiable cluster names such as **ocp4perf1** and **ocp4perf2**.

Procedure

1. Click **ODF Multicluster Orchestrator** to view the operator details.
   You can also click **View Operator** after the Multicluster Orchestrator is installed successfully.
2. Click on Mirror Peer API **Create instance** and then select **YAML** view.
3. Create Mirror Peer in YAML view.
   a. Copy the following YAML to filename **mirror-peer.yaml** after replacing *<cluster1>* and *<cluster2>* with the correct names of your managed clusters in the RHACM console.

```
apiVersion: multicluster.odf.openshift.io/v1alpha1
kind: MirrorPeer
```

```
metadata:
  name: mirrorpeer-<cluster1>-<cluster2>
spec:
  items:
  - clusterName: <cluster1>
    storageClusterRef:
      name: ocs-storagecluster
      namespace: openshift-storage
  - clusterName: <cluster2>
    storageClusterRef:
      name: ocs-storagecluster
      namespace: openshift-storage
```

NOTE

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

b. Copy the contents of your unique **mirror-peer.yaml** file into the YAML view. You must completely replace the original content.

c. Click **Create** at the bottom of the YAML view screen.

4. Verify that you can view **Phase** status as **ExchangedSecret**.

NOTE

In some deployments, the output for the validation can also be **ExchangingSecret** which is also an acceptable result.

3.4. ENABLING MIRRORING ON MANAGED CLUSTERS

To enable mirroring, you must enable the mirroring setting of the storage cluster for each managed cluster. This is a manual step using CLI and the **oc patch** command.

IMPORTANT

You must run the **oc patch storagecluster** command on the **Primary managed cluster** and the **Secondary managed cluster** as well as the follow-on validation commands after the StorageCluster has mirroring enabled.

Procedure

1. Enable cluster level mirroring flag using storage cluster name.

```
$ oc patch storagecluster $(oc get storagecluster -n openshift-storage -
o=jsonpath='{.items[0].metadata.name}')  -n openshift-storage --type json --patch  '[{ "op":
"replace", "path": "/spec/mirroring", "value": {"enabled": true} }]'
```

Example output:

```
storagecluster.ocs.openshift.io/ocs-storagecluster patched
```

2. Validate that mirroring is enabled on the default Ceph block pool.

```
$ oc get cephblockpool -n openshift-storage -o=jsonpath='{.items[?
(@.metadata.ownerReferences[*].kind=="StorageCluster")].spec.mirroring.enabled}{"\n"}'
```

Example output:

```
true
```

3. Validate that the **rbd-mirror** pod is up and running.

```
$ oc get pods -o name -l app=rook-ceph-rbd-mirror -n openshift-storage
```

Example output:

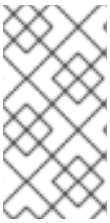
```
pod/rook-ceph-rbd-mirror-a-6486c7d875-56v2v
```

4. Validate the status of the **daemon** health.

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

Example output:

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



NOTE

It could take up to 10 minutes for the **daemon** health and health fields to change from **Warning** to **OK**. If the status does not change to **OK** in approximately 10 minutes then use the RHACM console to verify that the Submariner add-on connection is still in a **Healthy** state.

CHAPTER 4. CREATING VOLUMEREPLICATIONCLASS RESOURCE

The **VolumeReplicationClass** is used to specify the **mirroringMode** for each volume to be replicated as well as how often a volume or image is replicated (for example, every 5 minutes) from the local cluster to the remote cluster.

NOTE

This resource must be created on the **Primary managed cluster** and the **Secondary managed cluster**.

Procedure

1. Save the following YAML to filename **rbd-volumereplicationclass.yaml**.

```
apiVersion: replication.storage.openshift.io/v1alpha1
kind: VolumeReplicationClass
metadata:
  name: odf-rbd-volumereplicationclass
spec:
  provisioner: openshift-storage.rbd.csi.ceph.com
  parameters:
    mirroringMode: snapshot
    schedulingInterval: "5m"  # <-- Must be the same as scheduling interval in the DRPolicy
    replication.storage.openshift.io/replication-secret-name: rook-csi-rbd-provisioner
    replication.storage.openshift.io/replication-secret-namespace: openshift-storage
```

2. Create the file on both the managed clusters.

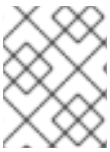
```
$ oc create -f rbd-volumereplicationclass.yaml
```

Example output:

```
volumereplicationclass.replication.storage.openshift.io/odf-rbd-volumereplicationclass
created
```

CHAPTER 5. CREATING MIRRORING STORAGECLASS RESOURCE

The block volumes with **mirroring enabled** must be created using a new **StorageClass** that has additional **imageFeatures** required to enable faster image replication between managed clusters. The new features are *exclusive-lock*, *object-map*, and *fast-diff*. The default OpenShift Data Foundation **StorageClass ocs-storagecluster-ceph-rbd** does not include these features.



NOTE

This resource must be created on the **Primary managed cluster** and the **Secondary managed cluster**.

Procedure

1. Save the following YAML to filename **ocs-storagecluster-ceph-rbdmirror.yaml**.

```
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocs-storagecluster-ceph-rbdmirror
parameters:
  clusterID: openshift-storage
  csi.storage.k8s.io/controller-expand-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/controller-expand-secret-namespace: openshift-storage
  csi.storage.k8s.io/fstype: ext4
  csi.storage.k8s.io/node-stage-secret-name: rook-csi-rbd-node
  csi.storage.k8s.io/node-stage-secret-namespace: openshift-storage
  csi.storage.k8s.io/provisioner-secret-name: rook-csi-rbd-provisioner
  csi.storage.k8s.io/provisioner-secret-namespace: openshift-storage
  imageFeatures: layering,exclusive-lock,object-map,fast-diff
  imageFormat: "2"
  pool: ocs-storagecluster-cephblockpool
provisioner: openshift-storage.rbd.csi.ceph.com
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

2. Create the file on both the managed clusters.

```
$ oc create -f ocs-storagecluster-ceph-rbdmirror.yaml
```

Example output:

```
storageclass.storage.k8s.io/ocs-storagecluster-ceph-rbdmirror created
```

CHAPTER 6. INSTALLING OPENSHIFT DR CLUSTER OPERATOR ON MANAGED CLUSTERS

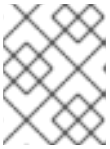
Procedure

1. On each managed cluster, navigate to OperatorHub and filter for **OpenShift DR Cluster Operator**.
2. Follow the screen instructions to install the operator into the project **openshift-dr-system**.

NOTE

The **OpenShift DR Cluster Operator** must be installed on both the **Primary managed cluster** and **Secondary managed cluster**.

3. **Configure 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 the **Hub cluster** for verifying access to the object buckets.



NOTE

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

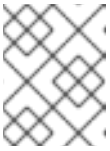
a. Extract the ingress certificate for the **Primary managed cluster** and save the output to **primary.crt**.

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

b. 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
```

c. Create a new **ConfigMap** to hold certificate bundle of the remote cluster with file name **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.

```
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
```

d. Run the following command on the **Primary managed cluster**, **Secondary managed cluster**, and the **Hub cluster** to create the file.

```
$ 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 be created on the **Hub cluster**.

e. Modify default **Proxy** cluster resource.

i. Copy and save the following content into the new YAML file **proxy-ca.yaml**.

```
apiVersion: config.openshift.io/v1
kind: Proxy
metadata:
  name: cluster
spec:
  trustedCA:
    name: user-ca-bundle
```

ii. Apply this new file to the default proxy resource on the **Primary managed cluster**, **Secondary managed cluster**, and the **Hub cluster**.

```
$ oc apply -f proxy-ca.yaml
```

Example output:

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

4. Retrieve Multicloud Object Gateway (MCG) keys and external S3 endpoint.

   a. Check if MCG is installed on the **Primary managed cluster** and the **Secondary managed cluster**, and if **Phase** is **Ready**.

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

   Example output:

   ```
   NAME     MGMT-ENDPOINTS                S3-ENDPOINTS                  IMAGE
   PHASE   AGE
   noobaa   ["https://10.70.56.161:30145"]   ["https://10.70.56.84:31721"]   quay.io/rhceph-
   dev/mcg-
   core@sha256:c4b8857ee9832e6efc5a8597a08b81730b774b2c12a31a436e0c3fadff48e73
   d   Ready   27h
   ```

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

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

   c. 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
   ```

   d. 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-dr-system -o
   jsonpath='{.data.AWS_ACCESS_KEY_ID}{"\n"}'
   ```

   Example output:

   ```
   cFpIYTZWN1NhemJjbEUyWlpwN1E=
   ```

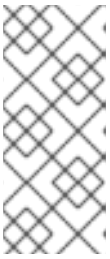
e. Extract the **odrbucket OBC** secret key for each managed cluster as their ***base-64 encoded*** values by using the following command.

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

Example output:

```
V1hUSnMzZUoxMHRRTXdGMU9jQXRmUlAyMmd5bGwwYjNvMHprZVhtNw==
```

5. Create S3 Secrets for Managed clusters.
Now that the necessary MCG information has been extracted there must be new Secrets created on the **Primary managed cluster** and the **Secondary managed cluster**. These new Secrets stores the MCG access and secret keys for both managed clusters.



NOTE

OpenShift DR requires one or more 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 Gateway (MCG). MCG should already be installed as a result of installing OpenShift Data Foundation.

a. 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
```

Replace *<primary cluster base-64 encoded access key>* and *<primary cluster base-64 encoded secret access key>* with actual values retrieved in an earlier step.

b. Create this secret on the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

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

c. 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
```

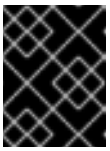
Replace *<secondary cluster base-64 encoded access key>* and *<secondary cluster base-64 encoded secret access key>* with actual values retrieved in step 4.

d. Create this secret on the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

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



IMPORTANT

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

6. Configure OpenShift DR Cluster Operator ConfigMaps on each of the managed clusters.

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

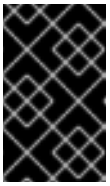
The unique **s3CompatibleEndpoint** route or **s3-openshift-storage.apps.<primary clusterID>.<baseDomain>** and **s3-openshift-storage.apps.<secondary clusterID>.<baseDomain>** must be retrieved for both the **Primary managed cluster** and **Secondary managed cluster** respectively.

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

```
$ oc get configmap odrbucket -n openshift-dr-system -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-dr-cluster-operator-config** to add the new content.

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

d. Add the following new content starting at **s3StoreProfiles** to the ConfigMap on the **Primary managed cluster** and the **Secondary managed cluster**.

```
[...]
data:
  ramen_manager_config.yaml: |
    apiVersion: ramendr.openshift.io/v1alpha1
    kind: RamenConfig
[...]
    ramenControllerType: "dr-cluster"
    ### 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 7. INSTALLING OPENSHIFT DR HUB OPERATOR ON HUB CLUSTER

Prerequisites

- Ensure that the values for the access and secret key are **base-64 encoded**. The encoded values for the keys were retrieved in the prior section and the resulting **Secrets** are *exactly* the same as those created already on the managed clusters.

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. Create S3 secrets for the Hub cluster using the following S3 secret YAML format for the **Primary managed cluster**.

```
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
```

   Run the following command to create this secret on the Hub cluster.

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

   Example output:

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

4. Create S3 secrets using the following S3 secret YAML format for the **Secondary managed cluster**.

```
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
```

   Run the following command to create this secret on the Hub cluster.

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

   Example output:

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

5. Configure ConfigMap for the OpenShift DR Hub Operator.
   After the operator is successfully created, a new ConfigMap called **ramen-hub-operator-config** is created.

   a. Run the following command to edit the file.

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

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

   ```
   [...]
   apiVersion: v1
   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
   [...]
   ```

NOTE

Make sure to replace *<primary clusterID>*, *<secondary clusterID>*, *baseDomain*, *odrbucket-<your value1>*, and *odrbucket-<your value2>* variables with exact same values as used for the **ramen-cluster-operator-config ConfigMap** on the managed clusters.

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.

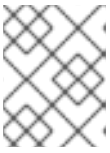
Prerequisites

- Ensure that there is a set of two clusters, which are peered for storage level replication and that CSI Volume Replication is enabled.
- Ensure that there is a scheduling interval that determines at what frequency data replication is performed which also serves as a coarse grained Recovery Point Objective (RPO) for the workload using the DRPolicy.
- 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. Copy and save the following YAML to filename **drpolicy.yaml** after replacing *<cluster1>* and *<cluster2>* with the correct names of your managed clusters in RHACM.

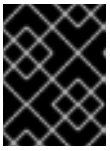
```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: odr-policy-5m
spec:
  drClusterSet:
  - name: <cluster1>
    s3ProfileName: s3-primary
  - name: <cluster2>
    s3ProfileName: s3-secondary
  schedulingInterval: 5m
```



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.



IMPORTANT

The DRPolicy scheduling interval must match the interval configured in the Creating VolumeReplicationClass resource section.

6. Verify that the **DRPolicy** is created successfully by running the command:

```
$ oc get drpolicy odr-policy-5m -n openshift-dr-system -o
jsonpath='{.status.conditions[].reason}{"\n"}'
```

Example output:

```
Succeeded
```

CHAPTER 9. CREATING A SAMPLE APPLICATION

You need a sample application to test failover from the Primary managed cluster to the Secondary managed cluster and back again. 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-5m
  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**.

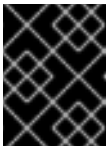
```
apiVersion: apps.open-cluster-management.io/v1
kind: PlacementRule
metadata:
  labels:
    app: busybox-sample
  name: busybox-placement
spec:
  clusterConditions:
  - status: "True"
    type: ManagedClusterConditionAvailable
  clusterReplicas: 1
  schedulerName: ramen
```

   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**.

IMPORTANT

Make sure that the new **StorageClass ocs-storagecluster-ceph-rbdmirror** is created as detailed in section [Create Mirroring StorageClass resource] before proceeding.

Verify that it is created using the following command:

```
oc get storageclass | grep rbdmirror | awk '{print $1}'
```

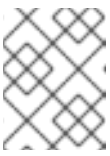
Example output:

```
ocs-storagecluster-ceph-rbdmirror
```

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**.
When the application is created, the application details page is displayed. You can scroll down to Resource topology section. There should be Green check marks on the elements and application in the topology.



NOTE

To view more information, click on any of the topology elements and a window will appear in the right of the topology view.

5. Verify 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. Logon 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 volumereplication,volumereplicationgroup -n busybox-sample
```

Example output:

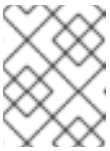
```
NAME                                                                AGE   VOLUMEREPLICATIONCLASS
PVCNAME       DESIREDSTATE   CURRENTSTATE
volumereplication.replication.storage.openshift.io/busybox-pvc   6m    odf-rbd-
volumereplicationclass   busybox-pvc   primary        Primary

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

c. Verify that the **busybox** volume has been replicated to the alternate cluster by running the following command on both the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

```
{"daemon_health":"OK","health":"OK","image_health":"OK","states":{"replaying":2}}
```



NOTE

Both managed clusters should have the exact same output with a new status of **"states":{"replaying":2}**`.

9.1. DELETING SAMPLE APPLICATION

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

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. Logon 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.

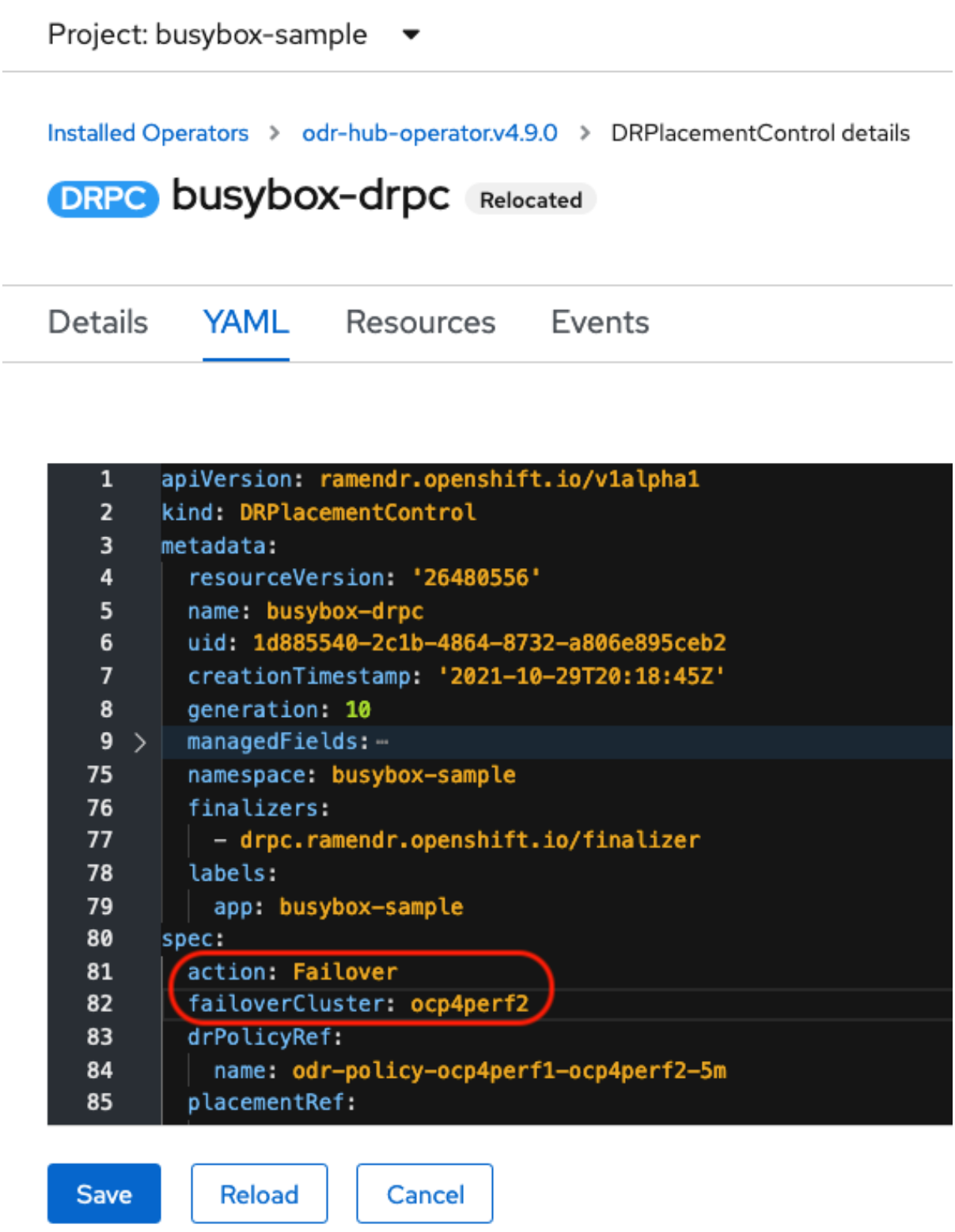
CHAPTER 10. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

This section provides instructions on how to failover the busybox sample application. The failover method for Regional-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. On the Hub cluster navigate to Installed Operators and then click **Openshift DR Hub Operator**.
2. Click **DRPlacementControl** tab.
3. Click DRPC **busybox-drpc** and then the YAML view.
4. 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



5. Click **Save**.

6. 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
```

7. 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 Regional-DR issues as documented in Known Issues section of Release Notes.

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 availability and work schedules.

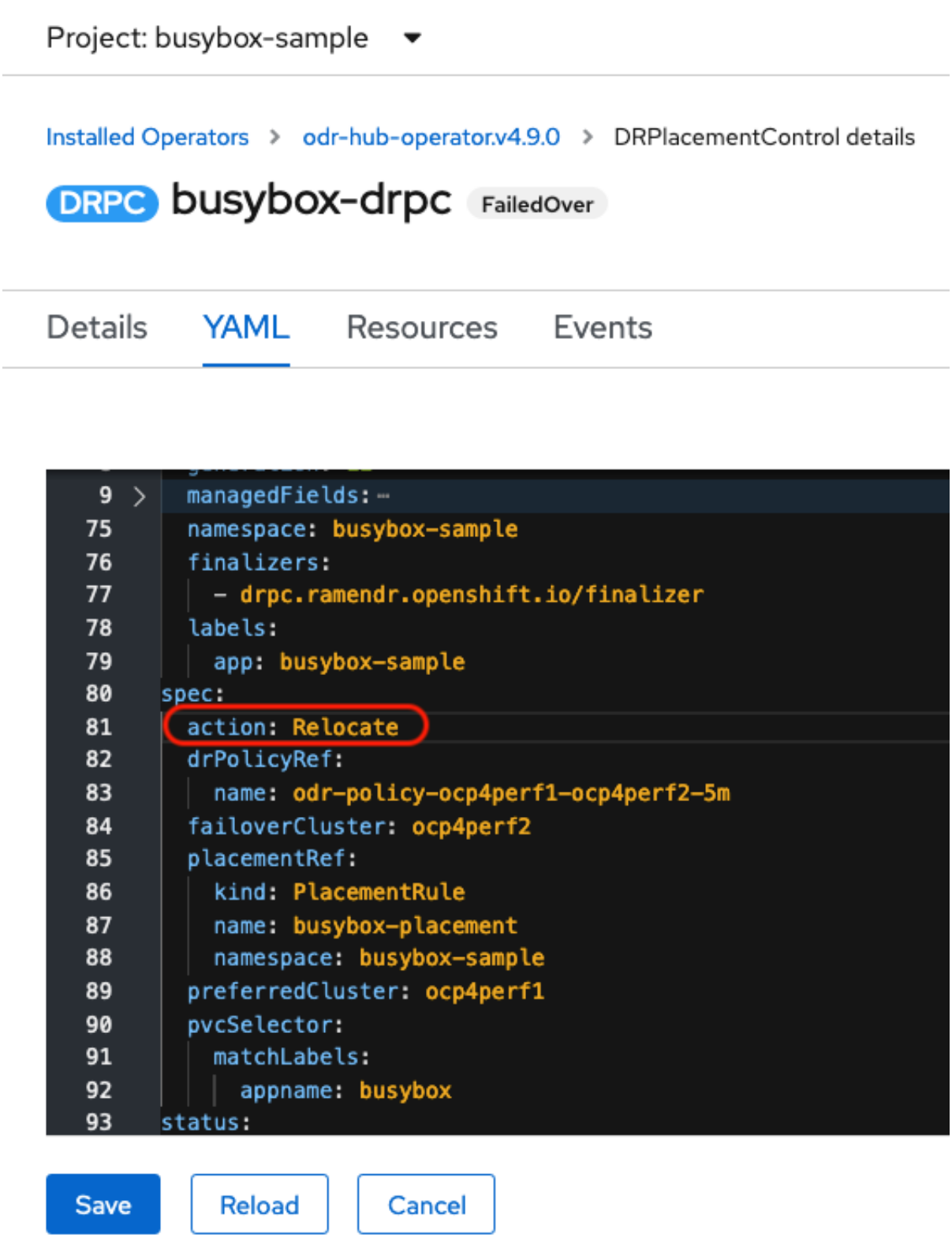
CHAPTER 11. RELOCATING AN APPLICATION BETWEEN MANAGED CLUSTERS

A relocation operation is very similar to failover. Relocate is application based and uses the DRPlacementControl to trigger the relocation. The main difference for relocation is that a **resync** is issued to make sure any new application data saved on the Secondary managed cluster is immediately, not waiting for the mirroring schedule interval, replicated to the Primary managed cluster.

Procedure

1. On the Hub cluster navigate to Installed Operators and then click **Openshift DR Hub Operator**.
2. Click **DRPlacementControl** tab.
3. Click DRPC **busybox-drpc** and then the YAML view.
4. Modify **action** to **Relocate**

   DRPlacementControl modify action to Relocate



5. Click **Save**.

6. Verify if the application **busybox** is now running in the Primary managed cluster, 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          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
```

7. 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 Regional-DR issues as documented in Known Issues section of Release Notes.

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 availability and work schedules.