Red Hat OpenShift Data Foundation 4.10

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

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

Last Updated: 2023-11-08

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

DEVELOPER PREVIEW: Instructions about setting up OpenShift Data Foundation with Regional-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 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, 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 similar 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

Red Hat Advanced Cluster Management 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: includes component that run on the multi-cluster control plane.
- Managed clusters: includes components that run on the clusters that are managed.

For more information about this product, see RHACM documentation and the RHACM “Managing 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 now enhanced with the following abilities:

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

- Provides csi-addons to manage per Persistent Volume Claim (PVC) 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 three components:

- **ODF Multicluster Orchestrator**: Installed on the multi-cluster control plane (RHACM Hub), it also performs the following actions:
  - Creates a bootstrap token and exchanges this token between the managed clusters.
  - Enables mirroring for the default **CephBlockPool** on the managed clusters.
  - Creates an object bucket using Multicloud Object Gateway (MCG) on each managed cluster for **PVC** and **PV** metadata.
  - Creates a **Secret** for each new object bucket that has the keys for bucket access on the **Hub cluster** in the **openshift-dr-system** project.
  - Creates a **VolumeReplicationClass** on the **Primary managed cluster** and the **Secondary managed cluster** for each **schedulingIntervals** (e.g. 5m, 15m, 30m).
  - Modifies the **ramen-hub-operator-config** ConfigMap on the Hub cluster and adds the s3StoreProfiles entries.
- **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 4.10 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 Regional-DR requirements which includes RHACM operator installation, creation or importing of OpenShift Container Platform into RHACM hub and network configuration. See Requirements for enabling Regional-DR.
2. Install OpenShift Data Foundation 4.10 on Primary and Secondary managed clusters. See Installing OpenShift Data Foundation on managed clusters.

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

4. Configure multisite storage replication by creating the mirroring relationship between two OpenShift Data Foundation managed clusters. See Configuring multisite storage replication .

5. 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.

6. 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 .

    NOTE

    There can be more than a single policy.

7. 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.

8. 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 REGIONAL-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), 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 RHACM operator and MultiClusterHub is installed on the Hub cluster. See RHACM installation guide for instructions.
  - 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 have either imported or created the **Primary managed cluster** and the **Secondary managed clusters** using the RHACM console.
- The managed clusters must have non-overlapping networks.
  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.

CHAPTER 3. INSTALLING OPENSHIFT DATA FOUNDATION ON MANAGED CLUSTERS

Procedure

1. Install OpenShift Data Foundation version 4.10 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).

2. 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"}'
   ```

   and for the Multicloud Object 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 on to enabling mirroring on the managed clusters.

CHAPTER 4. 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 5. 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.

5.1. 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** has installed successfully.
   a. Validate successful installation by having the ability to select View Operator.
   b. Verify that the operator Pod are in **Running** state.

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

      Example output:

      ```
      NAME                                    READY   STATUS    RESTARTS   AGE
      odfmo-controller-manager-65946fb99b-779v8   1/1     Running   0          5m3s
      ```

5.2. 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. Copy and save 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
  manageS3: true
  schedulingIntervals:
  - 5m
  - 15m
```

   NOTE

   The time values (e.g. 5m) for **schedulingIntervals** will be used to configure the desired interval for replicating persistent volumes. These values can be mapped to your Recovery Point Objective (RPO) for critical applications. Modify the values in **schedulingIntervals** to be correct for your application requirements. The minimum value is **1m** and the default is **5m**.

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

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

6. Verify that you can view **Phase** status as **ExchangedSecret** before proceeding.

5.3. VALIDATING CEPH MIRRORING ON MANAGED CLUSTERS

Perform the following validations on the **Primary managed cluster** and the **Secondary managed cluster** to check Ceph mirroring is active:

1. Verify 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
```

2. Verify 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
```

3. Check the status of the **daemon** health to ensure it is OK.

```
$ 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 become OK after 10 minutes then use the Advanced Cluster Manager console to verify that the **submariner add-on** connection is still in a healthy state.

4. Verify that **VolumeReplicationClass** is created on the **Primary managed cluster** and the **Secondary managed cluster** for each schedulingIntervals listed in the MirrorPeer (e.g. 5m, 15m).

```
$ oc get volumereplicationclass
```

Example output:

```
NAME                                    PROVISIONER
rbd-volumereplicationclass-1625360775   openshift-storage.rbd.csi.ceph.com
rbd-volumereplicationclass-539797778    openshift-storage.rbd.csi.ceph.com
```

NOTE

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.

5.4. VALIDATING OBJECT BUCKETS AND S3STOREPROFILES

Perform the following validations on the **Primary managed cluster** and the **Secondary managed cluster** to check Ceph mirroring is active.

Procedure

1. Verify that there is a new **Object Bucket Claim** and corresponding **Object Bucket** in the **Primary managed cluster** and the **Secondary managed cluster** in the **openshift-storage** namespace.

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

Example output:

```
NAME                                                   STORAGE-CLASS                 PHASE   AGE
objectbucketclaim.objectbucket.io/odrbucket-21eb5332f6b6   openshift-storage.noobaa.io   Bound   13m

NAME                                                         STORAGE-CLASS                 CLAIM-NAMESPACE   CLAIM-NAME   RECLAIM-POLICY   PHASE   AGE
objectbucket.objectbucket.io/obc-openshift-storage-odrbucket-21eb5332f6b6   openshift-storage.noobaa.io                    Delete           Bound   13m
```

2. Verify that there are two new **Secrets** in the **Hub cluster** **openshift-dr-system** namespace that contain the access and secret key for each new Object Bucket Class.

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

Example output:

```
8b3fb9ed90f66808d988c7edfa76eba35647092   Opaque  2      16m
af5f82f21f8f77faf3de2553e223b535002e480   Opaque  2      16m
```

3. The OBC and Secrets are written in the ConfigMap **ramen-hub-operator-config** on the Hub cluster in the newly created **s3StoreProfiles** section.

```
$ oc get cm ramen-hub-operator-config -n openshift-dr-system -o yaml | grep -A 14 s3StoreProfiles
```

Example output:

```
s3StoreProfiles:
- s3Bucket: odrbucket-21eb5332f6b6
  s3CompatibleEndpoint: https://s3-openshift-storage.apps.perf2.example.com
  s3ProfileName: s3profile-ocp4perf2-ocs-storagecluster
  s3Region: noobaa
  s3SecretRef:
    name: 8b3fb9ed90f66808d988c7edfa76eba35647092
    namespace: openshift-dr-system
- s3Bucket: odrbucket-21eb5332f6b6
  s3CompatibleEndpoint: https://s3-openshift-storage.apps.perf1.example.com
  s3ProfileName: s3profile-ocp4perf1-ocs-storagecluster
  s3Region: noobaa
```

```
s3SecretRef:
  name: af5f82f21f8f77faf3de2553e223b535002e480
  namespace: openshift-dr-system
```

NOTE

Record the names of the **s3ProfileName**. They will be used in the DRPolicy resource.

CHAPTER 6. CREATING MIRRORING STORAGECLASS RESOURCE

You must create the block volumes with **mirroring** enabled 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**.

```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 7. 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 the **Hub cluster** for verifying 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 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
```

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 ACM. Replace *<string_value_1>* and *<string_value_2>* with any values as long as they are unique (for example: east and west). The **schedulingInterval** should be one of the values configured in the **MirrorPeer** earlier (for example: 5m).

```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRPolicy
metadata:
  name: odr-policy-5m
spec:
  drClusterSet:
  - name: <cluster1>
    region: <string_value_1>
    s3ProfileName: s3profile-<cluster1>-ocs-storagecluster
  - name: <cluster2>
    region: <string_value_2>
    s3ProfileName: s3profile-<cluster2>-ocs-storagecluster
  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 schedulingInterval** ***must*** match one of the values configured in **MirroPeer** resource (e.g. 5m). To use one of the other **schedulingIntervals** for volume replication configured in the **MirrorPeer** requires creating additional **DRPolicy** resources with the new values (i.e., 15m). Make sure to change the **DRPolicy name** to be unique and useful in identifying the replication interval (e.g. odr-policy-15m).

6. Verify that the **DRPolicy** is created successfully by running the command on the **Hub cluster** for each **DRPolicy** resource created. This example is for **odr-policy-5m**:

   ```
   $ oc get drpolicy odr-policy-5m -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 to add **deploymentAutomationEnabled=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  ## <-- Add to enable installation of ODR Cluster
   operator on managed clusters
         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 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. Save the following YAML to filename **busybox-drpc.yaml** after replacing *<cluster1>* with the correct name of your managed cluster in Advanced Cluster Manager. Modify **drPolicyRef** *name* for the **DRPolicy** that has the desired replication interval.

    ```
    apiVersion: ramendr.openshift.io/v1alpha1
    kind: DRPlacementControl
    metadata:
      labels:
        app: busybox-sample
      name: busybox-drpc
    spec:
      drPolicyRef:
        name: odr-policy-5m     ## <-- Modify to specify desired DRPolicy and RPO
      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**.
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. 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. 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 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}**`.

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.

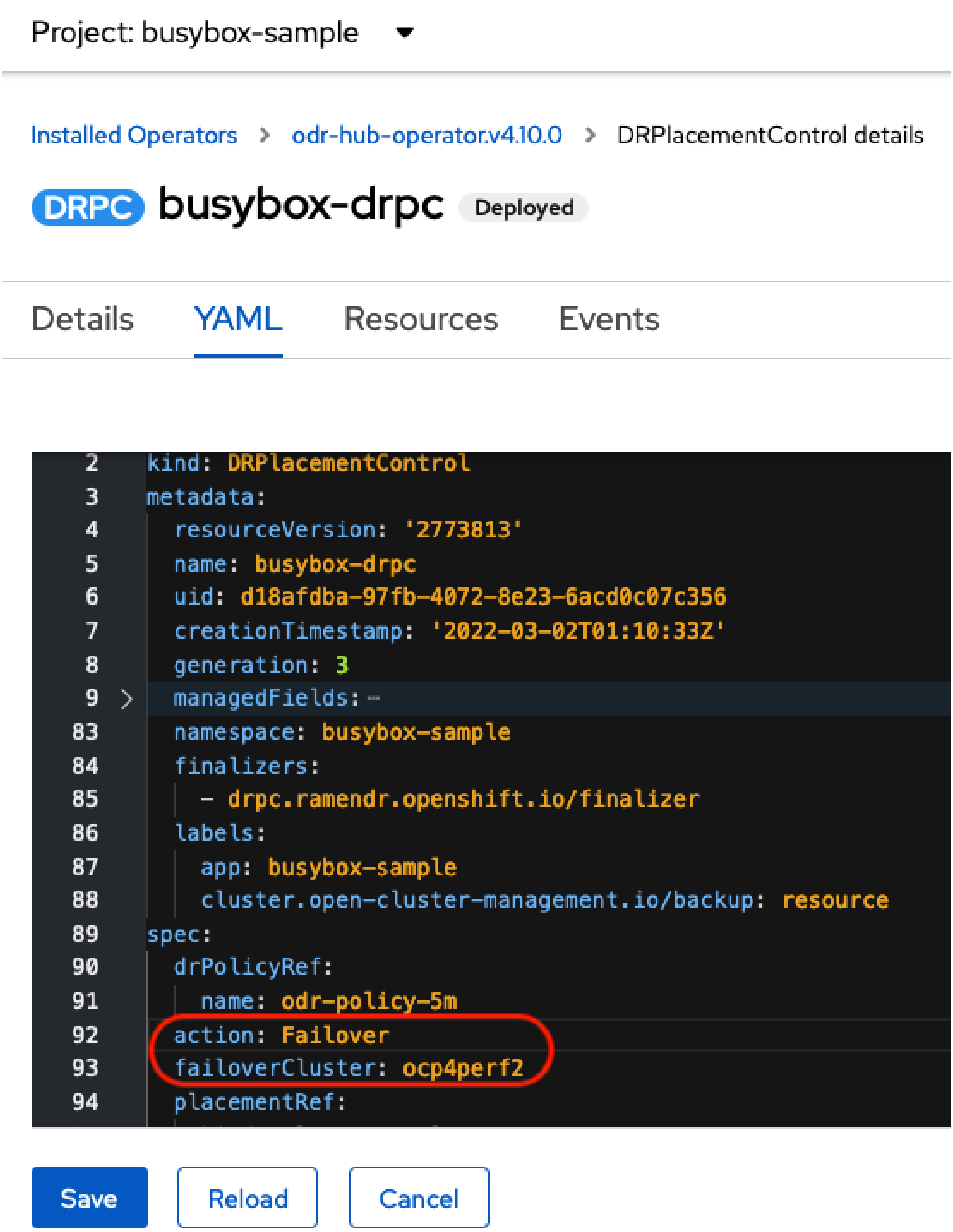
CHAPTER 12. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

This section provides instructions on how to failover the busybox sample application. The failover method for 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.

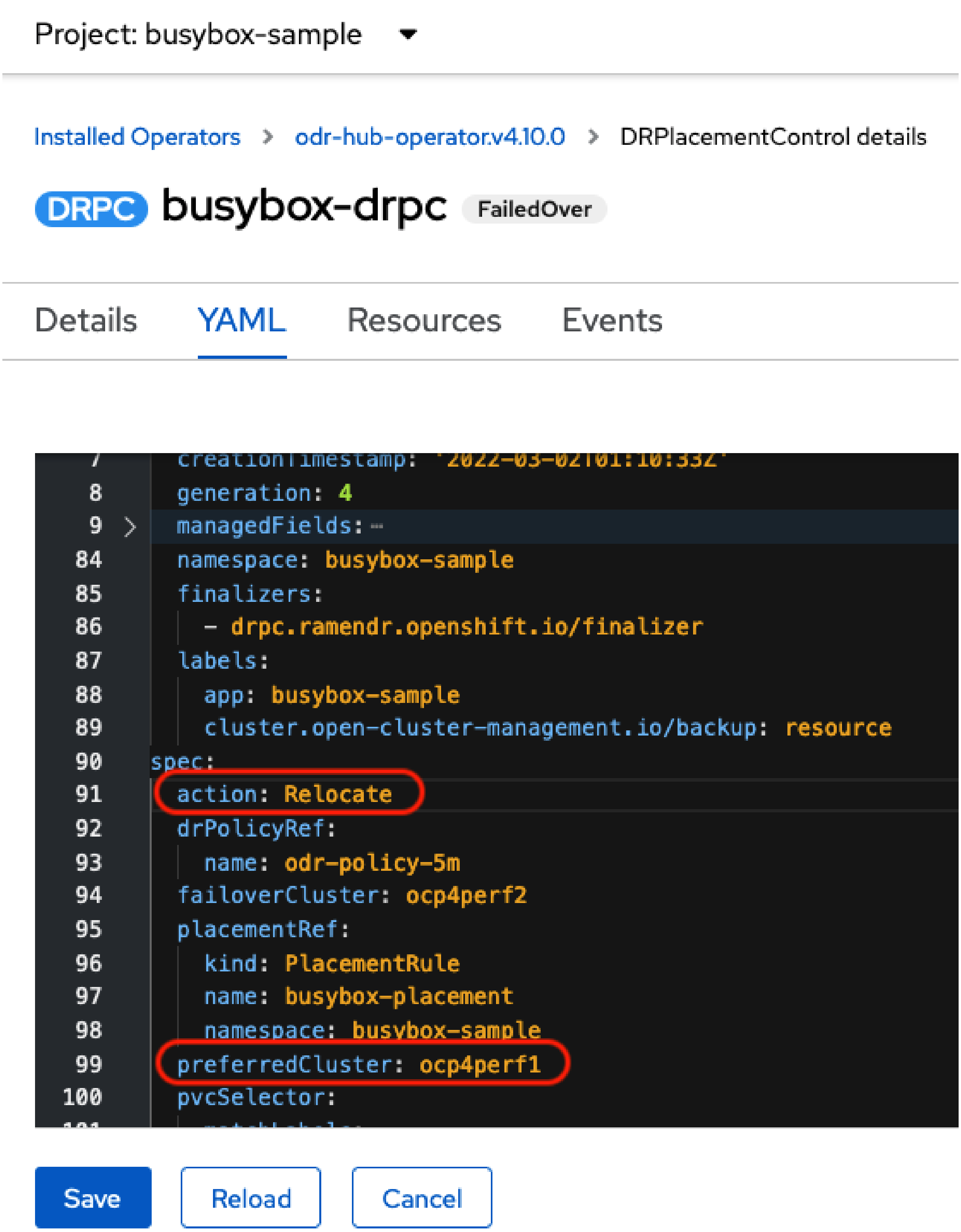
CHAPTER 13. RELOCATING AN APPLICATION BETWEEN MANAGED CLUSTERS

A relocation operation is very similar to failover. Relocate is application based and uses the DRPlacementControl to trigger the relocation. The main difference for 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.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
```

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.