



Red Hat OpenShift Data Foundation 4.11

Configuring OpenShift Data Foundation Disaster Recovery for OpenShift Workloads

TECHNOLOGY PREVIEW: This solution is a technology preview feature and is not intended to be run in production environments.

Red Hat OpenShift Data Foundation 4.11 Configuring OpenShift Data Foundation Disaster Recovery for OpenShift Workloads

TECHNOLOGY PREVIEW: This solution is a technology preview feature and is not intended to be run in production environments.

Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

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 Disaster Recovery solution is a Technology Preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	4
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	5
CHAPTER 1. INTRODUCTION TO OPENSIFT DATA FOUNDATION DISASTER RECOVERY	6
CHAPTER 2. DISASTER RECOVERY SUBSCRIPTION REQUIREMENT	7
CHAPTER 3. REGIONAL-DR SOLUTION FOR OPENSIFT DATA FOUNDATION	8
3.1. COMPONENTS OF REGIONAL-DR SOLUTION	8
3.2. REGIONAL-DR DEPLOYMENT WORKFLOW	9
3.3. REQUIREMENTS FOR ENABLING REGIONAL-DR	9
3.4. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER ON MANAGED CLUSTERS	11
3.5. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR	12
3.6. CONFIGURING SSL ACCESS ACROSS CLUSTERS	13
3.7. ENABLING MULTICLUSTER WEB CONSOLE	14
3.8. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER	15
3.9. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION	18
3.9.1. Creating a sample application	18
3.9.2. Apply DRPolicy to sample application	20
3.9.3. Deleting sample application	21
3.10. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	22
3.10.1. Modify DRPlacementControl to failover	22
3.11. RELOCATING AN APPLICATION BETWEEN MANAGED CLUSTERS	24
3.11.1. Modify DRPlacementControl to Relocate	25
CHAPTER 4. METRO-DR SOLUTION FOR OPENSIFT DATA FOUNDATION	28
4.1. COMPONENTS OF METRO-DR SOLUTION	28
4.2. METRO-DR DEPLOYMENT WORKFLOW	29
4.3. REQUIREMENTS FOR ENABLING METRO-DR	29
4.4. REQUIREMENTS FOR DEPLOYING RED HAT CEPH STORAGE STRETCH CLUSTER WITH ARBITER	31
4.4.1. Hardware requirements	31
4.4.2. Software requirements	31
4.4.3. Network configuration requirements	32
4.5. DEPLOYING RED HAT CEPH STORAGE	32
4.5.1. Node pre-deployment steps	32
4.5.2. Cluster bootstrapping and service deployment with Cephadm	35
4.5.3. Configuring Red Hat Ceph Storage stretch mode	42
4.6. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER ON MANAGED CLUSTERS	46
4.7. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR	48
4.8. CONFIGURING SSL ACCESS ACROSS CLUSTERS	49
4.9. ENABLING MULTICLUSTER WEB CONSOLE	51
4.10. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER	51
4.11. CONFIGURE DRCLUSTERS FOR FENCING AUTOMATION	54
4.11.1. Add node IP addresses to DRClusters	54
4.11.2. Add fencing annotations to DRClusters	55
4.12. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION	55
4.12.1. Creating a sample application	56
4.12.2. Apply DRPolicy to sample application	57
4.12.3. Deleting sample application	58
4.13. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	59
4.13.1. Enable fencing	59

4.13.2. Modify DRPlacementControl to failover	60
4.14. RELOCATING AN APPLICATION BETWEEN MANAGED CLUSTERS	63
4.14.1. Disable fencing	64
4.14.2. Modify DRPlacementControl to Relocate	65
CHAPTER 5. TROUBLESHOOTING DISASTER RECOVERY	69
5.1. TROUBLESHOOTING REGIONAL-DR	69
5.1.1. RBD mirroring scheduling is getting stopped for some images	69
5.1.2. Relocation failure	69
5.1.3. rbd-mirror daemon health is in warning state	70
5.1.4. statefulset application stuck after failover	70
5.1.5. Application is not running after failover	70
5.2. TROUBLESHOOTING METRO-DR	71
5.2.1. A statefulset application stuck after failover	71
5.2.2. DR policies protect all applications in the same namespace	71
5.2.3. During failback of an application stuck in Relocating state	72

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 OPENSIFT DATA FOUNDATION DISASTER RECOVERY

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

The OpenShift Data Foundation DR capability enables DR across multiple Red Hat OpenShift Container Platform clusters, and is categorized as follows:

- **Metro-DR**
Metro-DR ensures business continuity during the unavailability of a data center with no data loss. In the public cloud these would be similar to protecting from an Availability Zone failure.
- **Regional-DR**
Regional-DR ensures business continuity during the unavailability of a geographical region, accepting some loss of data in a predictable amount. In the public cloud this would be similar to protecting from a region failure.

Zone failure in Metro-DR and region failure in Regional-DR is usually expressed using the terms, **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 are notified of a business disruption?"

The intent of this guide is to detail the Disaster Recovery steps and commands necessary to be able to failover an application from one OpenShift Container Platform cluster to another and then relocate the same application to the original primary cluster.

CHAPTER 2. DISASTER RECOVERY SUBSCRIPTION REQUIREMENT

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

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

Any Red Hat OpenShift Data Foundation Cluster containing PVs participating in active replication either as a source or destination requires OpenShift Data Foundation Advanced entitlement. This subscription should be active on both source and destination clusters.

To know how subscriptions for OpenShift Data Foundation work, see [knowledgebase article on OpenShift Data Foundation subscriptions](#).

CHAPTER 3. REGIONAL-DR SOLUTION FOR OPENSIFT DATA FOUNDATION

3.1. COMPONENTS OF REGIONAL-DR SOLUTION

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

Red Hat Advanced Cluster Management for Kubernetes

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

RHACM is split into two parts:

- RHACM Hub: includes components 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 “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 now enhanced with the following abilities for disaster recovery:

- Enable RBD block pools for mirroring across OpenShift Data Foundation instances (clusters)
- Ability to mirror specific images within an RBD block pool
- Provides csi-addons to manage per Persistent Volume Claim (PVC) mirroring

OpenShift DR

OpenShift DR is a set of orchestrators to configure and manage stateful applications across a set of peer OpenShift clusters which are 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 and its state relationship across OpenShift clusters
- Failing over an application and its state to a peer cluster
- Relocate an application and its 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 orchestrates configuration and peering of OpenShift Data Foundation clusters for Metro and Regional DR relationships

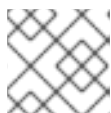
- **OpenShift DR Hub Operator:** Automatically installed as part of ODF Multicluster Orchestrator installation on the hub cluster to orchestrate failover or relocation of DR enabled applications.
- **OpenShift DR Cluster Operator:** Automatically installed on each managed cluster that is part of a Metro and Regional DR relationship to manage the lifecycle of all PVCs of an application.

3.2. REGIONAL-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Regional-DR capabilities using latest version of Red Hat OpenShift Data Foundation 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 Red Hat Advanced Cluster Management (RHACM).

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

1. Ensure requirements across the three: Hub, Primary and Secondary OpenShift Container Platform clusters that are part of the DR solution are met. See [Requirements for enabling Regional-DR](#).
2. Install OpenShift Data Foundation operator and create a storage system on Primary and Secondary managed clusters. See [Creating OpenShift Data Foundation cluster on managed clusters](#).
3. Install the ODF Multicluster Orchestrator on the Hub cluster. See [Installing ODF Multicluster Orchestrator on Hub cluster](#).
4. Configure SSL access between the Hub, Primary and Secondary clusters. See [Configuring SSL access across clusters](#).
5. Enable the Web Console. [Enabling Multicluster Web Console](#)
6. Create a DRPolicy resource for use with applications requiring DR protection across the Primary and Secondary clusters. See [Creating Disaster Recovery Policy on Hub cluster](#).



NOTE

There can be more than a single policy.

For testing your disaster recovery solution:

- Create a sample application using RHACM console. See [Creating sample application](#).
- Test failover and relocate operations using the sample application between managed clusters. See [application failover](#) and [relocating an application](#).

3.3. 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:

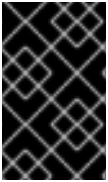
- You must have three OpenShift clusters that have network reachability between them:
 - **Hub cluster** where Red Hat Advanced Cluster Management for Kubernetes (RHACM operator) is installed.

- **Primary managed cluster** where OpenShift Data Foundation is installed.
- **Secondary managed cluster** where OpenShift Data Foundation is 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 RHACM console:


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

Example Output:

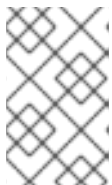
```
https://multicloud-console.apps.perf3.example.com/multicloud/clusters
```
 - Open your output link in a browser to login with OpenShift credentials. You should now see your local-cluster imported.



IMPORTANT

It is the user's responsibility to ensure that application traffic routing and redirection are configured appropriately. Configuration and updates to the application traffic routes are currently not supported.

- Ensure that you have either imported or created the **Primary managed cluster** and the **Secondary managed cluster** using the RHACM console. For instructions, see [Creating a cluster](#) and [Importing a target managed cluster to the hub cluster](#).
- 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.



NOTE

Version 0.12 of Submariner installed using RHACM 2.5 cluster add-ons, does not support the OpenShift OVNKubernetes CNI plugin. See [RHACM 2.5 Release Notes](#).

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

Example output for Primary cluster:

```
{
  "clusterNetwork": [
    {
      "cidr": "10.5.0.0/16",
      "hostPrefix": 23
    }
  ],
}
```

```

"externalIP": {
  "policy": {}
},
"networkType": "OpenShiftSDN",
"serviceNetwork": [
  "10.15.0.0/16"
]
}

```

Example output for Secondary cluster:

```

{
  "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](#).

CAUTION

Do not select **Enable Globalnet** because of overlapping cluster and service networks for the managed clusters. Using **Globalnet** is not supported with Regional Disaster Recovery currently. Ensure that cluster and service networks are non-overlapping before proceeding.

3.4. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER ON MANAGED CLUSTERS

In order to configure storage replication between the two OpenShift Container Platform clusters, create an OpenShift Data Foundation storage system after you install the OpenShift Data Foundation operator.



NOTE

Refer to OpenShift Data Foundation deployment guides and instructions that are specific to your infrastructure (AWS, VMware, BM, Azure, etc.).

Procedure

1. Install and configure the latest **OpenShift Data Foundation** cluster 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 of OpenShift Data Foundation on each managed cluster with the following command:

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

For the Multicloud Gateway (MCG):

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

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



NOTE

In the OpenShift Web Console, navigate to **Installed Operators** → **OpenShift Data Foundation** → **Storage System** → **ocs-storagecluster-storagesystem** → **Resources** and verify that **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.

3.5. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multiclustler Orchestrator is a controller that is installed from OpenShift Container Platform's OperatorHub on the Hub cluster.

Procedure

1. On the **Hub cluster**, navigate to **OperatorHub** and use the keyword filter to search for **ODF Multiclustler Orchestrator**.
2. Click **ODF Multiclustler Orchestrator** tile.
3. Keep all default settings and click **Install**.
Ensure that the operator resources are installed in **openshift-operators** project and available to all namespaces.



NOTE

The **ODF Multiclustler Orchestrator** also installs the **OpenShift DR Hub Operator** on the RHACM hub cluster as a dependency.

4. Verify that the operator **Pods** are in a **Running** state. The **OpenShift DR Hub operator** is also installed at the same time in **openshift-operators** namespace.

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

Example output:

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
odf-multicloud-console-6845b795b9-blxrn	1/1	Running	0	4d20h
odfmo-controller-manager-f9d9dfb59-jbrsd	1/1	Running	0	4d20h
ramen-hub-operator-6fb887f885-fss4w	2/2	Running	0	4d20h

3.6. CONFIGURING SSL ACCESS ACROSS CLUSTERS

Configure network (SSL) access between the **primary and secondary clusters** so that metadata can be stored on the alternate cluster in a Multicloud Gateway (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

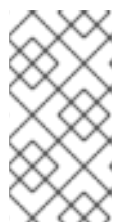
1. Extract the ingress certificate for the Primary managed cluster and save the output to **primary.crt**.

```
$ 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** file to hold the remote cluster's certificate bundle with filename **cm-clusters.crt.yaml**.



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** 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
```

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
```

3.7. ENABLING MULTICLUSTER WEB CONSOLE

This is a new capability that is required before creating a Data Policy or **DRPolicy**. It is only needed on the **Hub cluster** and **RHACM 2.5** must be installed.



IMPORTANT

Multicluster console is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Procedure

1. Navigate to **Administration** → **Cluster Settings** → **Configuration** → **FeatureGate**.
2. Edit the YAML template as follows:

```
[...]
spec:
  featureSet: TechPreviewNoUpgrade
```

3. Click **Save** to enable the multicluster console for all clusters in the RHACM console. Wait for the Nodes to become **Ready**.
4. Refresh the web console and verify that the managed cluster names are listed below **All Clusters**.



WARNING

Do not set this feature gate on production clusters. You will not be able to upgrade your cluster after applying the feature gate, and it cannot be undone.

3.8. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER

OpenShift Disaster Recovery Policy (DRPolicy) resource specifies OpenShift Container Platform clusters participating in the disaster recovery solution and the desired replication interval. DRPolicy is a cluster scoped resource that users can apply to applications that require Disaster Recovery solution.

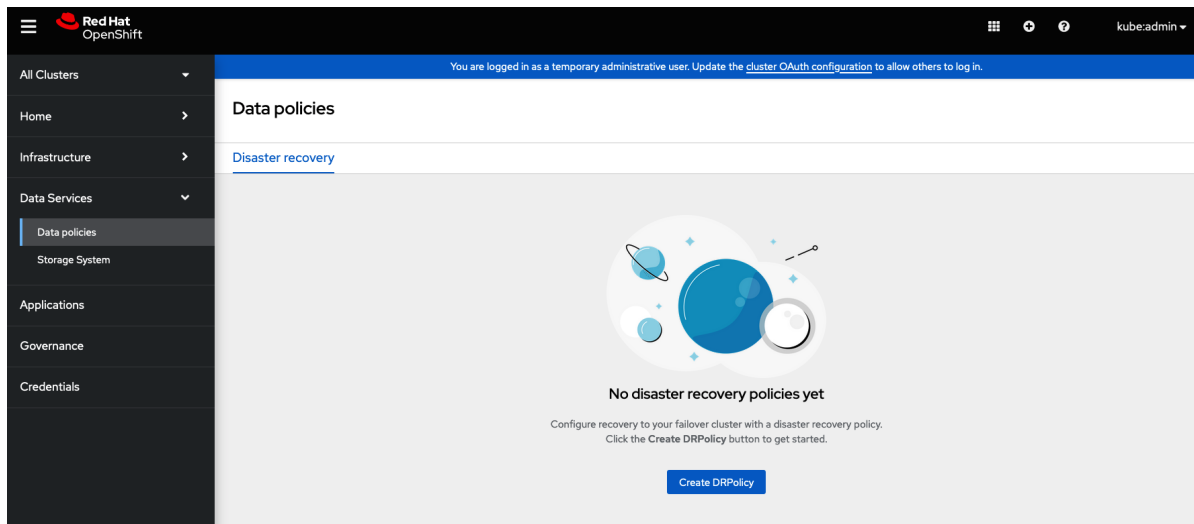
The **ODF MultiCluster Orchestrator** Operator facilitates the creation of each **DRPolicy** and the corresponding **DRClusters** through the **Multicluster Web console**.

Prerequisites

- Ensure that there is a minimum set of two managed clusters.
- Make sure to login to all the clusters from the **Multicluster Web console**.
 - Click on **All Clusters** to expand the list of managed clusters.
 - For each managed cluster listed below **All Clusters**, click on the `<cluster_name>` and then wait for a login screen to appear, where you can login using the credentials of the cluster that you have selected.

Procedure

1. On the **OpenShift console**, navigate to **All Clusters**.



2. Navigate to **Data Services** and click **Data policies**.
3. Click **Create DRPolicy**.
4. Enter **Policy name**. Ensure that each DRPolicy has a unique name (for example: **ocp4bos1-ocp4bos2-5m**).
5. Select two clusters from the list of managed clusters to which this new policy will be associated with.
6. **Replication policy** is automatically set to **Asynchronous**(async) based on the OpenShift clusters selected and a **Sync schedule** option will become available.
7. Set **Sync schedule**.



IMPORTANT

For every desired replication interval a new **DRPolicy** must be created with a unique name (such as: **ocp4bos1-ocp4bos2-10m**). The same clusters can be selected but the **Sync schedule** can be configured with a different replication interval in minutes/hours/days. The minimum is one minute.

8. Click **Create**.
9. Verify that the **DRPolicy** is created successfully. Run this command on the **Hub cluster** for each of the DRPolicy resources created.



NOTE

Replace <drpolicy_name> with your unique name.

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

Example output:

```
Succeeded
```

**NOTE**

When a DRPolicy is created, along with it, two DRCluster resources are also created. It could take up to 10 minutes for all three resources to be validated and for the status to show as **Succeeded**.

10. Verify the object bucket access from the **Hub cluster** to both the **Primary managed cluster** and the **Secondary managed cluster**.

- a. Get the names of the **DRClusters** on the Hub cluster.

```
$ oc get drclusters
```

Example output:

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

- b. Check S3 access to each bucket created on each managed cluster using this **DRCluster** validation command.

**NOTE**

Replace <drcluster_name> with your unique name.

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

Example output:

```
Succeeded
```

**NOTE**

Make sure to run command for both **DRClusters** on the **Hub cluster**.

11. Verify that the **OpenShift DR Cluster operator** installation was successful on the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

```
NAME                                     DISPLAY          VERSION
REPLACES PHASE
clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.11.0  Openshift DR
Cluster Operator  4.11.0          Succeeded

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

You can also verify that **OpenShift DR Cluster Operator** is installed successfully on the **OperatorHub** of each managed clusters.

12. Verify that the status of the ODF mirroring **daemon** health on the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

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

CAUTION

It could take up to 10 minutes for the **daemon_health** and **health** to go from **Warning** to **OK**. If the status does not become **OK** eventually then use the RHACM console to verify that the Submariner connection between managed clusters is still in a healthy state. Do not proceed until all values are **OK**.

3.9. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION

OpenShift Data Foundation disaster recovery (DR) solution supports disaster recovery for applications that are managed by RHACM. See [Managing Applications](#) for more details.

This solution orchestrates RHACM application placement, using the [PlacementRule](#), when an application is moved between clusters in a DRPolicy for failover or relocation requirements.

The following sections detail how to apply a **DRPolicy** to an application and how to manage the applications placement life-cycle during and after cluster unavailability.



NOTE

OpenShift Data Foundation DR solution does not support ApplicationSet, which is required for applications that are deployed via ArgoCD.

3.9.1. Creating a sample application

In order to test **failover** from the **Primary managed cluster** to the **Secondary managed cluster** and **relocate**, we need a simple application.

Prerequisites

- When creating an application for general consumption, ensure that:
 - the application is deployed to ONLY one cluster.
 - the application is deployed prior to applying the DRPolicy to the application.
- Use the sample application called **busybox** as an example.

- Ensure all external routes of the application are configured using either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service for traffic redirection when the application fails over or is relocated.

Procedure

1. 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
```

2. Navigate to **Applications** and click **Create application**.
3. Select type as **Subscription**.
4. Enter your application **Name** (for example, **busybox**) and **Namespace** (for example, **busybox-sample**).
5. In the Repository location for resources section, select **Repository type Git**.
6. 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/red-hat-storage/ocm-ramen-samples/tree/release-4.11> where the **Branch** is **main** and **Path** is **busybox-odr**.
7. Scroll down in the form until you see **Deploy application resources only on clusters matching specified labels** and then add a label with its value set to the **Primary managed cluster** name in RHACM cluster list view.

Deploy application resources only on clusters matching specified labels

Cluster labels

Enter one or more matching labels to select the clusters to deploy to

Label *	Value *
name	ocp4bos1

[+](#) **Add another label**

8. Click **Create** which is at the top right hand corner.
On the follow-on screen go to the **Topology** tab. 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.

- Validating the sample application deployment.

Now that the **busybox** application has been deployed to your preferred Cluster, the deployment can be validated.

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
5Gi   RWO          ocs-storagecluster-ceph-rbd 6m
```

3.9.2. Apply DRPolicy to sample application

- On the Hub cluster go back to the Multicluster Web console, navigate to **All Clusters**.
- Login to all the clusters listed under **All Clusters**.
- Navigate to **Data Services** and then click **Data policies**.
- Click the Actions menu at the end of DRPolicy to view the list of available actions.

The screenshot shows the Red Hat OpenShift Multicluster Web console interface. The left sidebar contains navigation options: All Clusters, Home, Infrastructure, Data Services, Applications, Governance, and Credentials. The main content area is titled 'Data policies' and includes a 'Disaster recovery' link. A search bar is present above a table of data policies. The table has columns for Name, Status, Clusters, Replication policy, and Connected applications. One policy is listed with Name 'ocp4bos1-ocp4bos2-5m', Status 'Validated', Clusters 'ocp4bos1, ocp4bos2', Replication policy 'async', and Connected applications '0 Applications'. A three-dot menu icon is visible at the end of the row, and a dropdown menu is open, showing the 'Apply DRPolicy' option circled in red.

- Click **Apply DRPolicy**.
- When the **Apply DRPolicy** modal is displayed, select **busybox** application and enter **PVC label** as **appname=busybox**.

**NOTE**

When multiple placements rules under the same application or more than one application are selected, all PVCs within the application's namespace will be protected by default.

7. Click **Apply**.
8. Verify that a **DRPlacementControl** or **DRPC** was created in the **busybox-sample** namespace on the **Hub cluster** and that its **CURRENTSTATE** shows as **Deployed**. This resource is used for both failover and relocate actions for this application.

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

Example output:

NAME	AGE	PREFERREDCLUSTER	FAILOVERCLUSTER
DESIREDSTATE	CURRENTSTATE		
busybox-placement-1-drpc	6m59s	ocp4bos1	Deployed

3.9.3. 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 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 the **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 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.

3.10. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

A failover is performed when a managed cluster becomes unavailable, due to any reason.

This section provides instructions on how to failover the busybox sample application. The failover method is application based. Each application that is to be protected in this manner must have a corresponding **DRPlacementControl** resource in the application namespace.

3.10.1. Modify DRPlacementControl to failover

Prerequisite

- Before initiating a failover, verify that the PEER READY of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub** cluster is **True**.

```
$ oc get drpc -n busybox-sample -o wide
```

Example output:

```
NAME                AGE   PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE  PROGRESSION  START TIME      DURATION
PEER READY
busybox-placement-1-drpc 6m59s  ocp4bos1                Deployed
Completed <timestamp>    <duration>  True
```

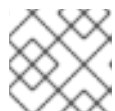


NOTE

If PEER READY is not true, see disaster recovery related known issues as documented in [Release Notes](#) for possible workarounds.

Procedure

1. On the Hub cluster, navigate to **Installed Operators** and then click **Openshift DR Hub Operator**.
2. Click **DRPlacementControl** tab.



NOTE

Make sure to be in the busybox-sample namespace.

3. Click DRPC **busybox-placement-1-drpc** and then the YAML view.
4. Add the **action** and **failoverCluster** details as shown in the screenshot below.

DRPlacementControl add action Failover

Project: busybox-sample ▾

Installed Operators > odr-hub-operator.v4.11.0 > DRPlacementControl details

DRPC busybox-placement-1-drpc Deployed

Details YAML Resources Events

```

1  apiVersion: ramendr.openshift.io/v1alpha1
2  kind: DRPlacementControl
3  metadata:
4    resourceVersion: '2225728'
5    name: busybox-placement-1-drpc
6    uid: a20e8df6-5fa5-4a5f-b1bc-2e9d350f4636
7    creationTimestamp: '2022-08-23T10:17:17Z'
8    generation: 2
9  > managedFields: ...
75 namespace: busybox-sample
76 finalizers:
77   - drpc.ramendr.openshift.io/finalizer
78 labels:
79   app: busybox
80   cluster.open-cluster-management.io/backup: resource
81 spec:
82   action: Failover
83   drPolicyRef:
84     name: ocp4bos1-ocp4bos2-5m
85   failoverCluster: ocp4bos2
86   placementRef:
87     kind: PlacementRule
88     name: busybox-placement-1
89     namespace: busybox-sample
90   preferredCluster: ocp4bos1
91   pvcSelector: {}

```

The **failoverCluster** should be the RHACM cluster name for the **Secondary managed cluster**.

5. Click **Save**.

- Verify that the **CURRENTSTATE** of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub cluster** is **FailedOver**.

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

Example output:

```
NAME                AGE   PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE
busybox-placement-1-drpc  6m59s  ocp4bos1          ocp4bos2      Failover
FailedOver
```

- Verify that for the failover cluster **ocp4bos2** as specified in the YAML file, the application **busybox** is now running in the **Secondary managed cluster**.

```
$ 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
```

- Verify if **busybox** is running in the **Primary 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.
```

- Verify if the DNS routes for the application are configured correctly. If the external routes are not configured, you can use either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service to reconfigure the routes.



IMPORTANT

Be aware of known DR issues as documented in [Known Issues](#) section of Release Notes.

3.11. 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.

Relocation is performed once the failed cluster is available and the application resources are cleaned up on the failed cluster.

In this case the action is **Relocate** back to the **preferredCluster**.

3.11.1. Modify DRPlacementControl to Relocate

Prerequisite

- Before initiating a relocate, verify that the PEER READY of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub cluster** is **True**.

```
$ oc get drpc -n busybox-sample -o wide
```

Example output:

```
NAME                AGE    PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE  PROGRESSION  START TIME    DURATION
PEER READY
busybox-placement-1-drpc  6m59s  ocp4bos1         ocp4bos2     Failover
FailedOver    Completed    <timestamp>    <duration>  True
```



NOTE

If PEER READY is not true, see disaster recovery related known issues as documented in [Release Notes](#) for possible workarounds.

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-placement-1-drpc** and then the YAML view.
4. Modify **action** to **Relocate**.

DRPlacementControl modify action to Relocate

Project: busybox-sample ▾

Installed Operators > odr-hub-operator.v4.11.0 > DRPlacementControl details

DRPC busybox-placement-1-drpc FailedOver

Details YAML Resources Events

```

1  apiVersion: ramendr.openshift.io/v1alpha1
2  kind: DRPlacementControl
3  metadata:
4    resourceVersion: '2240240'
5    name: busybox-placement-1-drpc
6    uid: a20e8df6-5fa5-4a5f-b1bc-2e9d350f4636
7    creationTimestamp: '2022-08-23T10:17:17Z'
8    generation: 3
9  > managedFields: ...
77 namespace: busybox-sample
78 finalizers:
79   - drpc.ramendr.openshift.io/finalizer
80 labels:
81   app: busybox
82   cluster.open-cluster-management.io/backup: resource
83 spec:
84   action: Relocate
85   drPolicyRef:
86     name: ocp4bos1-ocp4bos2-5m
87   failoverCluster: ocp4bos2
88   placementRef:
89     kind: PlacementRule
90     name: busybox-placement-1
91     namespace: busybox-sample
92   preferredCluster: ocp4bos1
93   pvcSelector: {}

```

- Click **Save**.
- Verify that the **CURRENTSTATE** of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub cluster** is **Relocated**.

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

Example output:

```

NAME          AGE   PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE
busybox-placement-1-drpc 6m59s ocp4bos1          ocp4bos2          Relocate
Relocated

```

- Verify if the application **busybox** is now running in the **Primary managed cluster**. The relocate is to the **preferredCluster ocp4bos1** 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

```

- 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.
```

- Verify if the DNS routes for the application are configured correctly. If the external routes are not configured, you can use either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service to reconfigure the routes.



IMPORTANT

Be aware of known DR issues as documented in [Known Issues](#) section of Release Notes.

CHAPTER 4. METRO-DR SOLUTION FOR OPENSIFT DATA FOUNDATION

4.1. COMPONENTS OF METRO-DR SOLUTION

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

Red Hat Advanced Cluster Management for Kubernetes

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

RHACM is split into two parts:

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

For more information about this product, see [RHACM documentation](#) and the [RHACM “Manage Applications” documentation](#).

Red Hat Ceph Storage

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

For more product information, see [Red Hat Ceph Storage](#).

OpenShift Data Foundation

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

OpenShift 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 and its state relationship across OpenShift clusters
- Failing over an application and its state to a peer cluster
- Relocate an application and its 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 orchestrates configuration and peering of OpenShift Data Foundation clusters for Metro and Regional DR relationships.

- **OpenShift DR Hub Operator:** Automatically installed as part of ODF Multicluster Orchestrator installation on the hub cluster to orchestrate failover or relocation of DR enabled applications.
- **OpenShift DR Cluster Operator:** Automatically installed on each managed cluster that is part of a Metro and Regional DR relationship to manage the lifecycle of all PVCs of an application.

4.2. METRO-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Metro-DR capabilities using latest version of Red Hat OpenShift Data Foundation, Red Hat Ceph Storage (RHCS) and Red Hat Advanced Cluster Management (RHACM) 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 requirements across the three: Hub, Primary and Secondary OpenShift Container Platform clusters that are part of the DR solution are met. See [Requirements for enabling Metro-DR](#).
2. Ensure you meet the requirements for deploying Red Hat Ceph Storage stretch cluster with arbiter. See [Requirements for deploying Red Hat Ceph Storage](#).
3. Deploy and configure Red Hat Ceph Storage stretch mode. For instructions on enabling Ceph cluster on two different data centers using stretched mode functionality, see [Deploying Red Hat Ceph Storage](#).
4. Install OpenShift Data Foundation operator and create a storage system on Primary and Secondary managed clusters. See [Creating OpenShift Data Foundation cluster on managed clusters](#).
5. Install the ODF Multicluster Orchestrator on the Hub cluster. See [Installing ODF Multicluster Orchestrator on Hub cluster](#).
6. Configure SSL access between the Hub, Primary and Secondary clusters. See [Configuring SSL access across clusters](#).
7. Enable the Web Console. [Enabling Multicluster Web Console](#)
8. Create a DRPolicy resource for use with applications requiring DR protection across the Primary and Secondary clusters. See [Creating Disaster Recovery Policy on Hub cluster](#).



NOTE

There can be more than a single policy.

For testing your disaster recovery solution:

- Create a sample application using RHACM console. See [Creating sample application](#).
- Test failover and relocate operations using the sample application between managed clusters. See [application failover](#) and [relocating an application](#).

4.3. REQUIREMENTS FOR ENABLING METRO-DR

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

- You must have three OpenShift clusters that have network reachability between them:
 - **Hub cluster** where Red Hat Advanced Cluster Management for Kubernetes (RHACM operator) is installed.
 - **Primary managed cluster** where OpenShift Data Foundation is installed.
 - **Secondary managed cluster** where OpenShift Data Foundation is 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 RHACM console:

```
$ oc get route multcloud-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
```

- Open your output link in a browser to login with OpenShift credentials. You should now see your local-cluster imported.



IMPORTANT

It is the user's responsibility to ensure that application traffic routing and redirection are configured appropriately. Configuration and updates to the application traffic routes are currently not supported.

- Ensure that you either import or create the **Primary managed cluster** and the **Secondary managed cluster** using the RHACM console. Choose the appropriate options for your environment. After the managed clusters are successfully created or imported, you can see the list of clusters that were imported or created on the console. For instructions, see [Creating a cluster](#) and [Importing a target managed cluster to the hub cluster](#) .



WARNING

There are distance limitations between the locations where the OpenShift Container Platform managed clusters reside as well as the RHCS nodes. The network latency between the sites must be below 10 milliseconds round-trip time (RTT).

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

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

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



NOTE

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



IMPORTANT

Erasure coded pools cannot be used with stretch mode.

4.4.1. Hardware requirements

For information on minimum hardware requirements for deploying Red Hat Ceph Storage, see [Minimum hardware recommendations for containerized Ceph](#).

Table 4.1. Physical server locations and Ceph component layout for Red Hat Ceph Storage cluster deployment:

Node name	Datacenter	Ceph components
ceph1	DC1	OSD+MON+MGR
ceph2	DC1	OSD+MON
ceph3	DC1	OSD+MDS+RGW
ceph4	DC2	OSD+MON+MGR
ceph5	DC2	OSD+MON
ceph6	DC2	OSD+MDS+RGW
ceph7	DC3	MON

4.4.2. Software requirements

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

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

4.4.3. Network configuration requirements

The recommended Red Hat Ceph Storage configuration is as follows:

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



NOTE

You can use different subnets for each of the datacenters.

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

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

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

For more information on the required network environment, see [Ceph network configuration](#).

4.5. DEPLOYING RED HAT CEPH STORAGE

4.5.1. Node pre-deployment steps

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

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

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

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

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

```
subscription-manager repos --disable="*" --enable="rhel-8-for-x86_64-baseos-rpms" --
enable="rhel-8-for-x86_64-appstream-rpms"
```

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

```
dnf update -y
reboot
```

4. Select a node from the cluster to be your bootstrap node. **ceph1** is our bootstrap node in this example going forward.

Only on the bootstrap node **ceph1**, enable the **ansible-2.9-for-rhel-8-x86_64-rpms** and **rhceph-5-tools-for-rhel-8-x86_64-rpms** repositories:

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

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

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

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

```
$ hostname
```

Example output:

```
ceph1
```

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

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

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

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

```
$ hostname -f
```

Example output:

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

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

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

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

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

**IMPORTANT**

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

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

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

- c. Build the ansible inventory

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

**NOTE**

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

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

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

Example output:

```
ceph6 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
  "changed": false,
  "ping": "pong"
}
ceph4 | SUCCESS => {
  "ansible_facts": {
    "discovered_interpreter_python": "/usr/libexec/platform-python"
  },
```

```

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

```

- e. Run the following ansible playbook.

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

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

4.5.2. Cluster bootstrapping and service deployment with Cephadm

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

In this guide we are going to bootstrap the cluster and deploy all the needed Red Hat Ceph Storage services in one step using a cluster specification yaml file.

If you find issues during the deployment, it may be easier to troubleshoot the errors by dividing the deployment into two steps:

1. Bootstrap
2. Service deployment



NOTE

For additional information on the bootstrapping process, see [Bootstrapping a new storage cluster](#).

Procedure

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

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

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

```
cat <<EOF > /root/cluster-spec.yaml
service_type: host
addr: 10.0.40.78 ## <XXX.XXX.XXX.XXX>
hostname: ceph1 ## <ceph-hostname-1>
location:
  root: default
  datacenter: DC1
labels:
  - osd
  - mon
  - mgr
---
service_type: host
addr: 10.0.40.35
hostname: ceph2
location:
  datacenter: DC1
labels:
  - osd
  - mon
---
service_type: host
addr: 10.0.40.24
hostname: ceph3
location:
```



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

```

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

```

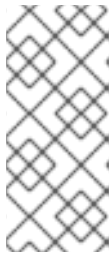
- Retrieve the IP for the NIC with the Red Hat Ceph Storage public network configured from the bootstrap node. After substituting **10.0.40.0** with the subnet that you have defined in your ceph public network, execute the following command.

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

Example output:

```
10.0.40.78
```

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

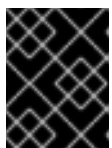


NOTE

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

If you are using non default/id_rsa ssh key names, then use **--ssh-private-key** and **--ssh-public-key** options with **cephadm** command.

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



IMPORTANT

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

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

You can access the Ceph CLI with:

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

Please consider enabling telemetry to help improve Ceph:

```
ceph telemetry on
```

For more information see:

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

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

```
$ ceph -s
```

Example output:

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

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

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



NOTE

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

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

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

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

```
$ ceph orch host ls
```

Example output:

```
HOST  ADDR      LABELS STATUS
ceph1 10.0.40.78 _admin osd mon mgr
ceph2 10.0.40.35 osd mon
```

```
ceph3 10.0.40.24  osd mds rgw
ceph4 10.0.40.185 osd mon mgr
ceph5 10.0.40.88  osd mon
ceph6 10.0.40.66  osd mds rgw
ceph7 10.0.40.221 mon
```



NOTE

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

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

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

Example output:

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

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

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

Example output:

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

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

```
$ ceph osd tree
```

Example output:

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

```

0  ssd 0.14650      osd.0   up  1.00000  1.00000
-9    0.14650      host ceph5
1  ssd 0.14650      osd.1   up  1.00000  1.00000
-7    0.14650      host ceph6
5  ssd 0.14650      osd.5   up  1.00000  1.00000

```

10. Create and enable a new RDB block pool.

```

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

```



NOTE

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

11. Verify that the RBD pool has been created.

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

Example output:

```
3 rbdpool
```

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

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

Example output:

```

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

```

13. Create the CephFS volume.

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



NOTE

The **ceph fs volume create** command also creates the needed data and meta CephFS pools. For more information, see [Configuring and Mounting Ceph File Systems](#).

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

■

```
$ ceph fs status
```

Example output:

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

15. Verify that RGW services are active.

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

Example output:

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

4.5.3. Configuring Red Hat Ceph Storage stretch mode

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

Procedure

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

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

Example output:

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

2. Change the monitor election to connectivity.

```
ceph mon set election_strategy connectivity
```

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

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

Example output:

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

To know more about the different election strategies, see [Configuring monitor election strategy](#).

- Set the location for all our Ceph monitors:

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

- Verify that each monitor has its appropriate location.

```
$ ceph mon dump
```

Example output:

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

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

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

To know more about CRUSH ruleset, see [Ceph CRUSH ruleset](#).

- Get the compiled CRUSH map from the cluster:

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

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

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

- Add the following rule to the CRUSH map at the end of the file **/etc/ceph/crushmap.txt**.

-

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

```
rule stretch_rule {
    id 1
    type replicated
    min_size 1
    max_size 10
    step take DC1
    step chooseleaf firstn 2 type host
    step emit
    step take DC2
    step chooseleaf firstn 2 type host
    step emit
}

# end crush map
```



NOTE

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

The CRUSH rule declared contains the following information:

- **Rule name:**
 - Description: A unique whole name for identifying the rule.
 - Value: **stretch_rule**
- **id:**
 - Description: A unique whole number for identifying the rule.
 - Value: **1**
- **type:**
 - Description: Describes a rule for either a storage drive replicated or erasure-coded.
 - Value: **replicated**
- **min_size:**
 - Description: If a pool makes fewer replicas than this number, CRUSH will not select this rule.
 - Value: **1**
- **max_size:**
 - Description: If a pool makes more replicas than this number, CRUSH will not select this rule.
 - Value: **10**

- **step take DC1**
 - Description: Takes a bucket name (DC1), and begins iterating down the tree.
- **step chooseleaf firstn 2 type host**
 - Description: Selects the number of buckets of the given type, in this case is two different hosts located in DC1.
- **step emit**
 - Description: Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to pick from different trees in the same rule.
- **step take DC2**
 - Description: Takes a bucket name (DC2), and begins iterating down the tree.
- **step chooseleaf firstn 2 type host**
 - Description: Selects the number of buckets of the given type, in this case, is two different hosts located in DC2.
- **step emit**
 - Description: Outputs the current value and empties the stack. Typically used at the end of a rule, but may also be used to pick from different trees in the same rule.

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

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

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

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

Example output:

```
17
```



NOTE

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

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

```
ceph osd crush rule ls
```

Example output:

```
replicated_rule
stretch_rule
```

13. Enable the stretch cluster mode.

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

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

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

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

Example output:

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

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

4.6. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER ON MANAGED CLUSTERS

In order to configure storage replication between the two OpenShift Container Platform clusters, create an OpenShift Data Foundation storage system after you install the OpenShift Data Foundation operator.

Prerequisites

- Ensure that you have met the hardware requirements for OpenShift Data Foundation external deployments. For a detailed description of the hardware requirements, see [External mode requirements](#).



NOTE

Refer to OpenShift Data Foundation deployment guides and instructions that are specific to your infrastructure (AWS, VMware, BM, Azure, etc.).

Procedure

1. Install and configure the latest **OpenShift Data Foundation** cluster on each of the managed clusters.
2. After installing the operator, create a StorageSystem using the option **Full deployment** type and **Connect with external storage platform** where your **Backing storage type** is **Red Hat Ceph Storage**.
For detailed instructions, refer to [Deploying OpenShift Data Foundation in external mode](#).

- At a minimum, you must use the following three flags with the **ceph-external-cluster-details-exporter.py** script:

--rbd-data-pool-name

With the name of the RBD pool that was created during RHCS deployment for OpenShift Container Platform. For example, the pool can be called **rbdpool**.

--rgw-endpoint

Provide the endpoint in the format **<ip_address>:<port>**. It is the RGW IP of the RGW daemon running on the same site as the OpenShift Container Platform cluster that you are configuring.

--run-as-user

With a different client name for each site.

The following flags are **optional** if default values were used during the RHCS deployment:

--cephfs-filesystem-name

With the name of the CephFS filesystem we created during RHCS deployment for OpenShift Container Platform, the default filesystem name is **cephfs**.

--cephfs-data-pool-name

With the name of the CephFS data pool we created during RHCS deployment for OpenShift Container Platform, the default pool is called **cephfs.data**.

--cephfs-metadata-pool-name

With the name of the CephFS metadata pool we created during RHCS deployment for OpenShift Container Platform, the default pool is called **cephfs.meta**.

- Run the following command on the bootstrap node, ceph1, to get the IP for the RGW endpoints in datacenter1 and datacenter2:

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

Example output:

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

```
host ceph3
host ceph6
```

Example output:

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

- Execute the **ceph-external-cluster-details-exporter.py** with the parameters configured for our first ocp managed cluster cluster1.

```
python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbdpool --cephfs-filesystem-name cephfs --cephfs-data-pool-name cephfs.cephfs.data --cephfs-metadata-pool-name cephfs.cephfs.meta --rgw-endpoint 10.0.40.24:8080 --run-as-user
```

```
client.odf.cluster1 > ocp-cluster1.json
```

- Execute the `ceph-external-cluster-details-exporter.py` with the parameters configured for our first ocp managed cluster `cluster2`

```
python3 ceph-external-cluster-details-exporter.py --rbd-data-pool-name rbdpool --cephfs-filesystem-name cephfs --cephfs-data-pool-name cephfs.cephfs.data --cephfs-metadata-pool-name cephfs.cephfs.meta --rgw-endpoint 10.0.40.66:8080 --run-as-user client.odf.cluster2 > ocp-cluster2.json
```

- Save the two files generated in the bootstrap cluster (ceph1) **ocp-cluster1.json** and **ocp-cluster2.json** to your local machine.
 - Use the contents of file **ocp-cluster1.json** on the OCP console on **cluster1** where external ODF is being deployed.
 - Use the contents of file **ocp-cluster2.json** on the OCP console on **cluster2** where external ODF is being deployed.
3. Review the settings and then select **Create StorageSystem**.
 4. Validate the successful deployment of OpenShift Data Foundation on each managed cluster with the following command:

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

For the Multicloud Gateway (MCG):

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

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



NOTE

In the OpenShift Web Console, navigate to **Installed Operators** → **OpenShift Data Foundation** → **Storage System** → **ocs-storagecluster-storagesystem** → **Resources** and verify that **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.

4.7. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multiclustor Orchestrator is a controller that is installed from OpenShift Container Platform's OperatorHub on the Hub cluster.

Procedure

1. On the **Hub cluster**, navigate to **OperatorHub** and use the keyword filter to search for **ODF Multiclustor Orchestrator**.
2. Click **ODF Multiclustor Orchestrator** tile.
3. Keep all default settings and click **Install**.

Ensure that the operator resources are installed in **openshift-operators** project and available to all namespaces.



NOTE

The **ODF Multicluster Orchestrator** also installs the **OpenShift DR Hub Operator** on the RHACM hub cluster as a dependency.

4. Verify that the operator **Pods** are in a **Running** state. The **OpenShift DR Hub operator** is also installed at the same time in **openshift-operators** namespace.

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

Example output:

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

4.8. CONFIGURING SSL ACCESS ACROSS CLUSTERS

Configure network (SSL) access between the **primary and secondary clusters** so that metadata can be stored on the alternate cluster in a Multicloud Gateway (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** file to hold the remote cluster's certificate bundle with filename **cm-clusters.crt.yaml**.

**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** 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
```

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

4.9. ENABLING MULTICLUSTER WEB CONSOLE

This is a new capability that is required before creating a Data Policy or **DRPolicy**. It is only needed on the **Hub cluster** and **RHACM 2.5** must be installed.



IMPORTANT

Multicluster console is a Technology Preview feature only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Procedure

1. Navigate to **Administration** → **Cluster Settings** → **Configuration** → **FeatureGate**.
2. Edit the YAML template as follows:

```
[...]
spec:
  featureSet: TechPreviewNoUpgrade
```

3. Click **Save** to enable the multicluster console for all clusters in the RHACM console. Wait for the Nodes to become **Ready**.
4. Refresh the web console and verify that the managed cluster names are listed below **All Clusters**.



WARNING

Do not set this feature gate on production clusters. You will not be able to upgrade your cluster after applying the feature gate, and it cannot be undone.

4.10. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER

Openshift Disaster Recovery Policy (DRPolicy) resource specifies OpenShift Container Platform clusters participating in the disaster recovery solution and the desired replication interval. DRPolicy is a cluster scoped resource that users can apply to applications that require Disaster Recovery solution.

The **ODF MultiCluster Orchestrator** Operator facilitates the creation of each **DRPolicy** and the corresponding **DRClusters** through the **Multicluster Web console**.

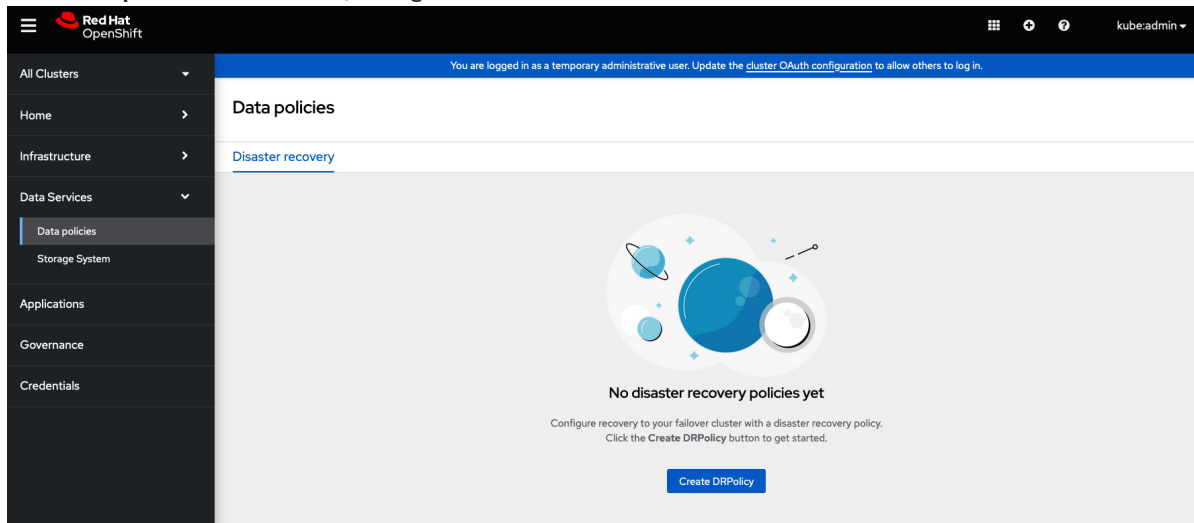
Prerequisites

- Ensure that there is a minimum set of two managed clusters.

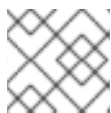
- Make sure to login to all the clusters from the **Multicluster Web console**.
 - Click on **All Clusters** to expand the list of managed clusters.
 - For each managed cluster listed below **All Clusters**, click on the `<cluster_name>` and then wait for a login screen to appear, where you can login using the credentials of the cluster that you have selected.

Procedure

1. On the **OpenShift console**, navigate to **All Clusters**.



2. Navigate to **Data Services** and click **Data policies**.
3. Click **Create DRPolicy**.
4. Enter **Policy name**. Ensure that each DRPolicy has a unique name (for example: **ocp4perf1-ocp4perf2**).
5. Select two clusters from the list of managed clusters to which this new policy will be associated with.
6. **Replication policy** is automatically set to **sync** based on the OpenShift clusters selected.
7. Click **Create**.
8. Verify that the **DRPolicy** is created successfully. Run this command on the **Hub cluster** for each of the DRPolicy resources created.



NOTE

Replace `<drpolicy_name>` with your unique name.

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

Example output:

```
Succeeded
```


**NOTE**

When a DRPolicy is created, along with it, two DRCluster resources are also created. It could take up to 10 minutes for all three resources to be validated and for the status to show as **Succeeded**.

9. Verify the object bucket access from the **Hub cluster** to both the **Primary managed cluster** and the **Secondary managed cluster**.

- a. Get the names of the **DRClusters** on the Hub cluster.

```
$ oc get drclusters
```

Example output:

```
NAME      AGE
ocp4perf1 4m42s
ocp4perf2 4m42s
```

- b. Check S3 access to each bucket created on each managed cluster using this **DRCluster** validation command.

**NOTE**

Replace <drcluster_name> with your unique name.

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

Example output:

```
Succeeded
```

**NOTE**

Make sure to run command for both **DRClusters** on the **Hub cluster**.

10. Verify that the **OpenShift DR Cluster operator** installation was successful on the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

```
NAME                                DISPLAY                                VERSION
REPLACES PHASE
clusterserviceversion.operators.coreos.com/odr-cluster-operator.v4.11.0  Openshift DR
Cluster Operator 4.11.0                Succeeded

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

You can also verify that **OpenShift DR Cluster Operator** is installed successfully on the **OperatorHub** of each managed clusters.

4.11. CONFIGURE DRCLUSTERS FOR FENCING AUTOMATION

This configuration is required for enabling fencing prior to application failover. In order to prevent writes to the persistent volume from the cluster which is hit by a disaster, OpenShift DR instructs Red Hat Ceph Storage (RHCS) to fence the nodes of the cluster from the RHCS external storage. This section guides you on how to add the IPs or the IP Ranges for the nodes of the DRCluster.

4.11.1. Add node IP addresses to DRClusters

1. Find the **IP addresses** for all of the OpenShift nodes in the managed clusters by running this command in the **Primary managed cluster** and the **Secondary managed cluster**.

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

Example output:

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

Once you have the **IP addresses** then the **DRCluster** resources can be modified for each managed cluster.

2. Find the **DRCluster** names on the Hub Cluster.

```
$ oc get drcluster
```

Example output:

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

3. Edit each **DRCluster** to add your unique IP addresses after replacing **<drcluster_name>** with your unique name.

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

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

```

- <IP_Address1>/32
- <IP_Address2>/32
- <IP_Address3>/32
- <IP_Address4>/32
- <IP_Address5>/32
- <IP_Address6>/32
[...]

```

Example output:

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



NOTE

There could be more than six IP addresses.

Modify this **DRCluster** configuration also for **IP addresses** on the **Secondary managed clusters** in the peer DRCluster resource (e.g., ocp4perf2).

4.11.2. Add fencing annotations to DRClusters

Add the following annotations to all the DRCluster resources. These annotations include details needed for the **NetworkFence** resource created later in these instructions (prior to testing application failover).



NOTE

Replace `<drcluster_name>` with your unique name.

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

```

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

```

Example output:

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

Make sure to add these annotations for both **DRCluster** resources (for example: **ocp4perf1** and **ocp4perf2**).

4.12. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION

OpenShift Data Foundation disaster recovery (DR) solution supports disaster recovery for applications that are managed by RHACM. See [Managing Applications](#) for more details.

This solution orchestrates RHACM application placement, using the [PlacementRule](#), when an application is moved between clusters in a DRPolicy for failover or relocation requirements.

The following sections detail how to apply a **DRPolicy** to an application and how to manage the applications placement life-cycle during and after cluster unavailability.



NOTE

OpenShift Data Foundation DR solution does not support ApplicationSet, which is required for applications that are deployed via ArgoCD.

4.12.1. Creating a sample application

In order to test **failover** from the **Primary managed cluster** to the **Secondary managed cluster** and **relocate**, we need a simple application.

Prerequisites

- When creating an application for general consumption, ensure that:
 - the application is deployed to ONLY one cluster.
 - the application is deployed prior to applying the DRPolicy to the application.
- Use the sample application called **busybox** as an example.
- Ensure all external routes of the application are configured using either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service for traffic redirection when the application fails over or is relocated.

Procedure

1. 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
```

2. Navigate to **Applications** and click **Create application**.
3. Select type as **Subscription**.
4. Enter your application **Name** (for example, **busybox**) and **Namespace** (for example, **busybox-sample**).
5. In the Repository location for resources section, select **Repository type Git**.
6. 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/red-hat-storage/ocm-ramen-samples/tree/release-4.11> where the **Branch** is **main** and **Path** is **busybox-odr-metro**.

7. Scroll down in the form until you see **Deploy application resources only on clusters matching specified labels** and then add a label with its value set to the **Primary managed cluster** name in **RHACM** cluster list view.

Deploy application resources only on clusters matching specified labels

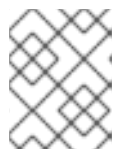
Cluster labels

Enter one or more matching labels to select the clusters to deploy to

Label *	Value *
name	ocp4perf1

+ Add another label

8. Click **Create** which is at the top right hand corner.
On the follow-on screen go to the **Topology** tab. 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.

9. Validating the sample application deployment.
Now that the **busybox** application has been deployed to your preferred Cluster, the deployment can be validated.

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
5Gi   RWO          ocs-storagecluster-ceph-rbd 6m
```

4.12.2. Apply DRPolicy to sample application

1. On the Hub cluster go back to the Multicluster Web console, navigate to **All Clusters**.

2. Login to all the clusters listed under **All Clusters**.
3. Navigate to **Data Services** and then click **Data policies**.
4. Click the Actions menu at the end of DRPolicy to view the list of available actions.

The screenshot shows the Red Hat OpenShift console interface. The left sidebar contains navigation options: All Clusters, Home, Infrastructure, Data Services, Applications, Governance, and Credentials. The main content area is titled 'Data policies' and includes a 'Disaster recovery' sub-section. A search bar is present above a table. The table has the following data:

Name	Status	Clusters	Replication policy	Connected applications
ocp4bos1-ocp4bos2-5m	Validated	ocp4bos1 ocp4bos2	async	0 Applications

An actions menu is open for the first row, listing: Apply DRPolicy, Edit labels, Edit annotations, Edit DR Policy, and Delete DRPolicy. The 'Apply DRPolicy' option is circled in red.

5. Click **Apply DRPolicy**.
6. When the **Apply DRPolicy** modal is displayed, select **busybox** application and enter **PVC label** as **apname=busybox**.



NOTE

When multiple placements rules under the same application or more than one application are selected, all PVCs within the application's namespace will be protected by default.

7. Click **Apply**.
8. Verify that a **DRPlacementControl** or **DRPC** was created in the **busybox-sample** namespace on the **Hub cluster** and that its **CURRENTSTATE** shows as **Deployed**. This resource is used for both failover and relocate actions for this application.

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

Example output:

```
NAME                AGE   PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE
busybox-placement-1-drpc  6m59s  ocp4perf1                Deployed
```

4.12.3. 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 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 the **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 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.

4.13. APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

A failover is performed when a managed cluster becomes unavailable, due to any reason.

This section provides instructions on how to failover the busybox sample application. The failover method is application based. Each application that is to be protected in this manner must have a corresponding **DRPlacementControl** resource in the application namespace.

Prerequisites

- Ensure that the primary cluster where the applications are running is fenced while the target cluster is unfenced.

4.13.1. Enable fencing

In order to failover the OpenShift cluster where the application is currently running all applications must be fenced from communicating with the external OpenShift Data Foundation external storage cluster. This is required to prevent simultaneous writes to the same persistent volume from both managed clusters.

The OpenShift cluster to **Fence** is the one where the applications are currently running.

Procedure

1. Edit the **DRCluster** resource for this cluster on the **Hub cluster**.

CAUTION

Once the managed cluster is fenced, **all** communication from applications to the OpenShift Data Foundation external storage cluster will fail and some **Pods** will be in an unhealthy state (for example: **CreateContainerError**, **CrashLoopBackOff**) on the cluster that is now fenced.



NOTE

Replace `<drcluster_name>` with your unique name.

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

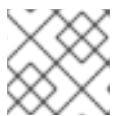
```
apiVersion: ramendr.openshift.io/v1alpha1
kind: DRCluster
metadata:
[...]
```

```
spec:
  ## Add this line
  clusterFence: Fenced
  cidrs:
  [...]
  [...]
```

Example output:

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

2. Verify the fencing status in the **Hub cluster** for the **Primary managed cluster**.



NOTE

Replace `<drcluster_name>` with your unique name.

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

Example output:

```
Fenced
```

4.13.2. Modify DRPlacementControl to failover

Prerequisite

- Before initiating a failover, verify that the PEER READY of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub cluster** is **True**.

```
$ oc get drpc -n busybox-sample -o wide
```


Example output:

```

NAME          AGE  PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE  PROGRESSION  START TIME    DURATION
PEER READY
busybox-placement-1-drpc 6m59s  ocp4perf1          Deployed
Completed <timestamp>    <duration>  True

```



NOTE

If PEER READY is not true, see disaster recovery related known issues as documented in [Release Notes](#) for possible workarounds.

Procedure

1. On the Hub cluster, navigate to **Installed Operators** and then click **Openshift DR Hub Operator**.
2. Click **DRPlacementControl** tab.



NOTE

Make sure to be in the busybox-sample namespace.

3. Click DRPC **busybox-placement-1-drpc** and then the YAML view.
4. Add the **action** and **failoverCluster** details as shown in the screenshot below.

DRPlacementControl add action Failover

Project: busybox-sample ▾

Installed Operators > odr-hub-operator.v4.11.0 > DRPlacementControl details

DRPC busybox-placement-1-drpc Deployed

Details YAML Resources Events

```

1  apiVersion: ramendr.openshift.io/v1alpha1
2  kind: DRPlacementControl
3  metadata:
4    resourceVersion: '19759627'
5    name: busybox-placement-1-drpc
6    uid: c193e6c4-709f-414f-9c5e-c9d63ff7641c
7    creationTimestamp: '2022-07-14T23:11:28Z'
8    generation: 2
9  > managedFields:--
73 namespace: busybox-sample
74 finalizers:
75   - drpc.ramendr.openshift.io/finalizer
76 labels:
77   app: busybox
78   cluster.open-cluster-management.io/backup: resource
79 spec:
80   action: Failover
81   failoverCluster: ocp4perf2
82   drPolicyRef:
83     name: ocp4perf1-ocp4perf2
84   placementRef:
85     kind: PlacementRule
86     name: busybox-placement-1
87     namespace: busybox-sample
88   preferredCluster: ocp4perf1
89   pvcSelector: {}

```

Save Reload Cancel

The **failoverCluster** should be the RHACM cluster name for the **Secondary** managed cluster.

- Click **Save**.
- Verify that the **CURRENTSTATE** of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub** cluster is **FailedOver**.

```
$ oc get drpc -n busybox-sample (update examples as required)
```

Example output:

```
NAME                AGE   PREFERREDCLUSTER  FAILOVERCLUSTER
DESIREDSTATE  CURRENTSTATE
busybox-placement-1-drpc  6m59s  ocp4perf1         ocp4perf2      Failover      FailedOver
```

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

```
$ 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
```

- Verify if **busybox** is running in the **Primary 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.
```

- Verify if the DNS routes for the application are configured correctly. If the external routes are not configured, you can use either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service to reconfigure the routes.



IMPORTANT

Be aware of known DR issues as documented in [Known Issues](#) section of Release Notes.

4.14. 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.

Relocation is performed once the failed cluster is available and the application resources are cleaned up on the failed cluster.

In this case the action is **Relocate** back to the **preferredCluster**.

Prerequisite

- Ensure both the primary and target clusters are unfenced.

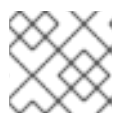
4.14.1. Disable fencing

Before a **failback** or **Relocate** action can be successful, the DRCluster for the Primary managed cluster must be **unfenced**.

The OpenShift cluster to be **Unfenced** is the one where applications are not currently running and the cluster that was **Fenced** earlier.

Procedure

1. On the Hub cluster, edit the **DRCluster resource** for this cluster.



NOTE

Replace `<drcluster_name>` with your unique name.

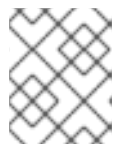
```
$ oc edit drcluster <drcluster_name>
```

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

Example output:

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

2. Gracefully reboot OpenShift Container Platform nodes that were **Fenced**. A reboot is required to resume the I/O operations after unfencing to avoid any further recovery orchestration failures. Reboot all worker nodes of the cluster by following the steps in the procedure, [Rebooting a node gracefully](#).



NOTE

Make sure that all the nodes are initially cordoned and drained before you reboot and perform uncordon operations on the nodes.

3. After all OpenShift nodes are rebooted and are in a **Ready** status, verify that all Pods are in a healthy state by running this command on the Primary managed cluster (or whatever cluster has been Unfenced).

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

Example output:

NAMESPACE	STATUS	RESTARTS	AGE	NAME	READY
-----------	--------	----------	-----	------	-------

The output for this query should be zero Pods before proceeding to the next step.



IMPORTANT

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

Alternatively, you can use the OpenShift Web Console dashboards and Overview tab to assess the health of applications and the external ODF storage cluster. The detailed OpenShift Data Foundation dashboard is found by navigating to **Storage → Data Foundation**

4. Verify that the **Unfenced** cluster is in a healthy state. Validate the fencing status in the Hub cluster for the Primary managed cluster.



NOTE

Replace `<drcluster_name>` with your unique name.

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

Example output:

```
Unfenced
```

4.14.2. Modify DRPlacementControl to Relocate

Prerequisite

- Ensure that both managed clusters are up and running.
- Before initiating a relocate, verify that the PEER READY of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub cluster** is **True**.

```
$ oc get drpc -n busybox-sample -o wide
```

Example output:

NAME	AGE	PREFERREDCLUSTER	FAILOVERCLUSTER	DESIREDSTATE	CURRENTSTATE	PROGRESSION	START TIME	DURATION
busybox-placement-1-drpc	6m59s	ocp4perf1	ocp4perf2	Completed	<timestamp>	<duration>	Failover	FailedOver
						True		

**NOTE**

If PEER READY is not true, see disaster recovery related known issues as documented in [Release Notes](#) for possible workarounds.

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-placement-1-drpc** and then the YAML view.
4. Modify **action** to **Relocate**.

DRPlacementControl modify action to Relocate

Project: busybox-sample ▾

Installed Operators > odr-hub-operator.v4.11.0 > DRPlacementControl details

DRPC busybox-placement-1-drpc FailedOver

Details YAML Resources Events

```

1  apiVersion: ramendr.openshift.io/v1alpha1
2  kind: DRPlacementControl
3  metadata:
4    resourceVersion: '19880202'
5    name: busybox-placement-1-drpc
6    uid: c193e6c4-709f-414f-9c5e-c9d63ff7641c
7    creationTimestamp: '2022-07-14T23:11:28Z'
8    generation: 3
9  > managedFields: --
75 namespace: busybox-sample
76 finalizers:
77   - drpc.ramendr.openshift.io/finalizer
78 labels:
79   app: busybox
80   cluster.open-cluster-management.io/backup: resource
81 spec:
82   action: Relocate
83   drPolicyRef:
84     name: ocp4perf1-ocp4perf2
85   failoverCluster: ocp4perf2
86   placementRef:
87     kind: PlacementRule
88     name: busybox-placement-1
89     namespace: busybox-sample
90   preferredCluster: ocp4perf1
91   pvcSelector: {}

```

Save **Reload** **Cancel**

- Click **Save**.
- Verify that the **CURRENTSTATE** of **DRPlacementControl** or **DRPC** in the **busybox-sample** namespace on the **Hub** cluster is **Relocated**.

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

Example output:

NAME	AGE	PREFERREDCLUSTER	FAILOVERCLUSTER	DESIREDSTATE	CURRENTSTATE
busybox-placement-1-drpc	6m59s	ocp4perf1	ocp4perf2	Relocate	Relocated

- Verify if the application **busybox** is now running in the **Primary managed cluster**. The relocate 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	

- 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.

- Verify if the DNS routes for the application are configured correctly. If the external routes are not configured, you can use either Global Traffic Manager (GTM) or Global Server Load Balancing (GLSB) service to reconfigure the routes.



IMPORTANT

Be aware of known DR issues as documented in [Known Issues](#) section of Release Notes.

CHAPTER 5. TROUBLESHOOTING DISASTER RECOVERY

5.1. TROUBLESHOOTING REGIONAL-DR

5.1.1. RBD mirroring scheduling is getting stopped for some images

Problem

There are a few common causes for RBD mirroring scheduling getting stopped for some images. After marking the applications for mirroring, for some reason, if it is not replicated, use the toolbox pod and run the following command to see which image scheduling is stopped.

```
$ rbd snap ls <poolname/imagename> --all
```

Resolution

- Restart the manager daemon on the primary cluster
- Disable and immediately re-enable mirroring on the affected images on the primary cluster

BZ reference: [2067095 and 2121514]

5.1.2. Relocation failure

Problem

Relocation stalls forever when the relocation is initiated before the peer (target cluster) is in a clean state.

Resolution

1. Check the condition **Status** by running the following command:

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

2. Change **DRPC.Spec.Action** back to **Failover**, and wait until the **PeerReady** condition status is **TRUE**. Use this command to change the action:

```
$ oc patch drpc <drpc_name> --type json -p "[{'op': 'add', 'path': '/spec/failoverCluster', 'value': '<failoverCluster_name>'}]" -n <application_namespace>
```

```
$ oc patch drpc <drpc_name> --type json -p "[{'op': 'add', 'path': '/spec/action', 'value': 'Failover'}]" -n <application_namespace>
```

3. Failover verification

To verify if workload has failed over, run the following command to check the status of the available condition in the DRPC resource:

```
JSONPATH='{range @.status.conditions[*]}{@.type}={@.status};{end}' && oc get drpc busybox-drpc -n busybox-sample -o jsonpath="$JSONPATH" | grep "Available=True"
```

BZ reference: [2056871]

5.1.3. rbd-mirror daemon health is in warning state

Problem

There appears to be numerous cases where WARNING gets reported if mirror service `::get_mirror_service_status` calls **Ceph** monitor to get service status for **rbd-mirror**. Following a network disconnection, **rbd-mirror** daemon health is in the **warning** state while the connectivity between both the managed clusters is fine.

Resolution

Run the following command in the toolbox and look for **leader:false**

```
rbd mirror pool status --verbose ocs-storagecluster-cephblockpool | grep 'leader:'
```

If you see the following in the output:

leader: false	It indicates that there is a daemon startup issue and the most likely root cause could be due to problems reliably connecting to the secondary cluster. Workaround: Move the rbd-mirror pod to a different node by simply deleting the pod and verify that it has been rescheduled on another node.
leader: true or no output	Contact Red Hat Support

BZ reference: [\[2118627\]](#)

5.1.4. statefulset application stuck after failover

Problem

Application is in **terminating** state after failover or relocate.

Resolution

Delete the application's persistent volume claim (pvc) using this command:

```
$ oc delete pvc <namespace/name>
```

BZ reference: [\[2087782\]](#)

5.1.5. Application is not running after failover

Problem

After failing over an application, the mirrored RBDs can be attached but the filesystems that are still in use cannot be mounted.

Resolution

1. Scale down the RBD mirror daemon deployment to **0** until the application pods can recover from the above error.

```
$ oc scale deployment rook-ceph-rbd-mirror-a -n openshift-storage --replicas=0
```

2. Post recovery, scale the RBD mirror daemon deployment back to **1**.

```
$ oc scale deployment rook-ceph-rbd-mirror-a -n openshift-storage --replicas=1
```

BZ reference: [\[2007376\]](#)

5.2. TROUBLESHOOTING METRO-DR

5.2.1. A statefulset application stuck after failover

Problem

Application is in terminating state after failover or relocate.

Resolution

1. If the workload uses **StatefulSets**, then run the following command before failing back or relocating to another cluster:

```
$ oc get drpc -n <namespace> -o wide
```

- If PeerReady is TRUE then you can proceed with the failback or relocation.
- If PeerReady is FALSE then run the following command on the peer cluster:

```
$ oc get pvc -n <namespace>
```

For each bounded PVC for that namespace that belongs to the StatefulSet, run

```
$ oc delete pvc <pvcname> -n namespace
```

Once all PVCs are deleted, Volume Replication Group (VRG) transitions to secondary, and then gets deleted.

2. Run the following command again

```
$ oc get drpc -n <namespace> -o wide
```

After a few seconds to a few minutes, the PeerReady column changes to **TRUE**. Then you can proceed with the failback or relocation.

BZ reference: [\[2118270\]](#)

5.2.2. DR policies protect all applications in the same namespace

Problem

While only single application is selected to be used by a DR policy, all applications in the same namespace will be protected. This results in PVCs, that match the **DRPlacementControl spec.pvcSelector** across multiple workloads or if the selector is missing across all workloads,

replication management to potentially manage each PVC multiple times and cause data corruption or invalid operations based on individual **DRPlacementControl** actions.

Resolution

Label PVCs that belong to a workload uniquely, and use the selected label as the DRPlacementControl **spec.pvcSelector** to disambiguate which DRPlacementControl protects and manages which subset of PVCs within a namespace. It is not possible to specify the **spec.pvcSelector** field for the DRPlacementControl using the user interface, hence the DRPlacementControl for such applications must be deleted and created using the command line.

BZ reference: [\[2111163\]](#)

5.2.3. During failback of an application stuck in Relocating state

Problem

This issue might occur after performing failover and failback of an application (all nodes or cluster are up). When performing failback application stuck in the **Relocating** state with a message of **Waiting** for PV restore to complete.

Resolution

Use S3 client or equivalent to clean up the duplicate PV objects from the s3 store. Keep only the one that has a timestamp closer to the failover or relocate time.

BZ reference: [\[2120201\]](#)