



Red Hat OpenShift Data Foundation 4.15

Configuring OpenShift Data Foundation Disaster Recovery for OpenShift Workloads

The OpenShift Data Foundation Disaster Recovery capabilities for Metropolitan and Regional regions is now General Available which also includes Disaster Recovery with stretch cluster.

Red Hat OpenShift Data Foundation 4.15 Configuring OpenShift Data Foundation Disaster Recovery for OpenShift Workloads

The OpenShift Data Foundation Disaster Recovery capabilities for Metropolitan and Regional regions is now General Available which also includes Disaster Recovery with stretch cluster.

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 and stretch cluster to achieve a highly available storage infrastructure.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE	5
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION	6
CHAPTER 1. INTRODUCTION TO OPENSIFT DATA FOUNDATION DISASTER RECOVERY	7
CHAPTER 2. DISASTER RECOVERY SUBSCRIPTION REQUIREMENT	8
CHAPTER 3. METRO-DR SOLUTION FOR OPENSIFT DATA FOUNDATION	9
3.1. COMPONENTS OF METRO-DR SOLUTION	9
3.2. METRO-DR DEPLOYMENT WORKFLOW	10
3.3. REQUIREMENTS FOR ENABLING METRO-DR	11
3.4. REQUIREMENTS FOR DEPLOYING RED HAT CEPH STORAGE STRETCH CLUSTER WITH ARBITER	12
3.4.1. Hardware requirements	13
3.4.2. Software requirements	13
3.4.3. Network configuration requirements	13
3.5. DEPLOYING RED HAT CEPH STORAGE	14
3.5.1. Node pre-deployment steps	14
3.5.2. Cluster bootstrapping and service deployment with cephadm utility	17
3.5.3. Configuring Red Hat Ceph Storage stretch mode	24
3.6. INSTALLING OPENSIFT DATA FOUNDATION ON MANAGED CLUSTERS	28
3.7. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR	31
3.8. CONFIGURING SSL ACCESS ACROSS CLUSTERS	31
3.9. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER	33
3.10. CONFIGURE DRCLUSTERS FOR FENCING AUTOMATION	35
3.10.1. Add node IP addresses to DRClusters	35
3.10.2. Add fencing annotations to DRClusters	36
3.11. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION	37
3.11.1. Subscription-based applications	37
3.11.1.1. Creating a sample Subscription-based application	37
3.11.1.2. Apply Data policy to sample application	39
3.11.2. ApplicationSet-based applications	39
3.11.2.1. Creating ApplicationSet-based applications	39
3.11.2.2. Apply Data policy to sample ApplicationSet-based application	41
3.11.3. Deleting sample application	41
3.12. SUBSCRIPTION-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	42
3.13. APPLICATIONSET-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	44
3.14. RELOCATING SUBSCRIPTION-BASED APPLICATION BETWEEN MANAGED CLUSTERS	46
3.15. RELOCATING AN APPLICATIONSET-BASED APPLICATION BETWEEN MANAGED CLUSTERS	49
3.16. RECOVERING TO A REPLACEMENT CLUSTER WITH METRO-DR	51
3.17. HUB RECOVERY USING RED HAT ADVANCED CLUSTER MANAGEMENT [TECHNOLOGY PREVIEW]	56
3.17.1. Configuring passive hub cluster	56
3.17.2. Switching to passive hub cluster	57
CHAPTER 4. REGIONAL-DR SOLUTION FOR OPENSIFT DATA FOUNDATION	59
4.1. COMPONENTS OF REGIONAL-DR SOLUTION	59
4.2. REGIONAL-DR DEPLOYMENT WORKFLOW	60
4.3. REQUIREMENTS FOR ENABLING REGIONAL-DR	60
4.4. CREATING AN OPENSIFT DATA FOUNDATION CLUSTER ON MANAGED CLUSTERS	62
4.5. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR	64
4.6. CONFIGURING SSL ACCESS ACROSS CLUSTERS	65
4.7. CREATING DISASTER RECOVERY POLICY ON HUB CLUSTER	66

4.8. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION	69
4.8.1. Subscription-based applications	69
4.8.1.1. Creating a sample Subscription-based application	69
4.8.1.2. Apply Data policy to sample application	71
4.8.2. ApplicationSet-based applications	72
4.8.2.1. Creating ApplicationSet-based applications	72
4.8.2.2. Apply Data policy to sample ApplicationSet-based application	73
4.8.3. Deleting sample application	75
4.9. SUBSCRIPTION-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	76
4.10. APPLICATIONSET-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS	77
4.11. RELOCATING SUBSCRIPTION-BASED APPLICATION BETWEEN MANAGED CLUSTERS	79
4.12. RELOCATING AN APPLICATIONSET-BASED APPLICATION BETWEEN MANAGED CLUSTERS	80
4.13. VIEWING RECOVERY POINT OBJECTIVE VALUES FOR DISASTER RECOVERY ENABLED APPLICATIONS	81
4.14. HUB RECOVERY USING RED HAT ADVANCED CLUSTER MANAGEMENT [TECHNOLOGY PREVIEW]	82
4.14.1. Configuring passive hub cluster	82
4.14.2. Switching to passive hub cluster	82
CHAPTER 5. DISASTER RECOVERY WITH STRETCH CLUSTER FOR OPENSIFT DATA FOUNDATION	85
5.1. REQUIREMENTS FOR ENABLING STRETCH CLUSTER	85
5.2. APPLYING TOPOLOGY ZONE LABELS TO OPENSIFT CONTAINER PLATFORM NODES	86
5.3. INSTALLING LOCAL STORAGE OPERATOR	87
5.4. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR	87
5.5. CREATING OPENSIFT DATA FOUNDATION CLUSTER	89
5.6. VERIFYING OPENSIFT DATA FOUNDATION DEPLOYMENT	93
5.6.1. Verifying the state of the pods	93
5.6.2. Verifying the OpenShift Data Foundation cluster is healthy	95
5.6.3. Verifying the Multicloud Object Gateway is healthy	96
5.6.4. Verifying that the specific storage classes exist	96
5.7. INSTALL ZONE AWARE SAMPLE APPLICATION	96
5.7.1. Scaling the application after installation	100
5.7.2. Modify Deployment to be Zone Aware	101
5.8. RECOVERING OPENSIFT DATA FOUNDATION STRETCH CLUSTER	104
5.8.1. Understanding zone failure	104
5.8.2. Recovering zone-aware HA applications with RWX storage	105
5.8.3. Recovering HA applications with RWX storage	105
5.8.4. Recovering applications with RWO storage	105
5.8.5. Recovering StatefulSet pods	107
CHAPTER 6. MONITORING DISASTER RECOVERY HEALTH	108
6.1. ENABLE MONITORING FOR DISASTER RECOVERY	108
6.2. ENABLING DISASTER RECOVERY DASHBOARD ON HUB CLUSTER	108
6.3. VIEWING HEALTH STATUS OF DISASTER RECOVERY REPLICATION RELATIONSHIPS	109
6.4. DISASTER RECOVERY METRICS	109
6.5. DISASTER RECOVERY ALERTS	111
CHAPTER 7. TROUBLESHOOTING DISASTER RECOVERY	114
7.1. TROUBLESHOOTING METRO-DR	114
7.1.1. A statefulset application stuck after failover	114
7.1.2. DR policies protect all applications in the same namespace	114
7.1.3. During failback of an application stuck in Relocating state	115
7.1.4. Relocate or failback might be stuck in Initiating state	115
7.2. TROUBLESHOOTING REGIONAL-DR	115
7.2.1. rbd-mirror daemon health is in warning state	115

7.2.2. volsync-rsync-src pod is in error state as it is unable to resolve the destination hostname	116
7.2.3. Cleanup and data sync for ApplicationSet workloads remain stuck after older primary managed cluster is recovered post failover	116
7.3. TROUBLESHOOTING 2-SITE STRETCH CLUSTER WITH ARBITER	118
7.3.1. Recovering workload pods stuck in ContainerCreating state post zone recovery	118

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.
- **Disaster Recovery with stretch cluster**
Stretch cluster solution ensures business continuity with no-data loss disaster recovery protection with OpenShift Data Foundation based synchronous replication in a single OpenShift cluster, stretched across two data centers with low latency and one arbiter node.

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. METRO-DR SOLUTION FOR OPENSIFT DATA FOUNDATION

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

The persistent storage for applications is provided by an external Red Hat Ceph Storage (RHCS) cluster stretched between the two locations with the OpenShift Container Platform instances connected to this storage cluster. An arbiter node with a storage monitor service is required at a third location (different location than where OpenShift Container Platform instances are deployed) to establish quorum for the RHCS cluster in the case of a site outage. This third location can be in the range of ~100ms RTT from the storage cluster connected to the OpenShift Container Platform instances.

This is a general overview of the Metro DR steps required to configure and execute OpenShift Disaster Recovery (ODR) capabilities using OpenShift Data Foundation and RHACM across two distinct OpenShift Container Platform clusters separated by distance. In addition to these two clusters called managed clusters, a third OpenShift Container Platform cluster is required that will be the Red Hat Advanced Cluster Management (RHACM) hub cluster.



IMPORTANT

You can now easily set up Metropolitan disaster recovery solutions for workloads based on OpenShift virtualization technology using OpenShift Data Foundation. For more information, see the [knowledgebase article](#).

3.1. 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.

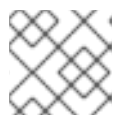
3.2. METRO-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Metro-DR capabilities using the latest versions of Red Hat OpenShift Data Foundation, Red Hat Ceph Storage (RHCS) and Red Hat Advanced Cluster Management for Kubernetes (RHACM) version 2.10 or later, 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 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 [Installing OpenShift Data Foundation 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. 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**

The Metro-DR solution can only have one DRpolicy.

8. Testing your disaster recovery solution with:
 - a. **Subscription-based** application:
 - Create sample applications. See [Creating sample application](#).
 - Test failover and relocate operations using the sample application between managed clusters. See [Subscription-based application failover](#) and [relocating subscription-based application](#).
 - b. **ApplicationSet-based** application:
 - Create sample applications. See [Creating ApplicationSet-based applications](#).
 - Test failover and relocate operations using the sample application between managed clusters. See [ApplicationSet-based application failover](#) and [relocating ApplicationSet-based application](#).

3.3. REQUIREMENTS FOR ENABLING METRO-DR

The prerequisites to installing a disaster recovery solution supported by Red Hat OpenShift Data Foundation are as follows:

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



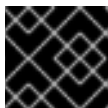
NOTE

For configuring hub recovery setup, you need a 4th cluster which acts as the passive hub. The primary managed cluster (Site-1) can be co-situated with the active RHACM hub cluster while the passive hub cluster is situated along with the secondary managed cluster (Site-2). Alternatively, the active RHACM hub cluster can be placed in a neutral site (Site-3) that is not impacted by the failures of either of the primary managed cluster at Site-1 or the secondary cluster at Site-2. In this situation, if a passive hub cluster is used it can be placed with the secondary cluster at Site-2. For more information, see [Configuring passive hub cluster for hub recovery](#).

Hub recovery is a Technology Preview feature and is subject to Technology Preview support limitations.

- Ensure that RHACM operator and MultiClusterHub is installed on the Hub cluster. See [RHACM installation guide](#) for instructions.

After the operator is successfully installed, a popover with a message that the Web console update is available appears on the user interface. Click **Refresh web console** from this popover for the console changes to reflect.



IMPORTANT

Ensure that application traffic routing and redirection are configured appropriately.

- On the Hub cluster
 - Navigate to **All Clusters** → **Infrastructure** → **Clusters**.
 - 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

The Openshift Container Platform managed clusters and the Red Hat Ceph Storage (RHCS) nodes have distance limitations. The network latency between the sites must be below 10 milliseconds round-trip time (RTT).

3.4. 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 7](#) .



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

Erasasure coded pools cannot be used with stretch mode.

3.4.1. Hardware requirements

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

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

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

3.4.2. Software requirements

Use the latest software version of **Red Hat Ceph Storage 7**

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

3.4.3. 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](#).

3.5. DEPLOYING RED HAT CEPH STORAGE

3.5.1. 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:

- **rhel9-for-x86_64-baseos-rpms**
- **rhel9-for-x86_64-appstream-rpms**

```
subscription-manager repos --disable="*" --enable="rhel9-for-x86_64-baseos-rpms" --
enable="rhel9-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-9-x86_64-rpms** and **rhceph-6-tools-for-rhel-9-x86_64-rpms** repositories:

```
subscription-manager repos --enable="ansible-2.9-for-rhel-9-x86_64-rpms" --
enable="rhceph-6-tools-for-rhel-9-x86_64-rpms"
```

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

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

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

```
$ hostname
```

Example output:

```
ceph1
```

- 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
```

- 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](#).

- Run the following steps on the bootstrap node. In our example, the bootstrap node is **ceph1**.
 - 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
ceph4
EOF
```



NOTE

Here, the Hosts (**Ceph1** and **Ceph4**) belonging to two different data centers are configured as part of the [admin] group on the inventory file and are tagged as **_admin** by **cephadm**. Each of these admin nodes receive the admin ceph keyring during the bootstrap process so that when one data center is down, we can check using the other available admin node.

- d. Verify that **ansible** can access all nodes using the 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. Navigate to the `/usr/share/cephadm-ansible` directory.
- f. Run `ansible-playbook` with relative file paths.

```
$ 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`. For additional information, see [Running the preflight playbook](#)

3.5.2. Cluster bootstrapping and service deployment with cephadm utility

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

```

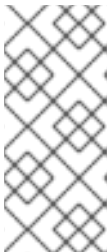
- Retrieve the IP for the NIC with the Red Hat Ceph Storage public network configured from the bootstrap node. After substituting **10.0.40.0** with the subnet that you have defined in your ceph public network, execute the following command.

```
$ 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
```


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.khooot(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 do not 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.2	up	1.00000	1.00000	
-3		0.14650	host	ceph2			
3	ssd	0.14650	osd.3	up	1.00000	1.00000	
-13		0.14650	host	ceph3			
4	ssd	0.14650	osd.4	up	1.00000	1.00000	
-17		0.43950	datacenter	DC2			
-5		0.14650	host	ceph4			
0	ssd	0.14650	osd.0	up	1.00000	1.00000	
-9		0.14650	host	ceph5			

```

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

```

10. Create and enable a new RDB block pool.

```

$ ceph osd pool create 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 have 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.kkxmgc 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
```

3.5.3. 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 by editing the text file **/etc/ceph/crushmap.txt** at the end of the file.

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

-

```
rule stretch_rule {
    id 1
    type replicated
    min_size 1
    max_size 10
    step take default
    step choose firstn 0 type datacenter
    step chooseleaf firstn 2 type host
    step emit
}
# end crush map
```

This example is applicable for active applications in both OpenShift Container Platform clusters.



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 default**
 - Description: Takes the root bucket called **default**, and begins iterating down the tree.

- **step choose firstn 0 type datacenter**
 - Description: Selects the datacenter bucket, and goes into its subtrees.
- **step chooseleaf firstn 2 type host**
 - Description: Selects the number of buckets of the given type. In this case, it is two different hosts located in the datacenter it entered at the previous level.
- **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.

3.6. INSTALLING OPENSIFT DATA FOUNDATION ON MANAGED CLUSTERS

To configure storage replication between the two OpenShift Container Platform clusters, OpenShift Data Foundation operator must be installed first on each managed cluster.

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](#).

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](#).

Use the following flags with the **ceph-external-cluster-details-exporter.py** script.

- a. 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.

- b. 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**.

- c. 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.example.com
host ceph6.example.com
```

Example output:

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

- d. Run the **ceph-external-cluster-details-exporter.py** with the parameters that are configured for the first OpenShift Container Platform managed cluster **cluster1** on bootstrapped node **ceph1**.

```
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> XXX.XXX.XXX.XXX:8080 --run-as-user client.odf.cluster1 > ocp-cluster1.json
```

**NOTE**

Modify the <rgw-endpoint> XXX.XXX.XXX.XXX according to your environment.

- e. Run the **ceph-external-cluster-details-exporter.py** with the parameters that are configured for the first OpenShift Container Platform managed cluster **cluster2** on bootstrapped node **ceph1**.

```
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 XXX.XXX.XXX.XXX:8080 --run-as-user
```

```
client.odf.cluster2 > ocp-cluster2.json
```



NOTE

Modify the <rgw-endpoint> XXX.XXX.XXX.XXX according to your environment.

- 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 OpenShift Container Platform console on **cluster1** where external OpenShift Data Foundation is being deployed.
 - Use the contents of file **ocp-cluster2.json** on the OpenShift Container Platform console on **cluster2** where external OpenShift Data Foundation 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}'
```

Wait for the status result to be Ready for both queries on the **Primary managed cluster** and the **Secondary managed cluster**.

5. On the OpenShift Web Console, navigate to **Installed Operators → OpenShift Data Foundation → Storage System → ocs-external-storagecluster-storagesystem → Resources**. Verify that the **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.
6. Enable read affinity for RBD and CephFS volumes to be served from the nearest datacenter.
 - a. On the Primary managed cluster, label all the nodes.

```
$ oc label nodes --all metro-dr.openshift-storage.topology.io/datacenter=DC1
```

Execute the following commands to enable read affinity:

```
$ oc patch storageclusters.ocs.openshift.io -n openshift-storage ocs-external-storagecluster -p '{"spec":{"csi":{"readAffinity":{"enabled":true,"crushLocationLabels":["metro-dr.openshift-storage.topology.io/datacenter"]}}}}' --type=merge
```

```
$ oc delete po -n openshift-storage -l 'app in (csi-cephfsplugin,csi-rbdplugin)'
```

- b. On the Secondary managed cluster, label all the nodes:

```
$ oc label nodes --all metro-dr.openshift-storage.topology.io/datacenter=DC2
```

Execute the following commands to enable read affinity:

```
$ oc patch storageclusters.ocs.openshift.io -n openshift-storage ocs-external-storagecluster -p '{"spec":{"csi":{"readAffinity":{"enabled":true,"crushLocationLabels":["metro-dr.openshift-storage.topology.io/datacenter"]}}}}' --type=merge
```

```
$ oc delete po -n openshift-storage -l 'app in (csi-cephfsplugin,csi-rbdplugin)'
```

3.7. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multicluster 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 Multicluster Orchestrator**.
2. Click **ODF Multicluster 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	STATUS	RESTARTS	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

3.8. 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

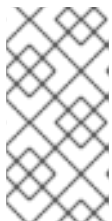
1. Extract the ingress certificate for the Primary managed cluster and save the output to **primary.crt**.

```
$ 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
```

3.9. 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.

Procedure

1. On the **OpenShift console**, navigate to **All Clusters** → **Data Services** → **Data policies**.
2. Click **Create DRPolicy**.
3. Enter **Policy name**. Ensure that each DRPolicy has a unique name (for example: **ocp4perf1-ocp4perf2**).
4. Select two clusters from the list of managed clusters to which this new policy will be associated with.
5. **Replication policy** is automatically set to **sync** based on the OpenShift clusters selected.
6. Click **Create**.

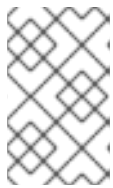
7. Verify that the **DRPolicy** is created successfully. Run this command on the **Hub cluster** for each of the DRPolicy resources created, where `<drpolicy_name>` is replaced with your unique name.

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

Example output:

```
Succeeded
```

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**.



NOTE

Editing of **SchedulingInterval**, **ReplicationClassSelector**, **VolumeSnapshotClassSelector** and **DRClusters** field values are not supported in the DRPolicy.

8. 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:

- b. Check S3 access to each bucket created on each managed cluster. Use the DRCluster validation command, where `<drcluster_name>` is replaced with your unique name.



NOTE

Editing of **Region** and **S3ProfileName** field values are non supported in DRClusters.

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

Example output:

```
Succeeded
```



NOTE

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

9. 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.v4.15.0		OpenShift DR
Cluster Operator 4.15.0	Succeeded	
clusterserviceversion.operators.coreos.com/volsync-product.v0.8.0		VolSync
0.8.0	Succeeded	

NAME	READY	STATUS	RESTARTS	AGE
pod/ramen-dr-cluster-operator-6467cf5d4c-cc8kz	2/2	Running	0	3d12h

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

- Verify that the secret is propagated correctly on the Primary managed cluster and the Secondary managed cluster.

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

Match the output with the s3SecretRef from the Hub cluster:

```
oc get cm -n openshift-operators ramen-hub-operator-config -oyaml
```

3.10. 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.

3.10.1. Add node IP addresses to DRClusters

- 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.

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

```
$ oc get drcluster
```

Example output:

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

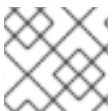
3. Edit each **DRCluster** to add your unique IP addresses after replacing `<drcluster_name>` with your unique name.

```
$ 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).

3.10.2. 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**).

3.11. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION

OpenShift Data Foundation disaster recovery (DR) solution supports disaster recovery for Subscription-based and ApplicationSet-based applications that are managed by RHACM. For more details, see [Subscriptions](#) and [ApplicationSet](#) documentation.

The following sections detail how to create an application and apply a DRPolicy to an application.

- [Subscription-based applications](#)
OpenShift users that do not have cluster-admin permissions, see the [knowledge article](#) on how to assign necessary permissions to an application user for executing disaster recovery actions.
- [ApplicationSet-based applications](#)
OpenShift users that do not have cluster-admin permissions cannot create ApplicationSet-based applications.

3.11.1. Subscription-based applications

3.11.1.1. Creating a sample Subscription-based application

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

Prerequisites

- When creating an application for general consumption, ensure that the application is deployed to ONLY one cluster.
- 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.
- As a best practice, group Red Hat Advanced Cluster Management (RHACM) subscriptions that belong together, refer to a single Placement Rule to DR protect them as a group. Further create them as a single application for a logical grouping of the subscriptions for future DR actions like

failover and relocate.



NOTE

If unrelated subscriptions refer to the same Placement Rule for placement actions, they are also DR protected as the DR workflow controls all subscriptions that references the Placement Rule.

Procedure

1. On the Hub cluster, navigate to **Applications** and click **Create application**.
2. Select type as **Subscription**.
3. Enter your application **Name** (for example, **busybox**) and **Namespace** (for example, **busybox-sample**).
4. In the Repository location for resources section, select **Repository type Git**.
5. 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> where the **Branch** is **release-4.15** and **Path** is **busybox-odr-metro**.
6. Scroll down in the form until you see **Deploy application resources on clusters with all specified labels**.
 - Select the global **Cluster sets** or the one that includes the correct managed clusters for your environment.
 - Add a label `<name>` with its value set to the **managed cluster** name.
7. 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.

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

Log in 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-67bf494b9-zl5tr 1/1   Running 0      77s
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS
MODES STORAGECLASS		AGE		
persistentvolumeclaim/busybox-pvc	Bound	pvc-c732e5fe-daaf-4c4d-99dd-462e04c18412		
5Gi RWO		ocs-storagecluster-ceph-rbd	77s	

3.11.1.2. Apply Data policy to sample application

Prerequisites

- Ensure that both managed clusters referenced in the Data policy are reachable. If not, the application will not be protected for disaster recovery until both clusters are online.

Procedure

1. On the Hub cluster, navigate to **All Clusters** → **Applications**.
2. Click the Actions menu at the end of application to view the list of available actions.
3. Click **Manage data policy** → **Assign data policy**.
4. Select **Policy** and click **Next**.
5. Select an **Application resource** and then use **PVC label selector** to select **PVC label** for the selected application resource.



NOTE

You can select more than one PVC label for the selected application resources. You can also use the **Add application resource** option to add multiple resources.

6. After adding all the application resources, click **Next**.
7. Review the **Policy configuration details** and click **Assign**. The newly assigned Data policy is displayed on the **Manage data policy** modal list view.
8. Verify that you can view the assigned policy details on the Applications page.
 - a. On the Applications page, navigate to the **Data policy** column and click the **policy link** to expand the view.
 - b. Verify that you can see the number of policies assigned along with failover and relocate status.
 - c. Click **View more details** to view the status of ongoing activities with the policy in use with the application.
9. After you apply DRPolicy to the applications, confirm whether the **ClusterDataProtected** is set to **True** in the drpc yaml output.

3.11.2. ApplicationSet-based applications

3.11.2.1. Creating ApplicationSet-based applications

Prerequisite

- Ensure that the Red Hat OpenShift GitOps operator is installed on the Hub cluster. For instructions, see [RHACM documentation](#).
- Ensure that both Primary and Secondary managed clusters are registered to GitOps. For registration instructions, see [Registering managed clusters to GitOps](#). Then check if the Placement used by **GitOpsCluster** resource to register both managed clusters, has the tolerations to deal with cluster unavailability. You can verify if the following tolerations are added to the Placement using the command **oc get placement <placement-name> -n openshift-gitops -o yaml**.

```
tolerations:
- key: cluster.open-cluster-management.io/unreachable
  operator: Exists
- key: cluster.open-cluster-management.io/unavailable
  operator: Exists
```

In case the tolerations are not added, see [Configuring application placement tolerations for Red Hat Advanced Cluster Management and OpenShift GitOps](#).

Procedure

1. On the Hub cluster, navigate to **All Clusters** → **Applications** and click **Create application**.
2. Choose application type as **Argo CD ApplicationSet - Push model**
3. In General step 1, enter your **Application set name**.
4. Select **Argo server openshift-gitops** and **Requeue time** as **180** seconds.
5. Click **Next**.
6. In the Repository location for resources section, select **Repository type Git**.
7. Enter the Git repository URL for the sample application, the github Branch and Path where the resources busybox Pod and PVC will be created.
 - a. Use the sample application repository as <https://github.com/red-hat-storage/ocm-ramen-samples>
 - b. Select **Revision** as **release-4.15**
 - c. Choose Path as **busybox-odr-metro**.
8. Enter **Remote namespace** value. (example, busybox-sample) and click **Next**.
9. Select **Sync policy** settings and click **Next**.
You can choose one or more options.
10. Add a label *<name>* with its value set to the **managed cluster** name.
11. Click **Next**.
12. Review the setting details and click **Submit**.

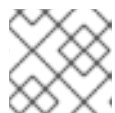
3.11.2.2. Apply Data policy to sample ApplicationSet-based application

Prerequisites

- Ensure that both managed clusters referenced in the Data policy are reachable. If not, the application will not be protected for disaster recovery until both clusters are online.

Procedure

1. On the Hub cluster, navigate to **All Clusters → Applications**.
2. Click the Actions menu at the end of application to view the list of available actions.
3. Click **Manage data policy → Assign data policy**.
4. Select **Policy** and click **Next**.
5. Select an **Application resource** and then use **PVC label selector** to select **PVC label** for the selected application resource.



NOTE

You can select more than one PVC label for the selected application resources.

6. After adding all the application resources, click **Next**.
7. Review the **Policy configuration details** and click **Assign**. The newly assigned Data policy is displayed on the **Manage data policy** modal list view.
8. Verify that you can view the assigned policy details on the Applications page.
 - a. On the Applications page, navigate to the **Data policy** column and click the **policy link** to expand the view.
 - b. Verify that you can see the number of policies assigned along with failover and relocate status.
9. After you apply DRPolicy to the applications, confirm whether the **ClusterDataProtected** is set to **True** in the drpc yaml output.

3.11.3. Deleting sample application

This section provides instructions for deleting the sample application **busybox** using the RHACM console.





IMPORTANT

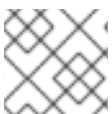
When deleting a DR protected application, access to both clusters that belong to the DRPolicy is required. This is to ensure that all protected API resources and resources in the respective S3 stores are cleaned up as part of removing the DR protection. If access to one of the clusters is not healthy, deleting the **DRPlacementControl** resource for the application, on the hub, would remain in the Deleting state.

Prerequisites

- These 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, delete the **DRPlacementControl** if it is not auto-deleted after deleting the **busybox** application.
 - a. Log in to the OpenShift Web console for the Hub cluster and navigate to Installed Operators for the project **busybox-sample**.
For ApplicationSet applications, select the project as **openshift-gitops**.
 - b. Click **OpenShift DR Hub Operator** and then click the **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.

3.12. SUBSCRIPTION-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

Perform a failover when a managed cluster becomes unavailable, due to any reason. This failover method is application-based.

Prerequisites

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update.

1. Navigate to the **RHACM console** → **Infrastructure** → **Clusters** → **Cluster listtab**.
2. Check the status of both the managed clusters individually before performing failover operation.
However, failover operation can still be performed when the cluster you are failing over to is in a *Ready* state.

Procedure

1. Enable fencing on the **Hub cluster**.
 - a. Open CLI terminal and edit the **DRCluster resource**, where `<drcluster_name>` is your unique name.

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.

```
$ 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
```

- b. Verify the fencing status on the **Hub cluster** for the **Primary managed cluster**, replacing `<drcluster_name>` is your unique identifier.

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

Example output:

```
Fenced
```

- c. Verify that the IPs that belong to the OpenShift Container Platform cluster nodes are now in the blacklist.

```
$ ceph osd blacklist ls
```

Example output

```
cidr:10.1.161.1:0/32 2028-10-30T22:30:03.585634+0000
cidr:10.1.161.14:0/32 2028-10-30T22:30:02.483561+0000
cidr:10.1.161.51:0/32 2028-10-30T22:30:01.272267+0000
cidr:10.1.161.63:0/32 2028-10-30T22:30:05.099655+0000
cidr:10.1.161.129:0/32 2028-10-30T22:29:58.335390+0000
cidr:10.1.161.130:0/32 2028-10-30T22:29:59.861518+0000
```

2. On the Hub cluster, navigate to **Applications**.
3. Click the **Actions** menu at the end of application row to view the list of available actions.
4. Click **Failover application**.
5. After the **Failover application** modal is shown, select **policy** and **target cluster** to which the associated application will failover in case of a disaster.
6. Click the **Select subscription group** dropdown to verify the default selection or modify this setting.
By default, the subscription group that replicates for the application resources is selected.
7. Check the status of the **Failover readiness**.
 - If the status is **Ready** with a green tick, it indicates that the target cluster is ready for failover to start. Proceed to step 7.
 - If the status is **Unknown** or **Not ready**, then wait until the status changes to **Ready**.
8. Click **Initiate**. The busybox application is now failing over to the **Secondary-managed cluster**.
9. Close the modal window and track the status using the **Data policy** column on the Applications page.
10. Verify that the activity status shows as **FailedOver** for the application.
 - a. Navigate to the **Applications** → **Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, click the **View more details** link.

3.13. APPLICATIONSET-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

Perform a failover when a managed cluster becomes unavailable, due to any reason. This failover method is application-based.

Prerequisites

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update.

1. Navigate to the **RHACM console** → **Infrastructure** → **Clusters** → **Cluster listtab**.
2. Check the status of both the managed clusters individually before performing failover operation.
However, failover operation can still be performed when the cluster you are failing over to is in a *Ready* state.

Procedure

1. Enable fencing on the **Hub cluster**.
 - a. Open CLI terminal and edit the **DRCluster resource**, where *<drcluster_name>* is your unique name.

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.

```
$ 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
```

- b. Verify the fencing status on the **Hub cluster** for the **Primary managed cluster**, replacing *<drcluster_name>* is your unique identifier.

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

Example output:

```
Fenced
```

- c. Verify that the IPs that belong to the OpenShift Container Platform cluster nodes are now in the blacklist.

```
$ ceph osd blacklist ls
```

Example output

```
cidr:10.1.161.1:0/32 2028-10-30T22:30:03.585634+0000
cidr:10.1.161.14:0/32 2028-10-30T22:30:02.483561+0000
cidr:10.1.161.51:0/32 2028-10-30T22:30:01.272267+0000
cidr:10.1.161.63:0/32 2028-10-30T22:30:05.099655+0000
cidr:10.1.161.129:0/32 2028-10-30T22:29:58.335390+0000
cidr:10.1.161.130:0/32 2028-10-30T22:29:59.861518+0000
```

2. On the Hub cluster, navigate to **Applications**.
3. Click the **Actions** menu at the end of application row to view the list of available actions.
4. Click **Failover application**.
5. When the **Failover application** modal is shown, verify the details presented are correct and check the status of the Failover **readiness**. If the status is **Ready** with a green tick, it indicates that the target cluster is ready for failover to start.
6. Click **Initiate**. The busybox resources are now created on the target cluster.
7. Close the modal window and track the status using the **Data policy** column on the Applications page.
8. Verify that the activity status shows as **FailedOver** for the application.
 - a. Navigate to the **Applications** → **Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, verify that you can see one or more policy names and the ongoing activities associated with the policy in use with the application.

3.14. RELOCATING SUBSCRIPTION-BASED APPLICATION BETWEEN MANAGED CLUSTERS

Relocate an application to its preferred location when all managed clusters are available.

Prerequisite

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update. Relocate can only be performed when both primary and preferred clusters are up and running.
 1. Navigate to **RHACM console** → **Infrastructure** → **Clusters** → **Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing relocate operation.
- Verify that applications were cleaned up from the cluster before un fencing it.

Procedure

1. Disable fencing on the Hub cluster.
 - a. Edit the **DRCluster resource** for this cluster, replacing <drcluster_name> with a 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
```

- b. 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 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 uncoron operations on the nodes.

- c. 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          NAME
READY STATUS      RESTARTS  AGE
```

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**.

- d. Verify that the **Unfenced** cluster is in a healthy state. Validate the fencing status in the Hub cluster for the Primary-managed cluster, replacing <drcluster_name> with a unique name.

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

Example output:

```
Unfenced
```

- e. Verify that the IPs that belong to the OpenShift Container Platform cluster nodes are NOT in the blocklist.

```
$ ceph osd blocklist ls
```

Ensure that you do not see the IPs added during fencing.

2. On the Hub cluster, navigate to **Applications**.
3. Click the **Actions** menu at the end of application row to view the list of available actions.
4. Click **Relocate application**.
5. When the **Relocate application** modal is shown, select **policy** and **target cluster** to which the associated application will relocate to in case of a disaster.
6. By default, the subscription group that will deploy the application resources is selected. Click the **Select subscription group** dropdown to verify the default selection or modify this setting.
7. Check the status of the **Relocation readiness**.
 - If the status is **Ready** with a green tick, it indicates that the target cluster is ready for relocation to start. Proceed to step 7.
 - If the status is **Unknown** or **Not ready**, then wait until the status changes to **Ready**.
8. Click **Initiate**. The busybox resources are now created on the target cluster.
9. Close the modal window and track the status using the **Data policy** column on the Applications page.
10. Verify that the activity status shows as **Relocated** for the application.
 - a. Navigate to the **Applications → Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, click the **View more details** link.

3.15. RELOCATING AN APPLICATIONSET-BASED APPLICATION BETWEEN MANAGED CLUSTERS

Relocate an application to its preferred location when all managed clusters are available.

Prerequisite

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update. Relocate can only be performed when both primary and preferred clusters are up and running.
 1. Navigate to **RHACM console** → **Infrastructure** → **Clusters** → **Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing relocate operation.
- Verify that applications were cleaned up from the cluster before unfencing it.

Procedure

1. Disable fencing on the Hub cluster.
 - a. Edit the **DRCluster resource** for this cluster, replacing <drcluster_name> with a 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
```

- b. 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 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 uncordeon operations on the nodes.

- c. 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          NAME
READY STATUS    RESTARTS   AGE
```

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**.

- d. Verify that the **Unfenced** cluster is in a healthy state. Validate the fencing status in the Hub cluster for the Primary-managed cluster, replacing <drcluster_name> with a unique name.

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

Example output:

```
Unfenced
```

- e. Verify that the IPs that belong to the OpenShift Container Platform cluster nodes are NOT in the blocklist.

```
$ ceph osd blocklist ls
```

Ensure that you do not see the IPs added during fencing.

2. On the Hub cluster, navigate to **Applications**.
3. Click the **Actions** menu at the end of application row to view the list of available actions.
4. Click **Relocate application**.

5. When the **Relocate application** modal is shown, select **policy** and **target cluster** to which the associated application will relocate to in case of a disaster.
6. Click **Initiate**. The busybox resources are now created on the target cluster.
7. Close the modal window and track the status using the **Data policy** column on the Applications page.
8. Verify that the activity status shows as **Relocated** for the application.
 - a. Navigate to the **Applications → Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, verify that you can see one or more policy names and the relocation status associated with the policy in use with the application.

3.16. RECOVERING TO A REPLACEMENT CLUSTER WITH METRO-DR

When there is a failure with the primary cluster, you get the options to either repair, wait for the recovery of the existing cluster, or replace the cluster entirely if the cluster is irredeemable. This solution guides you when replacing a failed primary cluster with a new cluster and enables failback (relocate) to this new cluster.

In these instructions, we are assuming that a RHACM managed cluster must be replaced after the applications have been installed and protected. For purposes of this section, the RHACM managed cluster is the **replacement cluster**, while the cluster that is not replaced is the **surviving cluster** and the new cluster is the **recovery cluster**.

Prerequisite

- Ensure that the Metro-DR environment has been configured with applications installed using Red Hat Advance Cluster Management (RHACM).
- Ensure that the applications are assigned a Data policy which protects them against cluster failure.

Procedure

1. Perform the following steps on the **Hub cluster**:
 - a. Fence the replacement cluster by using the CLI terminal to edit the **DRCluster** resource, where `<drcluster_name>` is the replacement cluster name.

```
oc edit drcluster <drcluster_name>

apiVersion: ramendr.openshift.io/v1alpha1
kind: DRCluster
metadata:
[...]
spec:
  ## Add or modify this line
  clusterFence: Fenced
  cidrs:
  [...]
  [...]
```

-
- b. Using the RHACM console, navigate to **Applications** and failover all protected applications from the failed cluster to the surviving cluster.
- c. Verify and ensure that all protected applications are now running on the surviving cluster.

**NOTE**

The **PROGRESSION** state for each application DRPlacementControl will show as **Cleaning Up**. This is expected if the replacement cluster is offline or down.

2. Unfence the replacement cluster.

Using the CLI terminal, edit the DRCluster resource, where `<drcluster_name>` is the replacement cluster name.

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

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

spec:

```
  ## Modify this line
  clusterFence: Unfenced
  cidrs:
  [...]
```

3. Delete the DRCluster for the replacement cluster.

```
$ oc delete drcluster <drcluster_name> --wait=false
```

**NOTE**

Use `--wait=false` since the DRCluster will not be deleted until a later step.

4. Disable disaster recovery on the **Hub cluster** for each protected application on the surviving cluster.
 - a. For each application, edit the Placement and ensure that the surviving cluster is selected.

**NOTE**

For Subscription-based applications the associated Placement can be found in the same namespace on the hub cluster similar to the managed clusters. For ApplicationSets-based applications the associated Placement can be found in the **openshift-gitops** namespace on the hub cluster.

```
$ oc edit placement <placement_name> -n <namespace>
```



```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
annotations:
  cluster.open-cluster-management.io/experimental-scheduling-disable: "true"
[...]
spec:
clusterSets:
- submariner
predicates:
- requiredClusterSelector:
  claimSelector: {}
  labelSelector:
  matchExpressions:
  - key: name
    operator: In
    values:
  - cluster1 <-- Modify to be surviving cluster name
[...]

```

- b. Verify that the **s3Profile** is removed for the replacement cluster by running the following command on the surviving cluster for each protected application's VolumeReplicationGroup.

```
$ oc get vrg -n <application_namespace> -o jsonpath='{.items[0].spec.s3Profiles}' | jq
```

- c. After the protected application **Placement** resources are all configured to use the surviving cluster and replacement cluster s3Profile(s) removed from protected applications, all **DRPlacementControl** resources must be deleted from the **Hub cluster**.

```
$ oc delete drpc <drpc_name> -n <namespace>
```



NOTE

For Subscription-based applications the associated DRPlacementControl can be found in the same namespace as the managed clusters on the hub cluster. For ApplicationSets-based applications the associated DRPlacementControl can be found in the **openshift-gitops** namespace on the hub cluster.

- d. Verify that all DRPlacementControl resources are deleted before proceeding to the next step. This command is a query across all namespaces. There should be no resources found.

```
$ oc get drpc -A
```

- e. The last step is to edit each applications **Placement** and remove the annotation **cluster.open-cluster-management.io/experimental-scheduling-disable: "true"**.

```
$ oc edit placement <placement_name> -n <namespace>
```

```

apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:

```

```

annotations:
  ## Remove this annotation
  cluster.open-cluster-management.io/experimental-scheduling-disable: "true"
[...]

```

5. Repeat the process detailed in the last step and the sub-steps for every protected application on the surviving cluster. Disabling DR for protected applications is now completed.
6. On the Hub cluster, run the following script to remove all disaster recovery configurations from the **surviving cluster** and the **hub cluster**.

```

#!/bin/bash
secrets=$(oc get secrets -n openshift-operators | grep Opaque | cut -d" " -f1)
echo $secrets
for secret in $secrets
do
  oc patch -n openshift-operators secret/$secret -p '{"metadata":{"finalizers":null}}' --
  type=merge
done
mirrorpeers=$(oc get mirrorpeer -o name)
echo $mirrorpeers
for mp in $mirrorpeers
do
  oc patch $mp -p '{"metadata":{"finalizers":null}}' --type=merge
  oc delete $mp
done
drpolicies=$(oc get drpolicy -o name)
echo $drpolicies
for drp in $drpolicies
do
  oc patch $drp -p '{"metadata":{"finalizers":null}}' --type=merge
  oc delete $drp
done
drclusters=$(oc get drcluster -o name)
echo $drclusters
for drp in $drclusters
do
  oc patch $drp -p '{"metadata":{"finalizers":null}}' --type=merge
  oc delete $drp
done
oc delete project openshift-operators
managedclusters=$(oc get managedclusters -o name | cut -d"/" -f2)
echo $managedclusters
for mc in $managedclusters
do
  secrets=$(oc get secrets -n $mc | grep multicluster.odf.openshift.io/secret-type | cut -d" " -
f1)
  echo $secrets
  for secret in $secrets
  do
    set -x
    oc patch -n $mc secret/$secret -p '{"metadata":{"finalizers":null}}' --type=merge
    oc delete -n $mc secret/$secret
  done
done

```

```
done
```

```
oc delete clusterrolebinding spoke-clusterrole-bindings
```



NOTE

This script used the command **oc delete project openshift-operators** to remove the Disaster Recovery (DR) operators in this namespace on the hub cluster. If there are other non-DR operators in this namespace, you must install them again from OperatorHub.

7. After the namespace **openshift-operators** is automatically created again, add the monitoring label back for collecting the disaster recovery metrics.

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

8. On the surviving cluster, ensure that the object bucket created during the DR installation is deleted. Delete the object bucket if it was not removed by script. The name of the object bucket used for DR starts with **odrbucket**.

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

9. On the RHACM console, navigate to **Infrastructure** → **Clusters view**.
 - a. Detach the replacement cluster.
 - b. Create a new OpenShift cluster (recovery cluster) and import the new cluster into the RHACM console. For instructions, see [Creating a cluster](#) and [Importing a target managed cluster to the hub cluster](#).
10. Install OpenShift Data Foundation operator on the recovery cluster and connect it to the same external Ceph storage system as the surviving cluster. For detailed instructions, refer to [Deploying OpenShift Data Foundation in external mode](#).



NOTE

Ensure that the OpenShift Data Foundation version is 4.15 (or greater) and the same version of OpenShift Data Foundation is on the surviving cluster.

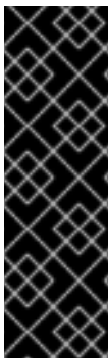
11. On the hub cluster, install the ODF Multicluster Orchestrator operator from OperatorHub. For instructions, see chapter on [Installing OpenShift Data Foundation Multicluster Orchestrator operator](#).
12. Using the RHACM console, navigate to **Data Services** → **Data policies**.
 - a. Select **Create DRPolicy** and name your policy.
 - b. Select the **recovery cluster** and the **surviving cluster**.
 - c. Create the policy. For instructions see chapter on [Creating Disaster Recovery Policy on Hub cluster](#).

Proceed to the next step only after the status of DRPolicy changes to **Validated**.

13. Apply the DRPolicy to the applications on the surviving cluster that were originally protected before the replacement cluster failed.
14. Relocate the newly protected applications on the surviving cluster back to the new recovery (primary) cluster. Using the RHACM console, navigate to the **Applications** menu to perform the relocation.

3.17. HUB RECOVERY USING RED HAT ADVANCED CLUSTER MANAGEMENT [TECHNOLOGY PREVIEW]

When your setup has active and passive Red Hat Advanced Cluster Management for Kubernetes (RHACM) hub clusters, and in case where the active hub is down, you can use the passive hub to failover or relocate the disaster recovery protected workloads.



IMPORTANT

Hub recovery is a Technology Preview feature and is subject to Technology Preview support limitations. 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.

For more information, see [Technology Preview Features Support Scope](#).

3.17.1. Configuring passive hub cluster

To perform hub recovery in case the active hub is down or unreachable, follow the procedure in this section to configure the passive hub cluster and then failover or relocate the disaster recovery protected workloads.

Procedure

1. Ensure that RHACM operator and **MultiClusterHub** is installed on the passive hub cluster. See [RHACM installation guide](#) for instructions.
After the operator is successfully installed, a popover with a message that the Web console update is available appears on the user interface. Click **Refresh web console** from this popover for the console changes to reflect.
2. Before hub recovery, configure backup and restore. See [Backup and restore](#) topics of *RHACM Business continuity* guide.
3. Install the multicluster orchestrator (MCO) operator along with Red Hat OpenShift GitOps operator on the passive RHACM hub prior to the restore. For instructions to restore your RHACM hub, see [Installing OpenShift Data Foundation Multicluster Orchestrator operator](#).
4. Ensure that **.spec.cleanupBeforeRestore** is set to **None** for the **Restore.cluster.open-cluster-management.io** resource. For details, see [Restoring passive resources while checking for backups](#) chapter of RHACM documentation.
5. If SSL access across clusters was configured manually during setup, then re-configure SSL access across clusters. For instructions, see [Configuring SSL access across clusters](#) chapter.
6. On the passive hub, add the monitoring label for collecting the disaster recovery metrics. For alert details, see [Disaster recovery alerts](#) chapter.

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

3.17.2. Switching to passive hub cluster

Use this procedure when active hub is down or unreachable.

Procedure

1. Restore the backups on the passive hub cluster. For information, see [Restoring a hub cluster](#) from backup.



IMPORTANT

Recovering a failed hub to its passive instance will only restore applications and their DR protected state to its last scheduled backup. Any application that was DR protected after the last scheduled backup would need to be protected again on the new hub.

2. Verify that the Primary and Secondary managed clusters are successfully imported into the RHACM console and they are accessible. If any of the managed clusters are down or unreachable then they will not be successfully imported.
3. Wait until DRPolicy validation succeeds.
4. Verify that the **DRPolicy** is created successfully. Run this command on the **Hub cluster** for each of the DRPolicy resources created, where *<drpolicy_name>* is replaced with a unique name.

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

Example output:

```
Succeeded
```

5. Refresh the RHACM console to make the DR monitoring dashboard tab accessible if it was enabled on the Active hub cluster.
6. If only the active hub cluster is down, restore the hub by performing hub recovery, and restoring the backups on the passive hub. If the managed clusters are still accessible, no further action is required.
7. If the primary managed cluster is down, along with the active hub cluster, you need to fail over the workloads from the primary managed cluster to the secondary managed cluster. For failover instructions, based on your workload type, see [Subscription-based applications](#) or [ApplicationSet-based applications](#).
8. Verify that the failover is successful. When the Primary managed cluster is down, then the PROGRESSION status for the workload would be in **Cleaning Up** phase until the down managed cluster is back online and successfully imported into the RHACM console. On the passive hub cluster, run the following command to check the PROGRESSION status.

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

Example output:

```

NAMESPACE          NAME                               AGE  PREFERREDCLUSTER
FAILOVERCLUSTER    DESIREDSTATE  CURRENTSTATE  PROGRESSION  START
TIME              DURATION      PEER READY
[...]
busybox            cephfs-busybox-placement-1-drpc    103m  cluster-1        cluster-2
Failover          FailedOver    Cleaning Up   2024-04-15T09:12:23Z                False
busybox            cephfs-busybox-placement-1-drpc    102m  cluster-1
Deployed          Completed     2024-04-15T07:40:09Z  37.200569819s  True
[...]

```

CHAPTER 4. REGIONAL-DR SOLUTION FOR OPENSIFT DATA FOUNDATION

4.1. 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: components that run on the multi-cluster control plane.
- Managed clusters: components that run on the clusters that are managed.

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

OpenShift Data Foundation

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

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

OpenShift Data Foundation stack is 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

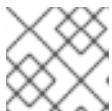
- **OpenShift DR Hub Operator:** Automatically installed as part of ODF Multicluster Orchestrator installation on the hub cluster to orchestrate failover or relocation of DR enabled applications.
- **OpenShift DR Cluster Operator:** Automatically installed on each managed cluster that is part of a Metro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

4.2. REGIONAL-DR DEPLOYMENT WORKFLOW

This section provides an overview of the steps required to configure and deploy Regional-DR capabilities using the 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. 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.

6. Testing your disaster recovery solution with:
 - a. **Subscription-based** application:
 - Create Subscription-based applications. See [Creating sample application](#).
 - Test failover and relocate operations using the sample subscription-based application between managed clusters. See [Subscription-based application failover](#) and [relocating subscription-based application](#).
 - b. **ApplicationSet-based** application:
 - Create sample applications. See [Creating ApplicationSet-based applications](#).
 - Test failover and relocate operations using the sample application between managed clusters. See [ApplicationSet-based application failover](#) and [relocating ApplicationSet-based application](#).

4.3. REQUIREMENTS FOR ENABLING REGIONAL-DR

The prerequisites to installing a disaster recovery solution supported by Red Hat OpenShift Data Foundation are as follows:

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



NOTE

For configuring hub recovery setup, you need a 4th cluster which acts as the passive hub. The primary managed cluster (Site-1) can be co-situated with the active RHACM hub cluster while the passive hub cluster is situated along with the secondary managed cluster (Site-2). Alternatively, the active RHACM hub cluster can be placed in a neutral site (Site-3) that is not impacted by the failures of either of the primary managed cluster at Site-1 or the secondary cluster at Site-2. In this situation, if a passive hub cluster is used it can be placed with the secondary cluster at Site-2. For more information, see [Configuring passive hub cluster for hub recovery](#).

Hub recovery is a Technology Preview feature and is subject to Technology Preview support limitations.

- Ensure that RHACM operator and MultiClusterHub is installed on the Hub cluster. See [RHACM installation guide](#) for instructions. After the operator is successfully installed, a popover with a message that the Web console update is available appears on the user interface. Click **Refresh web console** from this popover for the console changes to reflect.



IMPORTANT

Ensure that application traffic routing and redirection are configured appropriately.

- On the Hub cluster
 - Navigate to **All Clusters → Infrastructure → Clusters**.
 - Import or create the **Primary managed cluster** and the **Secondary managed cluster** using the RHACM console.
 - Choose the appropriate options for your environment.

For instructions, see [Creating a cluster](#) and [Importing a target managed cluster to the hub cluster](#).

- Connect the private OpenShift cluster and service networks using the RHACM Submariner add-ons. Verify that the two clusters have non-overlapping service and cluster private networks. Otherwise, ensure that the Globalnet is enabled during the Submariner add-ons installation. Run the following command for each of the managed clusters to determine if Globalnet needs to be enabled. The example shown here is for non-overlapping cluster and service networks so Globalnet would not be enabled.

```
$ 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": "OVNKubernetes",
  "serviceNetwork": [
    "10.15.0.0/16"
  ]
}
```

Example output for Secondary cluster:

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

For more information, see [Submariner documentation](#).

4.4. 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).



NOTE

While creating the storage cluster, in the **Data Protection** step, you must select the **Prepare cluster for disaster recovery (Regional-DR only)** checkbox.

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.

3. In the OpenShift Web Console, navigate to **Installed Operators → OpenShift Data Foundation → Storage System → ocs-storagecluster-storagesystem → Resources** and verify that the **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.
4. [Optional] If Globalnet was enabled when Submariner was installed, then edit the **StorageCluster** after the OpenShift Data Foundation install finishes. For Globalnet networks, manually edit the **StorageCluster** yaml to add the **clusterID** and set **enabled** to **true**. Replace `<clustername>` with your RHACM imported or newly created managed cluster name. Edit the **StorageCluster** on both the Primary managed cluster and the Secondary managed cluster.



WARNING

Do not make this change in the **StorageCluster** unless you enabled Globalnet when Submariner was installed.

```
$ oc edit storagecluster -o yaml -n openshift-storage
```

```
spec:
  network:
    multiClusterService:
      clusterID: <clustername>
      enabled: true
```

5. After the above changes are made,

- a. Wait for the OSD pods to restart and OSD services to be created.
- b. Wait for all MONS to failover.
- c. Ensure that the MONS and OSD services are exported.

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

```
NAME          AGE
rook-ceph-mon-d 4d14h
rook-ceph-mon-e 4d14h
rook-ceph-mon-f 4d14h
rook-ceph-osd-0 4d14h
rook-ceph-osd-1 4d14h
rook-ceph-osd-2 4d14h
```

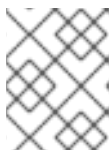
- d. Ensure that cluster is in a **Ready** state and cluster health has a green tick indicating **Health ok**. Verify using step 3.

4.5. INSTALLING OPENSIFT DATA FOUNDATION MULTICLUSTER ORCHESTRATOR OPERATOR

OpenShift Data Foundation Multicluster 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 Multicluster Orchestrator**.
2. Click **ODF Multicluster 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.

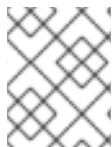
```
$ oc get pods -n openshift-operators
```

Example output:

```
NAME                                READY STATUS  RESTARTS  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
```

4.6. 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
```

4.7. 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.

Procedure

1. On the **OpenShift console**, navigate to **All Clusters** → **Data Services** → **Data policies**.
2. Click **Create DRPolicy**.
3. Enter **Policy name**. Ensure that each DRPolicy has a unique name (for example: **ocp4bos1-ocp4bos2-5m**).

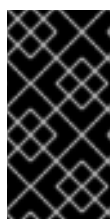
4. Select two clusters from the list of managed clusters to which this new policy will be associated with.



NOTE

If you get an error message "OSDs not migrated" after selecting the clusters, then follow the instructions from knowledgebase article on [Migration of existing OSD to the optimized OSD in OpenShift Data Foundation for Regional-DR cluster](#) before proceeding with the next step.

5. **Replication policy** is automatically set to **Asynchronous**(async) based on the OpenShift clusters selected and a **Sync schedule** option will become available.
6. 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.

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, where `<drpolicy_name>` is replaced with your unique name.

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

Example output:

```
Succeeded
```

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**.



NOTE

Editing of **SchedulingInterval**, **ReplicationClassSelector**, **VolumeSnapshotClassSelector** and **DRClusters** field values are not supported in the DRPolicy.

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
ocp4bos1  4m42s
ocp4bos2  4m42s
```

- b. Check S3 access to each bucket created on each managed cluster. Use the DRCluster validation command, where `<drcluster_name>` is replaced with your unique name.



NOTE

Editing of **Region** and **S3ProfileName** field values are non supported in DRClusters.

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

Example output:

```
Succeeded
```



NOTE

Make sure to run commands 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.v4.15.0  Openshift DR
Cluster Operator  4.15.0          Succeeded
clusterserviceversion.operators.coreos.com/volsync-product.v0.8.0        VolSync
0.8.0             Succeeded

NAME                                     READY STATUS  RESTARTS  AGE
pod/ramen-dr-cluster-operator-6467cf5d4c-cc8kz  2/2  Running  0         3d12h
```

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



NOTE

On the initial run, VolSync operator is installed automatically. VolSync is used to set up volume replication between two clusters to protect CephFs-based PVCs. The replication feature is enabled by default.

11. Verify that the status of the OpenShift Data Foundation 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}'{"\n"}
```

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**.

4.8. CREATE SAMPLE APPLICATION FOR TESTING DISASTER RECOVERY SOLUTION

OpenShift Data Foundation disaster recovery (DR) solution supports disaster recovery for Subscription-based and ApplicationSet-based applications that are managed by RHACM. For more details, see [Subscriptions](#) and [ApplicationSet](#) documentation.

The following sections detail how to create an application and apply a DRPolicy to an application.

- [Subscription-based applications](#)
OpenShift users that do not have cluster-admin permissions, see the [knowledge article](#) on how to assign necessary permissions to an application user for executing disaster recovery actions.
- [ApplicationSet-based applications](#)
OpenShift users that do not have cluster-admin permissions cannot create ApplicationSet-based applications.

4.8.1. Subscription-based applications

4.8.1.1. Creating a sample Subscription-based application

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

Prerequisites

- When creating an application for general consumption, ensure that the application is deployed to **ONLY** one cluster.
- 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.
- As a best practice, group Red Hat Advanced Cluster Management (RHACM) subscriptions that belong together, refer to a single Placement Rule to DR protect them as a group. Further create them as a single application for a logical grouping of the subscriptions for future DR actions like failover and relocate.

**NOTE**

If unrelated subscriptions refer to the same Placement Rule for placement actions, they are also DR protected as the DR workflow controls all subscriptions that references the Placement Rule.

Procedure

1. On the Hub cluster, navigate to **Applications** and click **Create application**.
2. Select type as **Subscription**.
3. Enter your application **Name** (for example, **busybox**) and **Namespace** (for example, **busybox-sample**).
4. In the Repository location for resources section, select **Repository type Git**.
5. 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>.
 - Select **Branch** as **release-4.15**.
 - Choose one of the following **Path**:
 - **busybox-odr** to use RBD Regional-DR.
 - **busybox-odr-cephfs** to use CephFS Regional-DR.
6. Scroll down in the form until you see **Deploy application resources on clusters with all specified labels**.
 - Select the global **Cluster sets** or the one that includes the correct managed clusters for your environment.
 - Add a label `<name>` with its value set to the **managed cluster** name.
7. 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.

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

Log in 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-67bf494b9-zl5tr  1/1   Running  0         77s

```

```

NAME                STATUS  VOLUME                                     CAPACITY  ACCESS
MODES STORAGECLASS    AGE
persistentvolumeclaim/busybox-pvc  Bound  pvc-c732e5fe-daa4-4c4d-99dd-462e04c18412
5Gi   RWO             ocs-storagecluster-ceph-rbd  77s

```

4.8.1.2. Apply Data policy to sample application

Prerequisites

- Ensure that both managed clusters referenced in the Data policy are reachable. If not, the application will not be protected for disaster recovery until both clusters are online.

Procedure

1. On the Hub cluster, navigate to **All Clusters** → **Applications**.
2. Click the Actions menu at the end of application to view the list of available actions.
3. Click **Manage data policy** → **Assign data policy**.
4. Select **Policy** and click **Next**.
5. Select an **Application resource** and then use **PVC label selector** to select **PVC label** for the selected application resource.



NOTE

You can select more than one PVC label for the selected application resources. You can also use the **Add application resource** option to add multiple resources.

6. After adding all the application resources, click **Next**.
7. Review the **Policy configuration details** and click **Assign**. The newly assigned Data policy is displayed on the **Manage data policy** modal list view.
8. Verify that you can view the assigned policy details on the Applications page.
 - a. On the Applications page, navigate to the **Data policy** column and click the **policy link** to expand the view.
 - b. Verify that you can see the number of policies assigned along with failover and relocate status.
 - c. Click **View more details** to view the status of ongoing activities with the policy in use with the application.
9. Optional: Verify RADOS block device (RBD) **volumereplication** and **volumereplicationgroup** on the primary cluster.



```
$ oc get volumereplications.replication.storage.openshift.io -A
```

Example output:

```
NAME          AGE  VOLUMEREPLICATIONCLASS          PVCNAME
DESIREDSTATE  CURRENTSTATE
busybox-pvc   2d16h  rbd-volumereplicationclass-1625360775  busybox-pvc   primary
Primary
```

```
$ oc get volumereplicationgroups.ramendr.openshift.io -A
```

Example output:

```
NAME          DESIREDSTATE  CURRENTSTATE
busybox-drpc  primary      Primary
```

- Optional: Verify CephFS volsync replication source has been set up successfully in the primary cluster and VolSync ReplicationDestination has been set up in the failover cluster.

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

Example output:

```
NAME          SOURCE          LAST SYNC          DURATION          NEXT SYNC
busybox-pvc   busybox-pvc     2022-12-20T08:46:07Z  1m7.794661104s   2022-12-20T08:50:00Z
```

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

Example output:

```
NAME          LAST SYNC          DURATION          NEXT SYNC
busybox-pvc   2022-12-20T08:46:32Z  4m39.52261108s
```

4.8.2. ApplicationSet-based applications

4.8.2.1. Creating ApplicationSet-based applications

Prerequisite

- Ensure that the Red Hat OpenShift GitOps operator is installed on the Hub cluster. For instructions, see [RHACM documentation](#).
- Ensure that both Primary and Secondary managed clusters are registered to GitOps. For registration instructions, see [Registering managed clusters to GitOps](#). Then check if the Placement used by **GitOpsCluster** resource to register both managed clusters, has the tolerations to deal with cluster unavailability. You can verify if the following tolerations are added to the Placement using the command **oc get placement <placement-name> -n openshift-gitops -o yaml**.

```
tolerations:
```

- key: cluster.open-cluster-management.io/unreachable
operator: Exists
- key: cluster.open-cluster-management.io/unavailable
operator: Exists

In case the tolerations are not added, see [Configuring application placement tolerations for Red Hat Advanced Cluster Management and OpenShift GitOps](#).

Procedure

1. On the Hub cluster, navigate to **All Clusters** → **Applications** and click **Create application**.
2. Choose application type as **Argo CD ApplicationSet - Push model**
3. In General step 1, enter your **Application set name**.
4. Select **Argo server openshift-gitops** and **Requeue time** as **180** seconds.
5. Click **Next**.
6. In the Repository location for resources section, select **Repository type Git**.
7. Enter the Git repository URL for the sample application, the github Branch and Path where the resources busybox Pod and PVC will be created.
 - a. Use the sample application repository as <https://github.com/red-hat-storage/ocm-ramen-samples>
 - b. Select **Revision** as **release-4.15**
 - c. Choose one of the following Path:
 - **busybox-odr** to use RBD Regional-DR.
 - **busybox-odr-cephfs** to use CephFS Regional-DR.
8. Enter **Remote namespace** value. (example, busybox-sample) and click **Next**.
9. Select **Sync policy** settings and click **Next**.
You can choose one or more options.
10. Add a label `<name>` with its value set to the **managed cluster** name.
11. Click **Next**.
12. Review the setting details and click **Submit**.

4.8.2.2. Apply Data policy to sample ApplicationSet-based application

Prerequisites

- Ensure that both managed clusters referenced in the Data policy are reachable. If not, the application will not be protected for disaster recovery until both clusters are online.

Procedure

1. On the Hub cluster, navigate to **All Clusters** → **Applications**.
2. Click the Actions menu at the end of application to view the list of available actions.
3. Click **Manage data policy** → **Assign data policy**.
4. Select **Policy** and click **Next**.
5. Select an **Application resource** and then use **PVC label selector** to select **PVC label** for the selected application resource.

**NOTE**

You can select more than one PVC label for the selected application resources.

6. After adding all the application resources, click **Next**.
7. Review the **Policy configuration details** and click **Assign**. The newly assigned Data policy is displayed on the **Manage data policy** modal list view.
8. Verify that you can view the assigned policy details on the Applications page.
 - a. On the Applications page, navigate to the **Data policy** column and click the **policy link** to expand the view.
 - b. Verify that you can see the number of policies assigned along with failover and relocate status.
9. Optional: Verify Rados block device (RBD) **volumereplication** and **volumereplicationgroup** on the primary cluster.

```
$ oc get volumereplications.replication.storage.openshift.io -A
```

Example output:

```
NAME          AGE  VOLUMEREPLICATIONCLASS          PVCNAME
DESIREDSTATE  CURRENTSTATE
busybox-pvc   2d16h  rbd-volumereplicationclass-1625360775  busybox-pvc   primary
Primary
```

```
$ oc get volumereplicationgroups.ramendr.openshift.io -A
```

Example output:

```
NAME          DESIREDSTATE  CURRENTSTATE
busybox-drpc  primary      Primary
```

10. Optional: Verify CephFS volsync replication source has been setup successfully in the primary cluster and VolSync ReplicationDestination has been setup in the failover cluster.

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

Example output:

NAME	SOURCE	LAST SYNC	DURATION	NEXT SYNC
busybox-pvc	busybox-pvc	2022-12-20T08:46:07Z	1m7.794661104s	2022-12-20T08:50:00Z

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

Example output:

NAME	LAST SYNC	DURATION	NEXT SYNC
busybox-pvc	2022-12-20T08:46:32Z	4m39.52261108s	

4.8.3. Deleting sample application

This section provides instructions for deleting the sample application **busybox** using the RHACM console.




IMPORTANT


When deleting a DR protected application, access to both clusters that belong to the DRPolicy is required. This is to ensure that all protected API resources and resources in the respective S3 stores are cleaned up as part of removing the DR protection. If access to one of the clusters is not healthy, deleting the **DRPlacementControl** resource for the application, on the hub, would remain in the Deleting state.

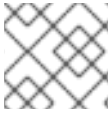
Prerequisites

- These 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, delete the **DRPlacementControl** if it is not auto-deleted after deleting the **busybox** application.
 - a. Log in to the OpenShift Web console for the Hub cluster and navigate to Installed Operators for the project **busybox-sample**.
For ApplicationSet applications, select the project as **openshift-gitops**.

- b. Click **OpenShift DR Hub Operator** and then click the **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**.



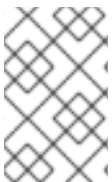
NOTE

This process can be used to delete any application with a **DRPlacementControl** resource.

4.9. SUBSCRIPTION-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

Failover is a process that transitions an application from a primary cluster to a secondary cluster in the event of a primary cluster failure. While failover provides the ability for the application to run on the secondary cluster with minimal interruption, making an uninformed failover decision can have adverse consequences, such as complete data loss in the event of unnoticed replication failure from primary to secondary cluster. If a significant amount of time has gone by since the last successful replication, it's best to wait until the failed primary is recovered.

[LastGroupSyncTime](#) is a [critical metric](#) that reflects the time since the last successful replication occurred for all PVCs associated with an application. In essence, it measures the synchronization health between the primary and secondary clusters. So, prior to initiating a failover from one cluster to another, check for this metric and only initiate the failover if the LastGroupSyncTime is within a reasonable time in the past.



NOTE

During the course of failover the Ceph-RBD mirror deployment on the failover cluster is scaled down to ensure a clean failover for volumes that are backed by Ceph-RBD as the storage provisioner.

Prerequisites

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update.
 1. Navigate to the **RHACM console** → **Infrastructure** → **Clusters** → **Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing failover operation.
However, failover operation can still be performed when the cluster you are failing over to is in a *Ready* state.
- Run the following command on the Hub Cluster to check if **lastGroupSyncTime** is within an acceptable data loss window, when compared to current time.

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```


Example output:

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

Procedure

1. On the Hub cluster, navigate to **Applications**.
2. Click the **Actions** menu at the end of application row to view the list of available actions.
3. Click **Failover application**.
4. After the **Failover application** modal is shown, select **policy** and **target cluster** to which the associated application will failover in case of a disaster.
5. Click the **Select subscription group** dropdown to verify the default selection or modify this setting.
By default, the subscription group that replicates for the application resources is selected.
6. Check the status of the **Failover readiness**.
 - If the status is **Ready** with a green tick, it indicates that the target cluster is ready for failover to start. Proceed to step 7.
 - If the status is **Unknown** or **Not ready**, then wait until the status changes to **Ready**.
7. Click **Initiate**. The busybox application is now failing over to the **Secondary-managed cluster**.
8. Close the modal window and track the status using the **Data policy** column on the Applications page.
9. Verify that the activity status shows as **FailedOver** for the application.
 - a. Navigate to the **Applications → Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, click the **View more details** link.
 - d. Verify that you can see one or more policy names and the ongoing activities (Last sync time and Activity status) associated with the policy in use with the application.

4.10. APPLICATIONSET-BASED APPLICATION FAILOVER BETWEEN MANAGED CLUSTERS

Failover is a process that transitions an application from a primary cluster to a secondary cluster in the event of a primary cluster failure. While failover provides the ability for the application to run on the secondary cluster with minimal interruption, making an uninformed failover decision can have adverse consequences, such as complete data loss in the event of unnoticed replication failure from primary to secondary cluster. If a significant amount of time has gone by since the last successful replication, it's best to wait until the failed primary is recovered.

[LastGroupSyncTime](#) is a [critical metric](#) that reflects the time since the last successful replication occurred for all PVCs associated with an application. In essence, it measures the synchronization health between the primary and secondary clusters. So, prior to initiating a failover from one cluster to another,

check for this metric and only initiate the failover if the `LastGroupSyncTime` is within a reasonable time in the past.



NOTE

During the course of failover the Ceph-RBD mirror deployment on the failover cluster is scaled down to ensure a clean failover for volumes that are backed by Ceph-RBD as the storage provisioner.

Prerequisites

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update.
 1. Navigate to the **RHACM console** → **Infrastructure** → **Clusters** → **Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing failover operation.
However, failover operation can still be performed when the cluster you are failing over to is in a *Ready* state.
- Run the following command on the Hub Cluster to check if **lastGroupSyncTime** is within an acceptable data loss window, when compared to current time.

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```

Example output:

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

Procedure

1. On the Hub cluster, navigate to **Applications**.
2. Click the **Actions** menu at the end of application row to view the list of available actions.
3. Click **Failover application**.
4. When the **Failover application** modal is shown, verify the details presented are correct and check the status of the Failover **readiness**. If the status is **Ready** with a green tick, it indicates that the target cluster is ready for failover to start.
5. Click **Initiate**. The busybox resources are now created on the target cluster.
6. Close the modal window and track the status using the **Data policy** column on the Applications page.
7. Verify that the activity status shows as **FailedOver** for the application.
 - a. Navigate to the **Applications** → **Overview** tab.

- b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
- c. On the **Data policy** popover, verify that you can see one or more policy names and the ongoing activities associated with the policy in use with the application.

4.11. RELOCATING SUBSCRIPTION-BASED APPLICATION BETWEEN MANAGED CLUSTERS

Relocate an application to its preferred location when all managed clusters are available.

Prerequisite

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update. Relocate can only be performed when both primary and preferred clusters are up and running.
 1. Navigate to **RHACM console** → **Infrastructure** → **Clusters** → **Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing relocate operation.
- Perform relocate when **lastGroupSyncTime** is within the replication interval (for example, 5 minutes) when compared to current time. This is recommended to minimize the Recovery Time Objective (RTO) for any single application.

Run this command on the Hub Cluster:

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```

Example output:

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

Compare the output time (UTC) to current time to validate that all **lastGroupSyncTime** values are within their application replication interval. If not, wait to Relocate until this is true for all **lastGroupSyncTime** values.

Procedure

1. On the Hub cluster, navigate to **Applications**.
2. Click the **Actions** menu at the end of application row to view the list of available actions.
3. Click **Relocate application**.
4. When the **Relocate application** modal is shown, select **policy** and **target cluster** to which the associated application will relocate to in case of a disaster.
5. By default, the subscription group that will deploy the application resources is selected. Click the **Select subscription group** dropdown to verify the default selection or modify this setting.
6. Check the status of the **Relocation readiness**.

- If the status is **Ready** with a green tick, it indicates that the target cluster is ready for relocation to start. Proceed to step 7.
 - If the status is **Unknown** or **Not ready**, then wait until the status changes to **Ready**.
7. Click **Initiate**. The busybox resources are now created on the target cluster.
 8. Close the modal window and track the status using the **Data policy** column on the Applications page.
 9. Verify that the activity status shows as **Relocated** for the application.
 - a. Navigate to the **Applications → Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, click the **View more details** link.
 - d. Verify that you can see one or more policy names and the ongoing activities (Last sync time and Activity status) associated with the policy in use with the application.

4.12. RELOCATING AN APPLICATIONSET-BASED APPLICATION BETWEEN MANAGED CLUSTERS

Relocate an application to its preferred location when all managed clusters are available.

Prerequisite

- If your setup has active and passive RHACM hub clusters, see [Hub recovery using Red Hat Advanced Cluster Management](#).
- When the primary cluster is in a state other than **Ready**, check the actual status of the cluster as it might take some time to update. Relocate can only be performed when both primary and preferred clusters are up and running.
 1. Navigate to **RHACM console → Infrastructure → Clusters → Cluster list** tab.
 2. Check the status of both the managed clusters individually before performing relocate operation.
- Perform relocate when **lastGroupSyncTime** is within the replication interval (for example, 5 minutes) when compared to current time. This is recommended to minimize the Recovery Time Objective (RTO) for any single application.
Run this command on the Hub Cluster:

```
$ oc get drpc -o yaml -A | grep lastGroupSyncTime
```

Example output:

```
[...]
lastGroupSyncTime: "2023-07-10T12:40:10Z"
```

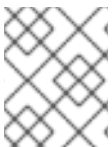
Compare the output time (UTC) to current time to validate that all **lastGroupSyncTime** values are within their application replication interval. If not, wait to Relocate until this is true for all **lastGroupSyncTime** values.

Procedure

1. On the Hub cluster, navigate to **Applications**.
2. Click the **Actions** menu at the end of application row to view the list of available actions.
3. Click **Relocate application**.
4. When the **Relocate application** modal is shown, select **policy** and **target cluster** to which the associated application will relocate to in case of a disaster.
5. Click **Initiate**. The busybox resources are now created on the target cluster.
6. Close the modal window and track the status using the **Data policy** column on the Applications page.
7. Verify that the activity status shows as **Relocated** for the application.
 - a. Navigate to the **Applications → Overview** tab.
 - b. In the **Data policy** column, click the **policy** link for the application you applied the policy to.
 - c. On the **Data policy** popover, verify that you can see one or more policy names and the relocation status associated with the policy in use with the application.

4.13. VIEWING RECOVERY POINT OBJECTIVE VALUES FOR DISASTER RECOVERY ENABLED APPLICATIONS

Recovery Point Objective (RPO) value is the most recent sync time of persistent data from the cluster where the application is currently active to its peer. This sync time helps determine duration of data lost during failover.



NOTE

This RPO value is applicable only for Regional-DR during failover. Relocation ensures there is no data loss during the operation, as all peer clusters are available.

You can view the Recovery Point Objective (RPO) value of all the protected volumes for their workload on the Hub cluster.

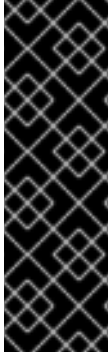
Procedure

1. On the Hub cluster, navigate to **Applications → Overview** tab.
2. In the **Data policy** column, click the **policy** link for the application you applied the policy to. A **Data Policies** modal page appears with the number of disaster recovery policies applied to each application along with failover and relocation status.
3. On the **Data Policies** modal page, click the **View more details** link. A detailed **Data Policies** modal page is displayed that shows the policy names and the ongoing activities (Last sync, Activity status) associated with the policy that is applied to the application.

The **Last sync time** reported in the modal page, represents the most recent sync time of all volumes that are DR protected for the application.

4.14. HUB RECOVERY USING RED HAT ADVANCED CLUSTER MANAGEMENT [TECHNOLOGY PREVIEW]

When your setup has active and passive Red Hat Advanced Cluster Management for Kubernetes (RHACM) hub clusters, and in case where the active hub is down, you can use the passive hub to failover or relocate the disaster recovery protected workloads.



IMPORTANT

Hub recovery is a Technology Preview feature and is subject to Technology Preview support limitations. 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.

For more information, see [Technology Preview Features Support Scope](#).

4.14.1. Configuring passive hub cluster

To perform hub recovery in case the active hub is down or unreachable, follow the procedure in this section to configure the passive hub cluster and then failover or relocate the disaster recovery protected workloads.

Procedure

1. Ensure that RHACM operator and **MultiClusterHub** is installed on the passive hub cluster. See [RHACM installation guide](#) for instructions.
After the operator is successfully installed, a popover with a message that the Web console update is available appears on the user interface. Click **Refresh web console** from this popover for the console changes to reflect.
2. Before hub recovery, configure backup and restore. See [Backup and restore](#) topics of *RHACM Business continuity* guide.
3. Install the multicluster orchestrator (MCO) operator along with Red Hat OpenShift GitOps operator on the passive RHACM hub prior to the restore. For instructions to restore your RHACM hub, see [Installing OpenShift Data Foundation Multicluster Orchestrator operator](#).
4. Ensure that **.spec.cleanupBeforeRestore** is set to **None** for the **Restore.cluster.openshift.io** resource. For details, see [Restoring passive resources while checking for backups](#) chapter of RHACM documentation.
5. If SSL access across clusters was configured manually during setup, then re-configure SSL access across clusters. For instructions, see [Configuring SSL access across clusters](#) chapter.
6. On the passive hub, add label to **openshift-operators** namespace to enable basic monitoring of **VolumeSynchronizationDelay** alert using this command. For alert details, see [Disaster recovery alerts](#) chapter.

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

4.14.2. Switching to passive hub cluster

Use this procedure when active hub is down or unreachable.

Procedure

1. Restore the backups on the passive hub cluster. For information, see [Restoring a hub cluster from backup](#).



IMPORTANT

Recovering a failed hub to its passive instance will only restore applications and their DR protected state to its last scheduled backup. Any application that was DR protected after the last scheduled backup would need to be protected again on the new hub.

2. Submariner is automatically installed once the managed clusters are imported on the passive hub.
3. Verify that the Primary and Secondary managed clusters are successfully imported into the RHACM console and they are accessible. If any of the managed clusters are down or unreachable then they will not be successfully imported.
4. Wait until DRPolicy validation succeeds.
5. Verify that the **DRPolicy** is created successfully. Run this command on the **Hub cluster** for each of the DRPolicy resources created, where `<drpolicy_name>` is replaced with a unique name.

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

Example output:

```
Succeeded
```

6. Refresh the RHACM console to make the DR monitoring dashboard tab accessible if it was enabled on the Active hub cluster.
7. If only the active hub cluster is down, restore the hub by performing hub recovery, and restoring the backups on the passive hub. If the managed clusters are still accessible, no further action is required.
8. If the primary managed cluster is down, along with the active hub cluster, you need to fail over the workloads from the primary managed cluster to the secondary managed cluster. For failover instructions, based on your workload type, see [Subscription-based applications](#) or [ApplicationSet-based applications](#).
9. Verify that the failover is successful. When the Primary managed cluster is down, then the PROGRESSION status for the workload would be in **Cleaning Up** phase until the down managed cluster is back online and successfully imported into the RHACM console. On the passive hub cluster, run the following command to check the PROGRESSION status.

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

Example output:

```
NAMESPACE          NAME          AGE  PREFERREDCLUSTER
```

FAILOVERCLUSTER	DESIREDSTATE	CURRENTSTATE	PROGRESSION	START TIME	DURATION	PEER READY
[...]						
busybox	cephfs-busybox-placement-1-drpc	103m	cluster-1			cluster-2
Failover	FailedOver	Cleaning Up	2024-04-15T09:12:23Z			False
busybox	cephfs-busybox-placement-1-drpc	102m	cluster-1			
Deployed	Completed	2024-04-15T07:40:09Z	37.200569819s			True
[...]						

CHAPTER 5. DISASTER RECOVERY WITH STRETCH CLUSTER FOR OPENSIFT DATA FOUNDATION

Red Hat OpenShift Data Foundation deployment can be stretched between two different geographical locations to provide the storage infrastructure with disaster recovery capabilities. When faced with a disaster, such as one of the two locations is partially or totally not available, OpenShift Data Foundation deployed on the OpenShift Container Platform deployment must be able to survive. This solution is available only for metropolitan spanned data centers with specific latency requirements between the servers of the infrastructure.



NOTE

The stretch cluster solution is designed for deployments where latencies do not exceed 10 ms maximum round-trip time (RTT) between the zones containing data volumes. For Arbiter nodes follow the latency requirements specified for etcd, see [Guidance for Red Hat OpenShift Container Platform Clusters - Deployments Spanning Multiple Sites\(Data Centers/Regions\)](#). Contact [Red Hat Customer Support](#) if you are planning to deploy with higher latencies.

The following diagram shows the simplest deployment for a stretched cluster:

OpenShift nodes and OpenShift Data Foundation daemons

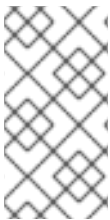


In the diagram the OpenShift Data Foundation monitor pod deployed in the Arbiter zone has a built-in tolerance for the master nodes. The diagram shows the master nodes in each Data Zone which are required for a highly available OpenShift Container Platform control plane. Also, it is important that the OpenShift Container Platform nodes in one of the zones have network connectivity with the OpenShift Container Platform nodes in the other two zones.

5.1. REQUIREMENTS FOR ENABLING STRETCH CLUSTER

- Ensure you have addressed OpenShift Container Platform requirements for deployments spanning multiple sites. For more information, see [knowledgebase article on cluster deployments spanning multiple sites](#).

- Ensure that you have at least three OpenShift Container Platform master nodes in three different zones. One master node in each of the three zones.
- Ensure that you have at least four OpenShift Container Platform worker nodes evenly distributed across the two Data Zones.
- For stretch clusters on bare metal, use the SSD drive as the root drive for OpenShift Container Platform master nodes.
- Ensure that each node is pre-labeled with its zone label. For more information, see the [Applying topology zone labels to OpenShift Container Platform node](#) section.
- The stretch cluster solution is designed for deployments where latencies do not exceed 10 ms between zones. Contact [Red Hat Customer Support](#) if you are planning to deploy with higher latencies.



NOTE

Flexible scaling and Arbiter both cannot be enabled at the same time as they have conflicting scaling logic. With Flexible scaling, you can add one node at a time to your OpenShift Data Foundation cluster. Whereas in an Arbiter cluster, you need to add at least one node in each of the two data zones.

5.2. APPLYING TOPOLOGY ZONE LABELS TO OPENSIFT CONTAINER PLATFORM NODES

During a site outage, the zone that has the arbiter function makes use of the arbiter label. These labels are arbitrary and must be unique for the three locations.

For example, you can label the nodes as follows:

```
topology.kubernetes.io/zone=arbiter for Master0
```

```
topology.kubernetes.io/zone=datacenter1 for Master1, Worker1, Worker2
```

```
topology.kubernetes.io/zone=datacenter2 for Master2, Worker3, Worker4
```

- To apply the labels to the node:

```
$ oc label node <NODENAME> topology.kubernetes.io/zone=<LABEL>
```

<NODENAME>

Is the name of the node

<LABEL>

Is the topology zone label

- To validate the labels using the example labels for the three zones:

```
$ oc get nodes -l topology.kubernetes.io/zone=<LABEL> -o name
```

<LABEL>

Is the topology zone label

Alternatively, you can run a single command to see all the nodes with its zone.

```
$ oc get nodes -L topology.kubernetes.io/zone
```

The stretch cluster topology zone labels are now applied to the appropriate OpenShift Container Platform nodes to define the three locations.

Next step

- [Install the local storage operator from the OpenShift Container Platform web console](#) .

5.3. INSTALLING LOCAL STORAGE OPERATOR

Install the Local Storage Operator from the Operator Hub before creating Red Hat OpenShift Data Foundation clusters on local storage devices.

Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators** → **OperatorHub**.
3. Type **local storage** in the **Filter by keyword** box to find the **Local Storage Operator** from the list of operators, and click on it.
4. Set the following options on the **Install Operator** page:
 - a. Update channel as **stable**.
 - b. Installation mode as **A specific namespace on the cluster**
 - c. Installed Namespace as **Operator recommended namespace openshift-local-storage**.
 - d. Update approval as **Automatic**.
5. Click **Install**.

Verification steps

- Verify that the Local Storage Operator shows a green tick indicating successful installation.

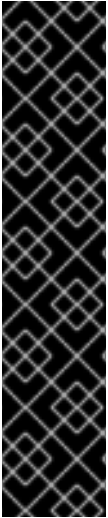
5.4. INSTALLING RED HAT OPENSIFT DATA FOUNDATION OPERATOR

You can install Red Hat OpenShift Data Foundation Operator using the Red Hat OpenShift Container Platform Operator Hub.

Prerequisites

- Access to an OpenShift Container Platform cluster using an account with cluster-admin and Operator installation permissions.
- You must have at least four worker nodes evenly distributed across two data centers in the Red Hat OpenShift Container Platform cluster.

- For additional resource requirements, see [Planning your deployment](#).



IMPORTANT

- When you need to override the cluster-wide default node selector for OpenShift Data Foundation, you can use the following command in command-line interface to specify a blank node selector for the **openshift-storage** namespace (create **openshift-storage** namespace in this case):

```
$ oc annotate namespace openshift-storage openshift.io/node-selector=
```

- Taint a node as **infra** to ensure only Red Hat OpenShift Data Foundation resources are scheduled on that node. This helps you save on subscription costs. For more information, see [How to use dedicated worker nodes for Red Hat OpenShift Data Foundation](#) chapter in the *Managing and Allocating Storage Resources* guide.

Procedure

1. Log in to the OpenShift Web Console.
2. Click **Operators** → **OperatorHub**.
3. Scroll or type **OpenShift Data Foundation** into the **Filter by keyword** box to search for the **OpenShift Data Foundation Operator**.
4. Click **Install**.
5. Set the following options on the **Install Operator** page:
 - a. Update Channel as **stable-4.15**.
 - b. Installation Mode as **A specific namespace on the cluster**
 - c. Installed Namespace as **Operator recommended namespace openshift-storage**. If Namespace **openshift-storage** does not exist, it is created during the operator installation.
 - d. Select **Approval Strategy** as **Automatic** or **Manual**.
If you select **Automatic** updates, then the Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your Operator without any intervention.

If you selected **Manual** updates, then the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to update the Operator to a newer version.
6. Ensure that the **Enable** option is selected for the **Console plugin**.
7. Click **Install**.

Verification steps

- After the operator is successfully installed, a pop-up with a message, **Web console update is available** appears on the user interface. Click **Refresh web console** from this pop-up for the console changes to reflect.
- In the Web Console:

- Navigate to Installed Operators and verify that the **OpenShift Data Foundation** Operator shows a green tick indicating successful installation.
- Navigate to **Storage** and verify if the **Data Foundation** dashboard is available.

Next steps

- [Create an OpenShift Data Foundation cluster](#).

5.5. CREATING OPENSIFT DATA FOUNDATION CLUSTER

Prerequisites

- Ensure that you have met all the requirements in [Requirements for enabling stretch cluster](#) section.

Procedure

1. In the OpenShift Web Console, click **Operators → Installed Operators** to view all the installed operators.
Ensure that the **Project** selected is **openshift-storage**.
2. Click on the **OpenShift Data Foundation** operator and then click **Create StorageSystem**.
3. In the Backing storage page, select the **Create a new StorageClass using the local storage devices** option.
4. Click **Next**.



IMPORTANT

You are prompted to install the Local Storage Operator if it is not already installed. Click **Install**, and follow the procedure as described in [Installing Local Storage Operator](#).

5. In the **Create local volume set** page, provide the following information:
 - a. Enter a name for the **LocalVolumeSet** and the **StorageClass**.
By default, the local volume set name appears for the storage class name. You can change the name.
 - b. Choose one of the following:
 - **Disks on all nodes**
Uses the available disks that match the selected filters on all the nodes.
 - **Disks on selected nodes**
Uses the available disks that match the selected filters only on selected nodes.



IMPORTANT

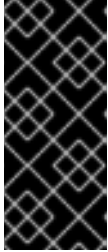
If the nodes selected do not match the OpenShift Data Foundation cluster requirement of an aggregated 30 CPUs and 72 GiB of RAM, a minimal cluster is deployed.

For minimum starting node requirements, see the [Resource requirements](#) section in the *Planning* guide.

- c. Select **SSD** or **NVMe** to build a supported configuration. You can select **HDDs** for unsupported test installations.
- d. Expand the **Advanced** section and set the following options:

Volume Mode	Block is selected by default.
Device Type	Select one or more device types from the dropdown list.
Disk Size	Set a minimum size of 100GB for the device and maximum available size of the device that needs to be included.
Maximum Disks Limit	This indicates the maximum number of PVs that can be created on a node. If this field is left empty, then PVs are created for all the available disks on the matching nodes.

- e. Click **Next**.
A pop-up to confirm the creation of LocalVolumeSet is displayed.
 - f. Click **Yes** to continue.
6. In the **Capacity and nodes** page, configure the following:
 - a. **Available raw capacity** is populated with the capacity value based on all the attached disks associated with the storage class. This takes some time to show up.
The **Selected nodes** list shows the nodes based on the storage class.
 - b. Select **Enable arbiter** checkbox if you want to use the stretch clusters. This option is available only when all the prerequisites for arbiter are fulfilled and the selected nodes are populated. For more information, see [Arbiter stretch cluster requirements in Requirements for enabling stretch cluster](#).
Select the **arbiter zone** from the dropdown list.
 7. Choose a performance profile for **Configure performance**.
You can also configure the performance profile after the deployment using the **Configure performance** option from the options menu of the **StorageSystems** tab.



IMPORTANT

Before selecting a resource profile, make sure to check the current availability of resources within the cluster. Opting for a higher resource profile in a cluster with insufficient resources might lead to installation failures. For more information about resource requirements, see [Resource requirement for performance profiles](#).

- a. Click **Next**.
8. Optional: In the Security and network page, configure the following based on your requirement:
 - a. To enable encryption, select **Enable data encryption for block and file storage**
 - b. Select one of the following **Encryption level**:
 - **Cluster-wide encryption** to encrypt the entire cluster (block and file).
 - **StorageClass encryption** to create encrypted persistent volume (block only) using encryption enabled storage class.
 - c. Optional: Select the **Connect to an external key management service** checkbox. This is optional for cluster-wide encryption.
 - i. From the **Key Management Service Provider** drop-down list, either select **Vault** or **Thales CipherTrust Manager (using KMIP)**. If you selected **Vault**, go to the next step. If you selected **Thales CipherTrust Manager (using KMIP)**, go to step iii.
 - ii. Select an **Authentication Method**.

Using Token authentication method

- Enter a unique **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), **Port** number and **Token**.
- Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:
 - Enter the Key Value secret path in the **Backend Path** that is dedicated and unique to OpenShift Data Foundation.
 - Optional: Enter **TLS Server Name** and **Vault Enterprise Namespace**
 - Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.
 - Click **Save** and skip to step iv.

Using Kubernetes authentication method

- Enter a unique Vault **Connection Name**, host **Address** of the Vault server ('https://<hostname or ip>'), **Port** number and **Role** name.
- Expand **Advanced Settings** to enter additional settings and certificate details based on your **Vault** configuration:
 - Enter the Key Value secret path in the **Backend Path** that is dedicated and unique to OpenShift Data Foundation.

- Optional: Enter **TLS Server Name** and **Authentication Path** if applicable.
 - Upload the respective PEM encoded certificate file to provide the **CA Certificate, Client Certificate** and **Client Private Key**.
 - Click **Save** and skip to step iv.
- iii. To use **Thales CipherTrust Manager (using KMIP)** as the KMS provider, follow the steps below:
- A. Enter a unique **Connection Name** for the Key Management service within the project.
 - B. In the **Address** and **Port** sections, enter the IP of Thales CipherTrust Manager and the port where the KMIP interface is enabled. For example:
 - **Address:** 123.34.3.2
 - **Port:** 5696
 - C. Upload the **Client Certificate, CA certificate**, and **Client Private Key**.
 - D. If StorageClass encryption is enabled, enter the Unique Identifier to be used for encryption and decryption generated above.
 - E. The **TLS Server** field is optional and used when there is no DNS entry for the KMIP endpoint. For example, **kmip_all_<port>.ciphertrustmanager.local**.
- d. **Network** is set to Default (OVN) if you are using a single network. You can switch to Custom (Multus) if you are using multiple network interfaces and then choose any one of the following:
- i. Select a Public Network Interface from the dropdown.
 - ii. Select a Cluster Network Interface from the dropdown.



NOTE

If you are using only one additional network interface, select the single **NetworkAttachmentDefinition**, that is, **ocs-public-cluster** for the Public Network Interface, and leave the Cluster Network Interface blank.

- a. Click **Next**.
9. In the **Data Protection** page, click **Next**.
10. In the **Review and create** page, review the configuration details. To modify any configuration settings, click **Back** to go back to the previous configuration page.
11. Click **Create StorageSystem**.

Verification steps

- To verify the final Status of the installed storage cluster:
 - a. In the OpenShift Web Console, navigate to **Installed Operators → OpenShift Data**

Foundation → **Storage System** → **ocs-storagecluster-storagesystem** → **Resources**.

- b. Verify that the **Status** of **StorageCluster** is **Ready** and has a green tick mark next to it.
- For arbiter mode of deployment:
 - a. In the OpenShift Web Console, navigate to **Installed Operators** → **OpenShift Data Foundation** → **Storage System** → **ocs-storagecluster-storagesystem** → **Resources** → **ocs-storagecluster**.
 - b. In the YAML tab, search for the **arbiter** key in the **spec** section and ensure **enable** is set to **true**.

```
spec:
  arbiter:
    enable: true
  [..]
  nodeTopologies:
    arbiterLocation: arbiter #arbiter zone
  storageDeviceSets:
  - config: {}
    count: 1
    [..]
    replica: 4
  status:
    conditions:
    [..]
    failureDomain: zone
```

- To verify that all the components for OpenShift Data Foundation are successfully installed, see [Verifying your OpenShift Data Foundation installation](#).

5.6. VERIFYING OPENSIFT DATA FOUNDATION DEPLOYMENT

To verify that OpenShift Data Foundation is deployed correctly:

- [Verify the state of the pods](#).
- [Verify that the OpenShift Data Foundation cluster is healthy](#).
- [Verify that the Multicloud Object Gateway is healthy](#).
- [Verify that the OpenShift Data Foundation specific storage classes exist](#).

5.6.1. Verifying the state of the pods

Procedure

1. Click **Workloads** → **Pods** from the OpenShift Web Console.
2. Select **openshift-storage** from the **Project** drop-down list.

**NOTE**

If the **Show default projects** option is disabled, use the toggle button to list all the default projects.

For more information about the expected number of pods for each component and how it varies depending on the number of nodes, see [Table 5.1, “Pods corresponding to OpenShift Data Foundation cluster”](#).

3. Click the **Running** and **Completed** tabs to verify that the following pods are in **Running** and **Completed** state:

Table 5.1. Pods corresponding to OpenShift Data Foundation cluster

Component	Corresponding pods
OpenShift Data Foundation Operator	<ul style="list-style-type: none"> ● ocs-operator-* (1 pod on any worker node) ● ocs-metrics-exporter-* (1 pod on any worker node) ● odf-operator-controller-manager-* (1 pod on any worker node) ● odf-console-* (1 pod on any worker node) ● csi-addons-controller-manager-* (1 pod on any worker node)
Rook-ceph Operator	<p>rook-ceph-operator-*</p> <p>(1 pod on any worker node)</p>
Multicloud Object Gateway	<ul style="list-style-type: none"> ● noobaa-operator-* (1 pod on any worker node) ● noobaa-core-* (1 pod on any storage node) ● noobaa-db-pg-* (1 pod on any storage node) ● noobaa-endpoint-* (1 pod on any storage node)
MON	<p>rook-ceph-mon-*</p> <p>(5 pods are distributed across 3 zones, 2 per data-center zones and 1 in arbiter zone)</p>
MGR	<p>rook-ceph-mgr-*</p> <p>(2 pods on any storage node)</p>

Component	Corresponding pods
MDS	<p>rook-ceph-mds-ocs-storagecluster-cephfilesystem-*</p> <p>(2 pods are distributed across 2 data-center zones)</p>
RGW	<p>rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*</p> <p>(2 pods are distributed across 2 data-center zones)</p>
CSI	<ul style="list-style-type: none"> ● cephfs <ul style="list-style-type: none"> ○ csi-cephfsplugin-* (1 pod on each worker node) ○ csi-cephfsplugin-provisioner-* (2 pods distributed across worker nodes) ● rbd <ul style="list-style-type: none"> ○ csi-rbdplugin-* (1 pod on each worker node) ○ csi-rbdplugin-provisioner-* (2 pods distributed across worker nodes)
rook-ceph-crashcollector	<p>rook-ceph-crashcollector-*</p> <p>(1 pod on each storage node and 1 pod in arbiter zone)</p>
OSD	<ul style="list-style-type: none"> ● rook-ceph-osd-* (1 pod for each device) ● rook-ceph-osd-prepare-ocs-deviceset-* (1 pod for each device)

5.6.2. Verifying the OpenShift Data Foundation cluster is healthy

Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
3. In the **Status** card of the **Block and File** tab, verify that the *Storage Cluster* has a green tick.
4. In the **Details** card, verify that the cluster information is displayed.

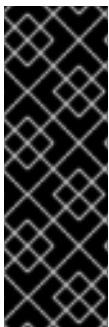
For more information on the health of the OpenShift Data Foundation cluster using the **Block and File** dashboard, see [Monitoring OpenShift Data Foundation](#).

5.6.3. Verifying the Multicloud Object Gateway is healthy

Procedure

1. In the OpenShift Web Console, click **Storage** → **Data Foundation**.
2. In the **Status** card of the **Overview** tab, click **Storage System** and then click the storage system link from the pop up that appears.
 - a. In the **Status card** of the **Object** tab, verify that both *Object Service* and *Data Resiliency* have a green tick.
 - b. In the **Details** card, verify that the MCG information is displayed.

For more information on the health of the OpenShift Data Foundation cluster using the object service dashboard, see [Monitoring OpenShift Data Foundation](#).



IMPORTANT

The Multicloud Object Gateway only has a single copy of the database (NooBaa DB). This means if NooBaa DB PVC gets corrupted and we are unable to recover it, can result in total data loss of applicative data residing on the Multicloud Object Gateway. Because of this, Red Hat recommends taking a backup of NooBaa DB PVC regularly. If NooBaa DB fails and cannot be recovered, then you can revert to the latest backed-up version. For instructions on backing up your NooBaa DB, follow the steps in [this knowledgebase article](#).

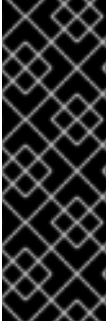
5.6.4. Verifying that the specific storage classes exist

Procedure

1. Click **Storage** → **Storage Classes** from the left pane of the OpenShift Web Console.
2. Verify that the following storage classes are created with the OpenShift Data Foundation cluster creation:
 - **ocs-storagecluster-ceph-rbd**
 - **ocs-storagecluster-cephfs**
 - **openshift-storage.noobaa.io**
 - **ocs-storagecluster-ceph-rgw**

5.7. INSTALL ZONE AWARE SAMPLE APPLICATION

Deploy a zone aware sample application to validate whether an OpenShift Data Foundation, stretch cluster setup is configured correctly.



IMPORTANT

With latency between the data zones, you can expect to see performance degradation compared to an OpenShift cluster with low latency between nodes and zones (for example, all nodes in the same location). The rate of or amount of performance degradation depends on the latency between the zones and on the application behavior using the storage (such as heavy write traffic). Ensure that you test the critical applications with stretch cluster configuration to ensure sufficient application performance for the required service levels.

A ReadWriteMany (RWX) Persistent Volume Claim (PVC) is created using the **ocs-storagecluster-cephfs** storage class. Multiple pods use the newly created RWX PVC at the same time. The application used is called File Uploader.

Demonstration on how an application is spread across topology zones so that it is still available in the event of a site outage:



NOTE

This demonstration is possible since this application shares the same RWX volume for storing files. It works for persistent data access as well because Red Hat OpenShift Data Foundation is configured as a stretched cluster with zone awareness and high availability.

1. Create a new project.

```
$ oc new-project my-shared-storage
```

2. Deploy the example PHP application called file-uploader.

```
$ oc new-app openshift/php:7.3-ubi8~https://github.com/christianh814/openshift-php-upload-demo --name=file-uploader
```

Example Output:

```
Found image 4f2dcc0 (9 days old) in image stream "openshift/php" under tag "7.2-ubi8" for "openshift/php:7.2-ubi8"
```

```
Apache 2.4 with PHP 7.2
```

```
-----
PHP 7.2 available as container is a base platform for building and running various PHP 7.2 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts to make it easy for developers to write dynamically generated web pages. PHP also offers built-in database integration for several commercial and non-commercial database management systems, so writing a database-enabled webpage with PHP is fairly simple. The most common use of PHP coding is probably as a replacement for CGI scripts.
```

```
Tags: builder, php, php72, php-72
```

```
* A source build using source code from https://github.com/christianh814/openshift-php-upload-demo will be created
```

```
eated
```

```
* The resulting image will be pushed to image stream tag "file-uploader:latest"
```

* Use 'oc start-build' to trigger a new build

--> Creating resources ...

imagestream.image.openshift.io "file-uploader" created

buildconfig.build.openshift.io "file-uploader" created

deployment.apps "file-uploader" created

service "file-uploader" created

--> Success

Build scheduled, use 'oc logs -f buildconfig/file-uploader' to track its progress.

Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:

'oc expose service/file-uploader'

Run 'oc status' to view your app.

3. View the build log and wait until the application is deployed.

```
$ oc logs -f bc/file-uploader -n my-shared-storage
```

Example Output:

```
Cloning "https://github.com/christianh814/openshift-php-upload-demo" ...
```

```
[...]
```

```
Generating dockerfile with builder image image-registry.openshift-image-registry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea
```

```
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea
```

```
STEP 2: LABEL "io.openshift.build.commit.author"="Christian Hernandez <christian.hernandez@yahoo.com>" "io.openshift.build.commit.date"="Sun Oct 1 17:15:09 2017 -0700" "io.openshift.build.commit.id"="288eda3dff43b02f7f7b6b6b6f93396ffdf34cb2" "io.openshift.build.commit.ref"="master" "
```

```
io.openshift.build.commit.message"="trying to modularize" "io.openshift.build.source-location"="https://github.com/christianh814/openshift-php-upload-demo" "io.openshift.build.image"="image-registry.openshift-image-registry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea"
```

```
STEP 3: ENV OPENSIFT_BUILD_NAME="file-uploader-1"
```

```
OPENSIFT_BUILD_NAMESP
```

```
ACE="my-shared-storage" OPENSIFT_BUILD_SOURCE="https://github.com/christianh814/openshift-php-upload-demo" OPENSIFT_BUILD_COMMIT="288eda3dff43b02f7f7b6b6b6f93396ffdf34cb2"
```

```
STEP 4: USER root
```

```
STEP 5: COPY upload/src /tmp/src
```

```
STEP 6: RUN chown -R 1001:0 /tmp/src
```

```
STEP 7: USER 1001
```

```
STEP 8: RUN /usr/libexec/s2i/assemble
```

```
---> Installing application source...
```

```
=> sourcing 20-copy-config.sh ...
```

```
---> 17:24:39 Processing additional arbitrary httpd configuration provided by s2i ...
```

```
=> sourcing 00-documentroot.conf ...
```

```
=> sourcing 50-mpm-tuning.conf ...
```

```

=> sourcing 40-ssl-certs.sh ...
STEP 9: CMD /usr/libexec/s2i/run
STEP 10: COMMIT temp.builder.openshift.io/my-shared-storage/file-uploader-1:3
b83e447
Getting image source signatures

[...]

```

The command prompt returns out of the tail mode after you see **Push successful**.



NOTE

The `new-app` command deploys the application directly from the git repository and does not use the OpenShift template, hence the OpenShift route resource is not created by default. You need to create the route manually.

Scaling the application

1. Scale the application to four replicas and expose its services to make the application zone aware and available.

```
$ oc expose svc/file-uploader -n my-shared-storage
```

```
$ oc scale --replicas=4 deploy/file-uploader -n my-shared-storage
```

```
$ oc get pods -o wide -n my-shared-storage
```

You should have four `file-uploader` pods in a few minutes. Repeat the above command until there are 4 `file-uploader` pods in the **Running** status.

2. Create a PVC and attach it into an application.

```

$ oc set volume deploy/file-uploader --add --name=my-shared-storage \
-t pvc --claim-mode=ReadWriteMany --claim-size=10Gi \
--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \
--mount-path=/opt/app-root/src/uploaded \
-n my-shared-storage

```

This command:

- Creates a PVC.
 - Updates the application deployment to include a volume definition.
 - Updates the application deployment to attach a volume mount into the specified mount-path.
 - Creates a new deployment with the four application pods.
3. Check the result of adding the volume.

```
$ oc get pvc -n my-shared-storage
```

Example Output:

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
STORAGECLASS		AGE		
my-shared-storage	Bound	pvc-5402cc8a-e874-4d7e-af76-1eb05bd2e7c7	10Gi	RWX
ocs-storagecluster-cephfs		52s		

Notice the **ACCESS MODE** is set to RWX.

All the four **file-uploader** pods are using the same RWX volume. Without this access mode, OpenShift does not attempt to attach multiple pods to the same Persistent Volume (PV) reliably. If you attempt to scale up the deployments that are using ReadWriteOnce (RWO) PV, the pods may get colocated on the same node.

5.7.1. Scaling the application after installation

Procedure

1. Scale the application to four replicas and expose its services to make the application zone aware and available.

```
$ oc expose svc/file-uploader -n my-shared-storage
```

```
$ oc scale --replicas=4 deploy/file-uploader -n my-shared-storage
```

```
$ oc get pods -o wide -n my-shared-storage
```

You should have four file-uploader pods in a few minutes. Repeat the above command until there are 4 file-uploader pods in the **Running** status.

2. Create a PVC and attach it into an application.

```
$ oc set volume deploy/file-uploader --add --name=my-shared-storage \
-t pvc --claim-mode=ReadWriteMany --claim-size=10Gi \
--claim-name=my-shared-storage --claim-class=ocs-storagecluster-cephfs \
--mount-path=/opt/app-root/src/uploaded \
-n my-shared-storage
```

This command:

- Creates a PVC.
 - Updates the application deployment to include a volume definition.
 - Updates the application deployment to attach a volume mount into the specified mount-path.
 - Creates a new deployment with the four application pods.
3. Check the result of adding the volume.

```
$ oc get pvc -n my-shared-storage
```

Example Output:

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES
STORAGECLASS		AGE		
my-shared-storage	Bound	pvc-5402cc8a-e874-4d7e-af76-1eb05bd2e7c7	10Gi	RWX
ocs-storagecluster-cephfs		52s		

Notice the **ACCESS MODE** is set to RWX.

All the four **file-uploader** pods are using the same RWX volume. Without this access mode, OpenShift does not attempt to attach multiple pods to the same Persistent Volume (PV) reliably. If you attempt to scale up the deployments that are using ReadWriteOnce (RWO) PV, the pods may get colocated on the same node.

5.7.2. Modify Deployment to be Zone Aware

Currently, the **file-uploader** Deployment is not zone aware and can schedule all the pods in the same zone. In this case, if there is a site outage then the application is unavailable. For more information, see [Controlling pod placement by using pod topology spread constraints](#) .

1. Add the pod placement rule in the application deployment configuration to make the application zone aware.
 - a. Run the following command, and review the output:

```
$ oc get deployment file-uploader -o yaml -n my-shared-storage | less
```

Example Output:

```
[...]
spec:
  progressDeadlineSeconds: 600
  replicas: 4
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      deployment: file-uploader
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:
        deployment: file-uploader
    spec: # <-- Start inserted lines after here
      containers: # <-- End inserted lines before here
      - image: image-registry.openshift-image-registry.svc:5000/my-shared-storage/file-
        uploader@sha256:a458ea62f990e431ad7d5f84c89e2fa27bdebdd5e29c5418c70c56eb81f
        0a26b
        imagePullPolicy: IfNotPresent
        name: file-uploader
[...]
```

- b. Edit the deployment to use the topology zone labels.

```
$ oc edit deployment file-uploader -n my-shared-storage
```

Add the following new lines between the **Start** and **End** (shown in the output in the previous step):

```
[...]
spec:
  topologySpreadConstraints:
    - labelSelector:
        matchLabels:
          deployment: file-uploader
      maxSkew: 1
      topologyKey: topology.kubernetes.io/zone
      whenUnsatisfiable: DoNotSchedule
    - labelSelector:
        matchLabels:
          deployment: file-uploader
      maxSkew: 1
      topologyKey: kubernetes.io/hostname
      whenUnsatisfiable: ScheduleAnyway
  nodeSelector:
    node-role.kubernetes.io/worker: ""
  containers:
[...]
```

Example output:

```
deployment.apps/file-uploader edited
```

2. Scale down the deployment to **zero** pods and then back to **four** pods. This is needed because the deployment changed in terms of pod placement.

Scaling down to zero pods

```
$ oc scale deployment file-uploader --replicas=0 -n my-shared-storage
```

Example output:

```
deployment.apps/file-uploader scaled
```

Scaling up to four pods

```
$ oc scale deployment file-uploader --replicas=4 -n my-shared-storage
```

Example output:

```
deployment.apps/file-uploader scaled
```

3. Verify that the four pods are spread across the four nodes in datacenter1 and datacenter2 zones.

-

```
$ oc get pods -o wide -n my-shared-storage | egrep '^file-uploader'| grep -v build | awk '{print $7}' | sort | uniq -c
```

Example output:

```
1 perf1-mz8bt-worker-d2hdm
1 perf1-mz8bt-worker-k68rv
1 perf1-mz8bt-worker-ntkp8
1 perf1-mz8bt-worker-qpwsr
```

Search for the zone labels used.

```
$ oc get nodes -L topology.kubernetes.io/zone | grep datacenter | grep -v master
```

Example output:

```
perf1-mz8bt-worker-d2hdm Ready worker 35d v1.20.0+5bfd19 datacenter1
perf1-mz8bt-worker-k68rv Ready worker 35d v1.20.0+5bfd19 datacenter1
perf1-mz8bt-worker-ntkp8 Ready worker 35d v1.20.0+5bfd19 datacenter2
perf1-mz8bt-worker-qpwsr Ready worker 35d v1.20.0+5bfd19 datacenter2
```

4. Use the file-uploader web application using your browser to upload new files.
 - a. Find the route that is created.

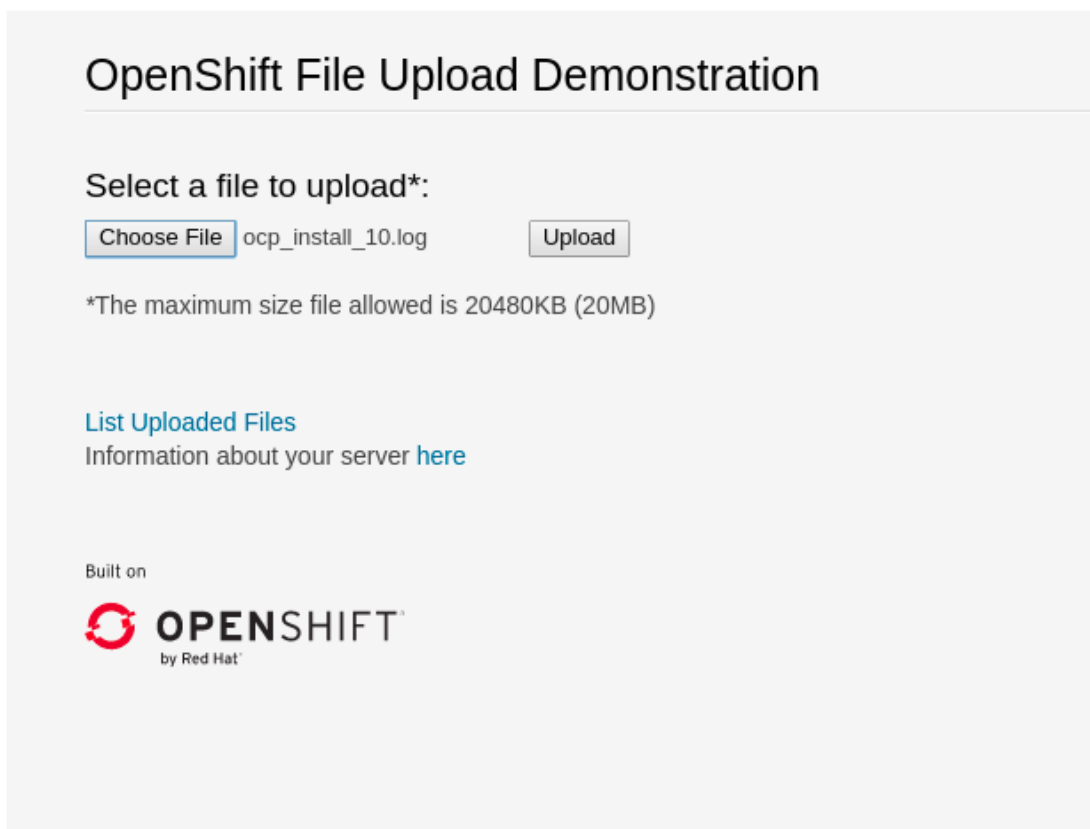
```
$ oc get route file-uploader -n my-shared-storage -o jsonpath --
template="http://{.spec.host}{\n}"
```

Example Output:

```
http://file-uploader-my-shared-storage.apps.cluster-ocs4-abdf.ocs4-
abdf.sandbox744.opentlc.com
```

- b. Point your browser to the web application using the route in the previous step. The web application lists all the uploaded files and offers the ability to upload new ones as well as you download the existing data. Right now, there is nothing.
 - c. Select an arbitrary file from your local machine and upload it to the application.
 - i. Click **Choose file** to select an arbitrary file.
 - ii. Click **Upload**.

Figure 5.1. A simple PHP-based file upload tool



- d. Click **List uploaded files** to see the list of all currently uploaded files.



NOTE

The OpenShift Container Platform image registry, ingress routing, and monitoring services are not zone aware.

5.8. RECOVERING OPENSIFT DATA FOUNDATION STRETCH CLUSTER

Given that the stretch cluster disaster recovery solution is to provide resiliency in the face of a complete or partial site outage, it is important to understand the different methods of recovery for applications and their storage.

How the application is architected determines how soon it becomes available again on the active zone.

There are different methods of recovery for applications and their storage depending on the site outage. The recovery time depends on the application architecture. The different methods of recovery are as follows:

- [Recovering zone-aware HA applications with RWX storage](#) .
- [Recovering HA applications with RWX storage](#) .
- [Recovering applications with RWO storage](#) .
- [Recovering StatefulSet pods](#) .

5.8.1. Understanding zone failure

For the purpose of this section, zone failure is considered as a failure where all OpenShift Container Platform, master and worker nodes in a zone are no longer communicating with the resources in the second data zone (for example, powered down nodes). If communication between the data zones is still partially working (intermittently up or down), the cluster, storage, and network admins should disconnect the communication path between the data zones for recovery to succeed.



IMPORTANT

When you install the sample application, power off the OpenShift Container Platform nodes (at least the nodes with OpenShift Data Foundation devices) to test the failure of a data zone in order to validate that your file-uploader application is available, and you can upload new files.

5.8.2. Recovering zone-aware HA applications with RWX storage

Applications that are deployed with **topologyKey: topology.kubernetes.io/zone** have one or more replicas scheduled in each data zone, and are using shared storage, that is, ReadWriteMany (RWX) CephFS volume, terminate themselves in the failed zone after few minutes and new pods are rolled in and stuck in pending state until the zones are recovered.

An example of this type of application is detailed in the [Install Zone Aware Sample Application](#) section.



IMPORTANT

During zone recovery if application pods go into CrashLoopBackOff (CLBO) state with permission denied error while mounting the CephFS volume, then restart the nodes where the pods are scheduled. Wait for some time and then check if the pods are running again.

5.8.3. Recovering HA applications with RWX storage

Applications that are using **topologyKey: kubernetes.io/hostname** or no topology configuration have no protection against all of the application replicas being in the same zone.



NOTE

This can happen even with *podAntiAffinity* and **topologyKey: kubernetes.io/hostname** in the **Pod** spec because this anti-affinity rule is host-based and not zone-based.

If this happens and all replicas are located in the zone that fails, the application using ReadWriteMany (RWX) storage takes 6–8 minutes to recover on the active zone. This pause is for the OpenShift Container Platform nodes in the failed zone to become **NotReady** (60 seconds) and then for the default pod eviction timeout to expire (300 seconds).

5.8.4. Recovering applications with RWO storage

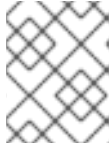
Applications that use ReadWriteOnce (RWO) storage have a known behavior described in this [Kubernetes issue](#). Because of this issue, if there is a data zone failure, any application pods in that zone mounting RWO volumes (for example, **cephrbd** based volumes) are stuck with **Terminating** status after 6–8 minutes and are not re-created on the active zone without manual intervention.

Check the OpenShift Container Platform nodes with a status of **NotReady**. There may be an issue that prevents the nodes from communicating with the OpenShift control plane. However, the nodes may still be performing I/O operations against Persistent Volumes (PVs).

If two pods are concurrently writing to the same RWO volume, there is a risk of data corruption. Ensure that processes on the **NotReady** node are either terminated or blocked until they are terminated.

Example solutions:

- Use an out of band management system to power off a node, with confirmation, to ensure process termination.
- Withdraw a network route that is used by nodes at a failed site to communicate with storage.



NOTE

Before restoring service to the failed zone or nodes, confirm that all the pods with PVs have terminated successfully.

To get the **Terminating** pods to recreate on the active zone, you can either force delete the pod or delete the finalizer on the associated PV. Once one of these two actions are completed, the application pod should recreate on the active zone and successfully mount its RWO storage.

Force deleting the pod

Force deletions do not wait for confirmation from the kubelet that the pod has been terminated.

```
$ oc delete pod <PODNAME> --grace-period=0 --force --namespace <NAMESPACE>
```

<PODNAME>

Is the name of the pod

<NAMESPACE>

Is the project namespace

Deleting the finalizer on the associated PV

Find the associated PV for the Persistent Volume Claim (PVC) that is mounted by the Terminating pod and delete the finalizer using the **oc patch** command.

```
$ oc patch -n openshift-storage pv/<PV_NAME> -p '{"metadata":{"finalizers":[]}}' --type=merge
```

<PV_NAME>

Is the name of the PV

An easy way to find the associated PV is to describe the Terminating pod. If you see a multi-attach warning, it should have the PV names in the warning (for example, **pvc-0595a8d2-683f-443b-ae0-6e547f5f5a7c**).

```
$ oc describe pod <PODNAME> --namespace <NAMESPACE>
```

<PODNAME>

Is the name of the pod

<NAMESPACE>

Is the project namespace

Example output:

```
[...]
Events:
```

Type	Reason	Age	From	Message
Normal	Scheduled	4m5s	default-scheduler	Successfully assigned openshift-storage/noobaa-db-pg-0 to perf1-mz8bt-worker-d2hdm
Warning	FailedAttachVolume	4m5s	attachdetach-controller	Multi-Attach error for volume "pvc-0595a8d2-683f-443b-ae0-6e547f5f5a7c" Volume is already exclusively attached to one node and can't be attached to another

5.8.5. Recovering StatefulSet pods

Pods that are part of a StatefulSet have a similar issue as pods mounting ReadWriteOnce (RWO) volumes. More information is referenced in the Kubernetes resource [StatefulSet considerations](#).

To get the pods part of a StatefulSet to re-create on the active zone after 6–8 minutes you need to force delete the pod with the same requirements (that is, OpenShift Container Platform node powered off or communication disconnected) as pods with RWO volumes.

CHAPTER 6. MONITORING DISASTER RECOVERY HEALTH

6.1. ENABLE MONITORING FOR DISASTER RECOVERY

Use this procedure to enable basic monitoring for your disaster recovery setup.

Procedure

1. On the Hub cluster, open a terminal window
2. Add the following label to **openshift-operator** namespace.

```
$ oc label namespace openshift-operators openshift.io/cluster-monitoring='true'
```

6.2. ENABLING DISASTER RECOVERY DASHBOARD ON HUB CLUSTER

This section guides you to enable the disaster recovery dashboard for advanced monitoring on the Hub cluster.

For Regional-DR, the dashboard shows monitoring status cards for operator health, cluster health, metrics, alerts and application count.

For Metro-DR, you can configure the dashboard to only monitor the ramen setup health and application count.

Prerequisites

- Ensure that you have already installed the following
 - OpenShift Container Platform version 4.15 and have administrator privileges.
 - ODF Multicluster Orchestrator with the console plugin enabled.
 - Red Hat Advanced Cluster Management for Kubernetes 2.10 (RHACM) from Operator Hub. For instructions on how to install, see [Installing RHACM](#).
- Ensure you have enabled observability on RHACM. See [Enabling observability guidelines](#).

Procedure

1. On the Hub cluster, open a terminal window and perform the next steps.
2. Create the configmap file named **observability-metrics-custom-allowlist.yaml**. You can use the following YAML to list the disaster recovery metrics on Hub cluster. For details, see [Adding custom metrics](#). To know more about ramen metrics, see [Disaster recovery metrics](#).

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: observability-metrics-custom-allowlist
  namespace: open-cluster-management-observability
data:
  metrics_list.yaml: |
    names:
```



```

- ceph_rbd_mirror_snapshot_sync_bytes
- ceph_rbd_mirror_snapshot_snapshots
matches:
- __name__="csv_succeeded",exported_namespace="openshift-dr-system",name=~"odr-
cluster-operator.*"
- __name__="csv_succeeded",exported_namespace="openshift-
operators",name=~"volsync.*"

```

3. In the **open-cluster-management-observability** namespace, run the following command:

```
$ oc apply -n open-cluster-management-observability -f observability-metrics-custom-allowlist.yaml
```

4. After **observability-metrics-custom-allowlist** yaml is created, RHACM starts collecting the listed OpenShift Data Foundation metrics from all the managed clusters.
To exclude a specific managed cluster from collecting the observability data, add the following cluster label to the **clusters: observability: disabled**.

6.3. VIEWING HEALTH STATUS OF DISASTER RECOVERY REPLICATION RELATIONSHIPS

Prerequisites

Ensure that you have enabled the disaster recovery dashboard for monitoring. For instructions, see chapter [Enabling disaster recovery dashboard on Hub cluster](#).

Procedure

1. On the Hub cluster, ensure **All Clusters** option is selected.
2. Refresh the console to make the DR monitoring dashboard tab accessible.
3. Navigate to **Data Services** and click **Data policies**.
4. On the Overview tab, you can view the health status of the operators, clusters and applications. Green tick indicates that the operators are running and available..
5. Click the **Disaster recovery** tab to view a list of DR policy details and connected applications.

6.4. DISASTER RECOVERY METRICS

These are the ramen metrics that are scrapped by prometheus.

- ramen_last_sync_timestamp_seconds
- ramen_policy_schedule_interval_seconds
- ramen_last_sync_duration_seconds
- ramen_last_sync_data_bytes

Run these metrics from the Hub cluster where Red Hat Advanced Cluster Management for Kubernetes (RHACM operator) is installed.

Last synchronization timestamp in seconds

This is the time in seconds which gives the time of the most recent successful synchronization of all PVCs per application.

Metric name

ramen_last_sync_timestamp_seconds

Metrics type

Gauge

Labels

- **ObjType:** Type of the object, here its DPPC
- **ObjName:** Name of the object, here it is DRPC-Name
- **ObjNamespace:** DRPC namespace
- **Policyname:** Name of the DRPolicy
- **SchedulingInterval:** Scheduling interval value from DRPolicy

Metric value

Value is set as Unix seconds which is obtained from **lastGroupSyncTime** from DRPC status.

Policy schedule interval in seconds

This gives the scheduling interval in seconds from DRPolicy.

Metric name

ramen_policy_schedule_interval_seconds

Metrics type

Gauge

Labels

- **Policyname:** Name of the DRPolicy

Metric value

This is set to a scheduling interval in seconds which is taken from DRPolicy.

Last synchronization duration in seconds

This represents the longest time taken to sync from the most recent successful synchronization of all PVCs per application.

Metric name

ramen_last_sync_duration_seconds

Metrics type

Gauge

Labels

- **obj_type:** Type of the object, here it is DPPC
- **obj_name:** Name of the object, here it is DRPC-Name
- **obj_namespace:** DRPC namespace

- **scheduling_interval**: Scheduling interval value from DRPolicy

Metric value

The value is taken from **lastGroupSyncDuration** from DRPC status.

Total bytes transferred from most recent synchronization

This value represents the total bytes transferred from the most recent successful synchronization of all PVCs per application.

Metric name

ramen_last_sync_data_bytes

Metrics type

Gauge

Labels

- **obj_type**: Type of the object, here it is DPPC
- **obj_name**: Name of the object, here it is DRPC-Name
- **obj_namespace**: DRPC namespace
- **scheduling_interval**: Scheduling interval value from DRPolicy

Metric value

The value is taken from **lastGroupSyncBytes** from DRPC status.

6.5. DISASTER RECOVERY ALERTS

This section provides a list of all supported alerts associated with Red Hat OpenShift Data Foundation within a disaster recovery environment.

Recording rules

- Record: **ramen_sync_duration_seconds**

Expression

```
sum by (obj_name, obj_namespace, obj_type, job, policyname)(time() -
(ramen_last_sync_timestamp_seconds > 0))
```

Purpose

The time interval between the volume group's last sync time and the time now in seconds.

- Record: **ramen_rpo_difference**

Expression

```
ramen_sync_duration_seconds{job="ramen-hub-operator-metrics-service"} /
on(policyname, job) group_left() (ramen_policy_schedule_interval_seconds{job="ramen-
hub-operator-metrics-service"})
```

Purpose

The difference between the expected sync delay and the actual sync delay taken by the volume replication group.

- Record: **count_persistentvolumeclaim_total**

Expression

```
count(kube_persistentvolumeclaim_info)
```

Purpose

Sum of all PVC from the managed cluster.

Alerts

- Alert: **VolumeSynchronizationDelay**

Impact

Critical

Purpose

Actual sync delay taken by the volume replication group is thrice the expected sync delay.

YAML

```
alert: VolumeSynchronizationDela
expr: ramen_rpo_difference >= 3
for: 5s
labels:
  cluster: '{{ $labels.cluster }}'
  severity: critical
annotations:
  description: >-
    Syncing of volumes (DRPC: {{ $labels.obj_name }}, Namespace: {{
    $labels.obj_namespace }}) is taking more than thrice the scheduled
    snapshot interval. This may cause data loss and a backlog of replication
    requests.
  alert_type: DisasterRecovery
```

- Alert: **VolumeSynchronizationDelay**

Impact

Warning

Purpose

Actual sync delay taken by the volume replication group is twice the expected sync delay.

YAML

```
alert: VolumeSynchronizationDela
expr: ramen_rpo_difference > 2 and ramen_rpo_difference < 3
for: 5s
labels:
  cluster: '{{ $labels.cluster }}'
  severity: critical
```

annotations:

description: >-

Syncing of volumes (DRPC: {{ \$labels.obj_name }}, Namespace: {{ \$labels.obj_namespace }}) is taking more than twice the scheduled snapshot interval. This may cause data loss and a backlog of replication requests.

alert_type: DisasterRecovery

CHAPTER 7. TROUBLESHOOTING DISASTER RECOVERY

7.1. TROUBLESHOOTING METRO-DR

7.1.1. A statefulset application stuck after failover

Problem

While relocating to a preferred cluster, DRPlacementControl is stuck reporting PROGRESSION as "MovingToSecondary".

Previously, before Kubernetes v1.23, the Kubernetes control plane never cleaned up the PVCs created for StatefulSets. This activity was left to the cluster administrator or a software operator managing the StatefulSets. Due to this, the PVCs of the StatefulSets were left untouched when their Pods were deleted. This prevents Ramen from relocating an application to its preferred cluster.

Resolution

1. If the workload uses StatefulSets, and relocation is stuck with PROGRESSION as "MovingToSecondary", then run:

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

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

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

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

3. Run the following command

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

After a few seconds to a few minutes, the PROGRESSION reports "Completed" and relocation is complete.

Result

The workload is relocated to the preferred cluster

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

7.1.2. DR policies protect all applications in the same namespace

Problem

While only a 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: [\[2128860\]](#)

7.1.3. 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 clusters are up). When performing failback, application is 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: [\[2120201\]](#)

7.1.4. Relocate or failback might be stuck in Initiating state

Problem

When a primary cluster is down and comes back online while the secondary goes down, **relocate** or **failback** might be stuck in the **Initiating** state.

Resolution

To avoid this situation, cut off all access from the old active hub to the managed clusters. Alternatively, you can scale down the ApplicationSet controller on the old active hub cluster either before moving workloads or when they are in the clean-up phase.

On the old active hub, scale down the two deployments using the following commands:

```
$ oc scale deploy -n openshift-gitops-operator openshift-gitops-operator-controller-manager --replicas=0
```

```
$ oc scale statefulset -n openshift-gitops openshift-gitops-application-controller --replicas=0
```

BZ reference: [\[2243804\]](#)

7.2. TROUBLESHOOTING REGIONAL-DR

7.2.1. 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	<p>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.</p> <p>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.</p>
leader: true or no output	Contact Red Hat Support

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

7.2.2. volsync-rsync-src pod is in error state as it is unable to resolve the destination hostname

Problem

VolSync source pod is unable to resolve the hostname of the VolSync destination pod. The log of the VolSync Pod consistently shows an error message over an extended period of time similar to the following log snippet.

```
$ oc logs -n busybox-workloads-3-2 volsync-rsync-src-dd-io-pvc-1-p25rz
```

Example output

```
VolSync rsync container version: ACM-0.6.0-ce9a280
Syncing data to volsync-rsync-dst-dd-io-pvc-1.busybox-workloads-3-2.svc.cluster.local:22 ...
ssh: Could not resolve hostname volsync-rsync-dst-dd-io-pvc-1.busybox-workloads-3-2.svc.cluster.local: Name or service not known
```

Resolution

Restart **submariner-lighthouse-agent** on both nodes.

```
$ oc delete pod -l app=submariner-lighthouse-agent -n submariner-operator
```

7.2.3. Cleanup and data sync for ApplicationSet workloads remain stuck after older primary managed cluster is recovered post failover

Problem

ApplicationSet based workload deployments to managed clusters are not garbage collected in cases when the hub cluster fails. It is recovered to a standby hub cluster, while the workload has been failed over to a surviving managed cluster. The cluster that the workload was failed over from, rejoins the new recovered standby hub.

ApplicationSets that are DR protected, with a regional DRPolicy, hence starts firing the VolumeSynchronizationDelay alert. Further such DR protected workloads cannot be failed over to the peer cluster or relocated to the peer cluster as data is out of sync between the two clusters.

Resolution

The workaround requires that **openshift-gitops** operators can own the workload resources that are orphaned on the managed cluster that rejoined the hub post a failover of the workload was performed from the new recovered hub. To achieve this the following steps can be taken:

1. Determine the Placement that is in use by the ArgoCD ApplicationSet resource on the hub cluster in the **openshift-gitops** namespace.
2. Inspect the placement label value for the ApplicationSet in this field:
spec.generators.clusterDecisionResource.labelSelector.matchLabels
This would be the name of the Placement resource *<placement-name>*
3. Ensure that there exists a **PlacemenDecision** for the ApplicationSet referenced **Placement**.

```
$ oc get placementdecision -n openshift-gitops --selector cluster.open-cluster-management.io/placement=<placement-name>
```

This results in a single **PlacementDecision** that places the workload in the currently desired failover cluster.

4. Create a new **PlacementDecision** for the ApplicationSet pointing to the cluster where it should be cleaned up.

For example:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/decision-group-index: "1" # Typically one higher
    than the same value in the esisting PlacementDecision determined at step (2)
    cluster.open-cluster-management.io/decision-group-name: ""
    cluster.open-cluster-management.io/placement: cephfs-appset-busybox10-placement
    name: <placemen-name>-decision-<n> # <n> should be one higher than the existing
    PlacementDecision as determined in step (2)
  namespace: openshift-gitops
```

5. Update the newly created **PlacementDecision** with a status **subresource**.

```
decision-status.yaml:
status:
  decisions:
    - clusterName: <managedcluster-name-to-clean-up> # This would be the cluster from
      where the workload was failed over, NOT the current workload cluster
      reason: FailoverCleanup
```

```
$ oc patch placementdecision -n openshift-gitops <placemen-name>-decision-<n> --
patch-file=decision-status.yaml --subresource=status --type=merge
```

6. Watch and ensure that the Application resource for the ApplicationSet has been placed on the desired cluster

```
$ oc get application -n openshift-gitops <applicationset-name>-<managedcluster-name-
to-clean-up>
```

In the output, check if the SYNC STATUS shows as **Synced** and the HEALTH STATUS shows as **Healthy**.

7. Delete the PlacementDecision that was created in step (3), such that ArgoCD can garbage collect the workload resources on the <managedcluster-name-to-clean-up>

```
$ oc delete placementdecision -n openshift-gitops <placemen-name>-decision-<n>
```

ApplicationSets that are DR protected, with a regional DRPolicy, stops firing the **VolumeSynchronizationDelay** alert.

BZ reference: [\[2268594\]](#)

7.3. TROUBLESHOOTING 2-SITE STRETCH CLUSTER WITH ARBITER

7.3.1. Recovering workload pods stuck in ContainerCreating state post zone recovery

Problem

After performing complete zone failure and recovery, the workload pods are sometimes stuck in **ContainerCreating** state with the any of the below errors:

- MountDevice failed to create newCsiDriverClient: driver name openshift-storage.rbd.csi.ceph.com not found in the list of registered CSI drivers
- MountDevice failed for volume <volume_name> : rpc error: code = Aborted desc = an operation with the given Volume ID <volume_id> already exists
- MountVolume.Setup failed for volume <volume_name> : rpc error: code = Internal desc = staging path <path> for volume <volume_id> is not a mountpoint

Resolution

If the workload pods are stuck with any of the above mentioned errors, perform the following workarounds:

- For **ceph-fs** workload stuck in **ContainerCreating**:
 1. Restart the nodes where the stuck pods are scheduled
 2. Delete these stuck pods
 3. Verify that the new pods are running
- For **ceph-rbd** workload stuck in **ContainerCreating** that do not self recover after sometime
 1. Restart csi-rbd plugin pods in the nodes where the stuck pods are scheduled
 2. Verify that the new pods are running