Red Hat OpenShift Data Foundation 4.9

Configuring OpenShift Data Foundation for Metro-DR stretch cluster

Instructions on how to configure OpenShift Data Foundation between two different geographical locations for providing storage infrastructure with disaster recovery capabilities.

Last Updated: 2022-05-16

Red Hat OpenShift Data Foundation 4.9 Configuring OpenShift Data Foundation for Metro-DR stretch cluster

Instructions on how to configure OpenShift Data Foundation between two different geographical locations for providing storage infrastructure with disaster recovery capabilities.

Legal Notice

Abstract

The intent of this solution guide is to detail the steps and commands necessary to deploy OpenShift Data Foundation along with Kubernetes zone topology labels to achieve a highly available storage infrastructure. Configuring OpenShift Data Foundation for Metro-DR is a technology preview feature. Technology Preview features are not supported with Red Hat production service level agreements (SLAs) and might not be functionally complete. Red Hat does not recommend using them in production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE ... 3

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ... 4

CHAPTER 1. INTRODUCTION TO METRO-DR STRETCH CLUSTER ... 5

CHAPTER 2. PREPARING TO DEPLOY STORAGE CLUSTER WITH DISASTER RECOVERY ENABLED ... 6
- 2.1. REQUIREMENTS FOR ENABLING METRO-DR 6
- 2.2. APPLYING TOPOLOGY ZONE LABELS TO OPENSHIFT CONTAINER PLATFORM NODES 6
- 2.3. INSTALLING LOCAL STORAGE OPERATOR 7
- 2.4. INSTALLING RED HAT OPENSHIFT DATA FOUNDATION OPERATOR 7

CHAPTER 3. CREATING OPENSHIFT DATA FOUNDATION CLUSTER ... 10

CHAPTER 4. VERIFYING OPENSHIFT DATA FOUNDATION DEPLOYMENT ... 14
- 4.1. VERIFYING THE STATE OF THE PODS 14
- 4.2. VERIFYING THE OPENSHIFT DATA FOUNDATION CLUSTER IS HEALTHY 16
- 4.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY 16
- 4.4. VERIFYING THAT THE OPENSHIFT DATA FOUNDATION SPECIFIC STORAGE CLASSES EXIST 16

CHAPTER 5. INSTALLING ZONE AWARE SAMPLE APPLICATION ... 18
- 5.1. INSTALL ZONE AWARE SAMPLE APPLICATION 18
- 5.2. MODIFY DEPLOYMENT TO BE ZONE AWARE 21

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

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

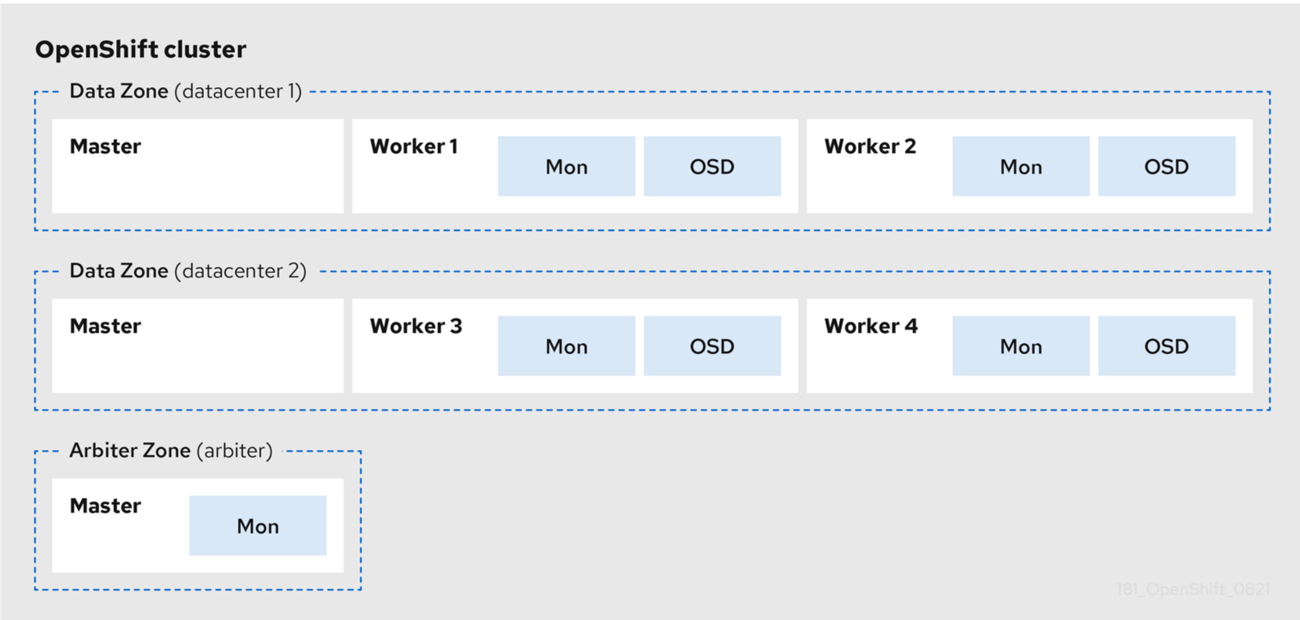
CHAPTER 1. INTRODUCTION TO METRO-DR STRETCH CLUSTER

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**
>
> Currently, you can deploy the Metro-DR solution using stretch cluster where latencies do not exceed four milliseconds round-trip time (RTT) between the OpenShift Container Platform nodes in different locations. Contact Red Hat Customer Support if you are planning to deploy with higher latencies.

The following diagram shows the simplest deployment for a Metro-DR 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 does not show 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.

CHAPTER 2. PREPARING TO DEPLOY STORAGE CLUSTER WITH DISASTER RECOVERY ENABLED

2.1. REQUIREMENTS FOR ENABLING METRO-DR

- 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 cluster 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 Metro-DR solution is designed for deployments where latencies do not exceed 2 ms between zones, with a maximum round-trip time (RTT) of 4 ms. 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.

2.2. APPLYING TOPOLOGY ZONE LABELS TO OPENSHIFT 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 it's zone.

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

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

Next step

- Install the storage operators from OpenShift Container Platform OperatorHub .

2.3. 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 either **4.9** or **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.

2.4. INSTALLING RED HAT OPENSHIFT 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-4.9**.
   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

Verify that the **OpenShift Data Foundation** Operator shows a green tick indicating successful installation.

Next steps

- Create an OpenShift Data Foundation cluster .

CHAPTER 3. CREATING OPENSHIFT DATA FOUNDATION CLUSTER

Prerequisites

- Ensure that you have met all the requirements in Preparing to deploy storage cluster with disaster recovery enabled 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 type 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. 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 Preparing to deploy storage cluster with disaster recovery enabled [Technology Preview].
   Select the **arbiter zone** from the dropdown list.

   b. **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.

   c. Click **Next**.

7. Optional: In the **Security and network** page, configure the following based on your requirement:

   a. Select the **Enable encryption** checkbox to encrypt block and file storage.

   b. Choose one or both of the following **Encryption level**:

      - **Cluster-wide encryption**
        Encrypts the entire cluster (block and file).

      - **StorageClass encryption**
        Creates encrypted persistent volume (block only) using encryption enabled storage class.

   c. Select **Connect to an external key management service** checkbox. This is optional for cluster-wide encryption.

      i. **Key Management Service Provider** is set to **Vault** by default.

      ii. Enter Vault **Service Name**, host **Address** of Vault server ('https://*<hostname or ip>*''), **Port** number and **Token**.

d. Expand **Advanced Settings** to enter the additional settings and certificate details based on your Vault configuration:

   i. Enter the Key Value secret path in the **Backend Path** that is dedicated and unique to OpenShift Data Foundation.

   ii. Optional: Enter the **TLS Server Name** and **Vault Enterprise Namespace**.

   iii. Upload the respective PEM encoded certificate file to provide the **CA Certificate**, **Client Certificate** and **Client Private Key**.

e. Click **Save**.

f. Choose one of the following:

   - **Default (SDN)**
     If you are using a single network.

   - **Custom (Multus)**
     If you are using multiple network interfaces.

     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 **NetworkAttachementDefinition**, that is,**ocs-public-cluster** for the Public Network Interface, and leave the Cluster Network Interface blank.

g. Click **Next**.

8. 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.

9. Click **Create StorageSystem**.

10. For cluster-wide encryption with Key Management System (KMS), if you have used the Vault Key/Value (KV) secret engine API, version 2, then you need to edit the **configmap**.

    a. In the OpenShift Web Console, navigate to **Workloads → ConfigMaps**.

    b. To view the KMS connection details, click **ocs-kms-connection-details**.

    c. Edit the configmap.

       i. Click **Action menu (⋮) → Edit ConfigMap**

       ii. Set the **VAULT_BACKEND** parameter to **v2**.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ocs-kms-connection-details
[...]
```

```
data:
  KMS_PROVIDER: vault
  KMS_SERVICE_NAME: vault
[...]
  VAULT_BACKEND: v2
[...]
```

iii. Click **Save**.

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 **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 .

CHAPTER 4. VERIFYING OPENSHIFT 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 .

4.1. 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 4.1, "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 4.1. Pods corresponding to OpenShift Data Foundation cluster

Component	Corresponding pods
OpenShift Data Foundation Operator	• **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)
Rook-ceph Operator	**rook-ceph-operator-*** (1 pod on any worker node)

Component	Corresponding pods
Multicloud Object Gateway	• **noobaa-operator-*** (1 pod on any worker node) • **noobaa-core-*** (1 pod on any storage node) • **nooba-db-*** (1 pod on any storage node) • **noobaa-endpoint-*** (1 pod on any storage node)
MON	**rook-ceph-mon-*** (5 pods are distributed across 3 zones, 2 per data-center zones and 1 in arbiter zone)
MGR	**rook-ceph-mgr-*** (2 pods on any storage node)
MDS	**rook-ceph-mds-ocs-storagecluster-cephfilesystem-*** (2 pods are distributed across 2 data-center zones)
RGW	**rook-ceph-rgw-ocs-storagecluster-cephobjectstore-*** (2 pods are distributed across 2 data-center zones)

CSI	• **cephfs**
	  ◦ **csi-cephfsplugin-*** (1 pod on each worker node)
	  ◦ **csi-cephfsplugin-provisioner-*** (2 pods distributed across worker nodes)
	• **rbd**
	  ◦ **csi-rbdplugin-*** (1 pod on each worker node)
	  ◦ **csi-rbdplugin-provisioner-*** (2 pods distributed across worker nodes)

rook-ceph-crashcollector	**rook-ceph-crashcollector-*** (1 pod on each storage node and 1 pod in arbiter zone)

Component	Corresponding pods
OSD	• **rook-ceph-osd-*** (1 pod for each device) • **rook-ceph-osd-prepare-ocs-deviceset-*** (1 pod for each device)

4.2. VERIFYING THE OPENSHIFT DATA FOUNDATION CLUSTER IS HEALTHY

Procedure

1. In the OpenShift Web Console, click **Storage → OpenShift 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 *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.

4.3. VERIFYING THE MULTICLOUD OBJECT GATEWAY IS HEALTHY

Procedure

1. In the OpenShift Web Console, click **Storage → OpenShift 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.

4.4. VERIFYING THAT THE OPENSHIFT DATA FOUNDATION 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**

CHAPTER 5. INSTALLING ZONE AWARE SAMPLE APPLICATION

Deploy a zone aware sample application to validate whether an OpenShift Data Foundation, Metro-DR setup is configured correctly.

IMPORTANT

With latency between the data zones, one 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). How much will the performance get degraded, 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 Metro-DR cluster configuration to ensure sufficient application performance for the required service levels.

5.1. INSTALL ZONE AWARE SAMPLE APPLICATION

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 Metro-DR 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 cr
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-regis
try.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610c
0e05b593844b41d5494ea
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/openshift/php@s
ha256:d97466f33999951739a76bce922ab17088885db610c0e05b593844b41d5494ea
STEP 2: LABEL "io.openshift.build.commit.author"="Christian Hernandez <christ
ian.hernandez@yahoo.com>"       "io.openshift.build.commit.date"="Sun Oct 1 1
7:15:09 2017 -0700"       "io.openshift.build.commit.id"="288eda3dff43b02f7f7
b6b6b6f93396ffdf34cb2"       "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-uploa
d-demo"       "io.openshift.build.image"="image-registry.openshift-image-regi
stry.svc:5000/openshift/php@sha256:d97466f33999951739a76bce922ab17088885db610
c0e05b593844b41d5494ea"
STEP 3: ENV OPENSHIFT_BUILD_NAME="file-uploader-1"
OPENSHIFT_BUILD_NAMESP
ACE="my-shared-storage"     OPENSHIFT_BUILD_SOURCE="https://github.com/christ
ianh814/openshift-php-upload-demo"     OPENSHIFT_BUILD_COMMIT="288eda3dff43b0
2f7f7b6b6b6f93396ffdf34cb2"
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 provide
d 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 once 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 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 it’s 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.

5.2. 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 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 tozero pods

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

Example output:

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

Scaling up tofour 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+5fbfd19   datacenter1
perf1-mz8bt-worker-k68rv   Ready    worker   35d   v1.20.0+5fbfd19   datacenter1
perf1-mz8bt-worker-ntkp8   Ready    worker   35d   v1.20.0+5fbfd19   datacenter2
perf1-mz8bt-worker-qpwsr   Ready    worker   35d   v1.20.0+5fbfd19   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
   ```

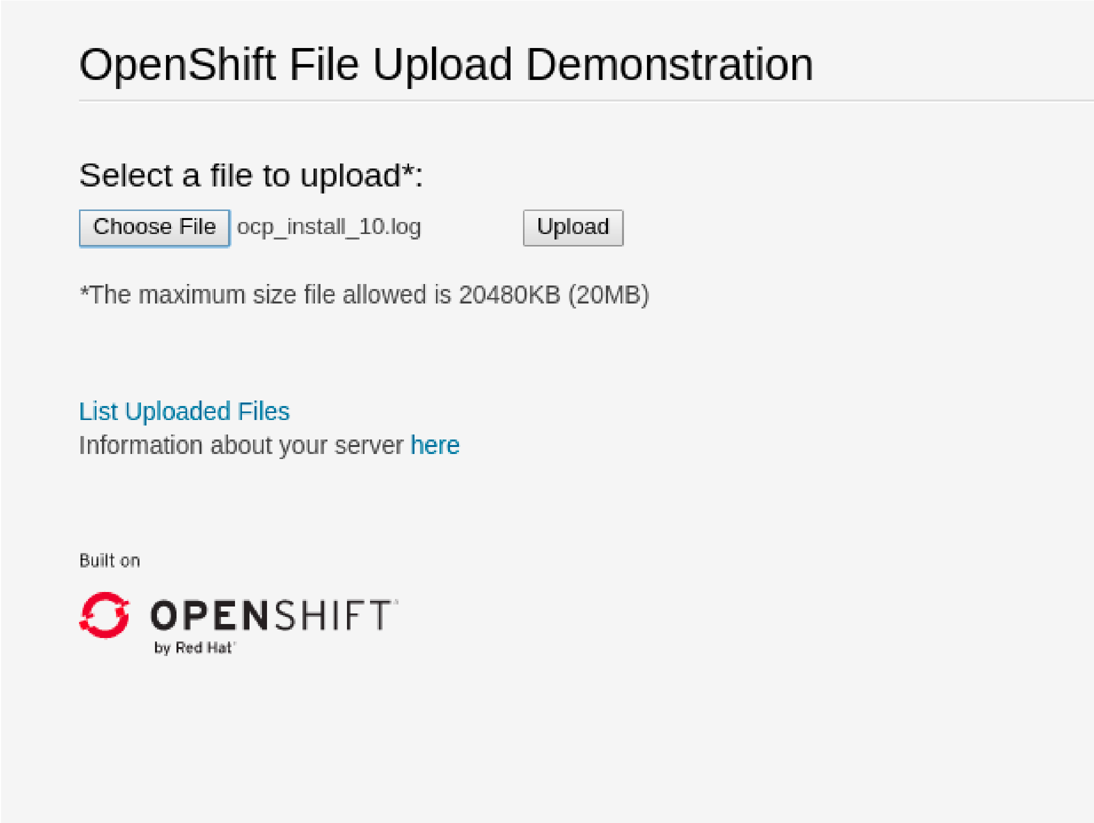
   b. Point your browser to the web application using the route in the previous step.
   The web application lists all the uploaded files and offers the ability to upload new ones as well as you download the existing data. Right now, there is nothing.

   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.