Red Hat OpenShift Container Storage 4.8

Troubleshooting OpenShift Container Storage

How to troubleshoot errors and issues in OpenShift Container Storage

Last Updated: 2022-08-25

Red Hat OpenShift Container Storage 4.8 Troubleshooting OpenShift Container Storage

How to troubleshoot errors and issues in OpenShift Container Storage

Legal Notice

Abstract

Read this document for instructions on troubleshooting Red Hat OpenShift Container Storage.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

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

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

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

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

CHAPTER 1. OVERVIEW

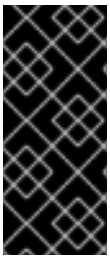
Troubleshooting OpenShift Container Storage is written to help administrators understand how to troubleshoot and fix their Red Hat OpenShift Container Storage cluster.

Most troubleshooting tasks focus on either a fix or a workaround. This document is divided into chapters based on the errors that an administrator may encounter:

- Chapter 2, *Downloading log files and diagnostic information using must-gather* shows you how to use the must-gather utility in OpenShift Container Storage.
- Chapter 3, *Commonly required logs for troubleshooting* shows you how to obtain commonly required log files for OpenShift Container Storage.
- Chapter 6, *Troubleshooting alerts and errors in OpenShift Container Storage* shows you how to identify the encountered error and perform required actions.

CHAPTER 2. DOWNLOADING LOG FILES AND DIAGNOSTIC INFORMATION USING MUST-GATHER

If Red Hat OpenShift Container Storage is unable to automatically resolve a problem, use the must-gather tool to collect log files and diagnostic information so that you or Red Hat support can review the problem and determine a solution.



IMPORTANT

When OpenShift Container Storage is deployed in external mode, must-gather only collects logs from the Red Hat OpenShift Container Storage cluster and does not collect debug data and logs from the external Red Hat Ceph Storage cluster. To collect debug logs from the external Red Hat Ceph Storage cluster, see Red Hat Ceph Storage Troubleshooting guide and contact your Red Hat Ceph Storage Administrator.

Procedure

- Run the **must-gather** command from the client connected to the OpenShift Container Storage cluster:

```
$ oc adm must-gather --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.8 --dest-dir=<directory-name>
```

  This collects the following information in the specified directory:

  - All OpenShift Container Storage cluster related Custom Resources (CRs) with their namespaces.
  - Pod logs of all the OpenShift Container Storage related pods.
  - Output of some standard Ceph commands like Status, Cluster health, and others.

Command variations

- If one or more master nodes are not in the **Ready** state, use **--node-name** to provide a master node that is **Ready** so that the **must-gather** pod can be safely scheduled.

```
$ oc adm must-gather --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.8 --dest-dir=<directory-name> --node-name=<node-name>
```

- If you want to gather information from a specific time:
  - To specify a relative time period for logs gathered, such as within 5 seconds or 2 days, add **/usr/bin/gather since=<duration>**:

```
$ oc adm must-gather --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.8 --dest-dir=<directory-name> /usr/bin/gather since=<duration>
```

  - To specify a specific time to gather logs after, add **/usr/bin/gather since-time=<rfc3339-timestamp>**:

```
$ oc adm must-gather --image=registry.redhat.io/ocs4/ocs-must-gather-rhel8:v4.8 --dest-dir=<directory-name> /usr/bin/gather since-time=<rfc3339-timestamp>
```

Replace the example values in these commands as follows:

<node-name>

If one or more master nodes are not in the **Ready** state, use this parameter to provide the name of a master node that is still in the **Ready** state. This avoids scheduling errors by ensuring that the **must-gather** pod is not scheduled on a master node that is not ready.

<directory-name>

The directory to store information collected by **must-gather**.

<duration>

Specify the period of time to collect information from as a relative duration, for example, **5h** (starting from 5 hours ago).

<rfc3339-timestamp>

Specify the period of time to collect information from as an RFC 3339 timestamp, for example, **2020-11-10T04:00:00+00:00** (starting from 4am UTC on 11 Nov 2020).

CHAPTER 3. COMMONLY REQUIRED LOGS FOR TROUBLESHOOTING

Some of the commonly used logs for troubleshooting OpenShift Container Storage are listed, along with the commands to generate them.

- Generating logs for a specific pod:

```
$ oc logs <pod-name> -n <namespace>
```

- Generating logs for Ceph or OpenShift Container Storage cluster:

```
$ oc logs rook-ceph-operator-<ID> -n openshift-storage
```

IMPORTANT

Currently, the rook-ceph-operator logs do not provide any information about the failure and this acts as a limitation in troubleshooting issues, see Enabling and disabling debug logs for rook-ceph-operator.

- Generating logs for plugin pods like cephfs or rbd to detect any problem in the PVC mount of the app-pod:

```
$ oc logs csi-cephfsplugin-<ID> -n openshift-storage -c csi-cephfsplugin
```

```
$ oc logs csi-rbdplugin-<ID> -n openshift-storage -c csi-rbdplugin
```

  - To generate logs for all the containers in the CSI pod:

```
$ oc logs csi-cephfsplugin-<ID> -n openshift-storage --all-containers
```

```
$ oc logs csi-rbdplugin-<ID> -n openshift-storage --all-containers
```

- Generating logs for cephfs or rbd provisioner pods to detect problems if PVC is not in **BOUND** state:

```
$ oc logs csi-cephfsplugin-provisioner-<ID> -n openshift-storage -c csi-cephfsplugin
```

```
$ oc logs csi-rbdplugin-provisioner-<ID> -n openshift-storage -c csi-rbdplugin
```

  - To generate logs for all the containers in the CSI pod:

```
$ oc logs csi-cephfsplugin-provisioner-<ID> -n openshift-storage --all-containers
```

```
$ oc logs csi-rbdplugin-provisioner-<ID> -n openshift-storage --all-containers
```

- Generating OpenShift Container Storage logs using cluster-info command:

```
$ oc cluster-info dump -n openshift-storage --output-directory=<directory-name>
```

- Check the OpenShift Container Storage operator logs and events.
  - To check the operator logs :

    ```
    # oc logs <ocs-operator> -n openshift-storage
    ```

    <ocs-operator>

    ```
    # oc get pods -n openshift-storage | grep -i "ocs-operator" | awk '{print $1}'
    ```

  - To check the operator events :

    ```
    # oc get events --sort-by=metadata.creationTimestamp -n openshift-storage
    ```

- Get the OpenShift Container Storage operator version and channel.

  ```
  # oc get csv -n openshift-storage
  ```

  Example output :

  ```
  NAME                  DISPLAY VERSION            REPLACES
  PHASE
  ocs-operator.v4.7.2      OpenShift Container Storage  4.7.2
  Succeeded
  ```

  ```
  # oc get subs -n openshift-storage
  ```

  Example output :

  ```
  NAME         PACKAGE        SOURCE
  CHANNEL
  ocs-operator  ocs-operator   redhat-operators
  stable-4.8
  ```

- Confirm that the installplan is created.

  ```
  # oc get installplan -n openshift-storage
  ```

- Verify the image of the components post updating OpenShift Container Storage.
  - Check the node on which the pod of the component you want to verify the image is running.

    ```
    # oc get pods -o wide | grep <component-name>
    ```

    For Example :

    ```
    # oc get pods -o wide | grep rook-ceph-operator
    ```

    Example output:

    ```
    rook-ceph-operator-566cc677fd-bjqnb 1/1 Running 20 4h6m 10.128.2.5 rook-ceph-
    ```

```
operator-566cc677fd-bjqnb 1/1 Running 20 4h6m 10.128.2.5 dell-r440-
12.gsslab.pnq2.redhat.com <none> <none>

<none> <none>
```

dell-r440-12.gsslab.pnq2.redhat.com is the **node-name**.

- Check the image ID.

```
# oc debug node/<node name>
```

<node-name>

Is the name of the node on which the pod of the component you want to verify the image is running.

```
# chroot /host
```

```
# crictl images | grep <component>
```

For Example :

```
# crictl images | grep rook-ceph
```

Example output:

```
IMAGE                                        TAG
    IMAGEID         SIZE
registry.redhat.io/ocs4/rook-ceph-rhel8-operator@sha256   <none>
    5600a36370df4    1.55GB
```

Take a note of the **IMAGEID** and map it to the **Digest** ID on the Rook Ceph Operator page.

Additional resources

- Using must-gather

CHAPTER 4. OVERRIDING THE CLUSTER-WIDE DEFAULT NODE SELECTOR FOR OPENSHIFT CONTAINER STORAGE POST DEPLOYMENT

When a cluster-wide default node selector is used for Openshift Container Storage, the pods generated by CSI daemonsets are able to start only on the nodes that match the selector. To be able to use Openshift Container Storage from nodes which do not match the selector, override the **cluster-wide default node selector** by performing the following steps in the command line interface :

Procedure

1. Specify a blank node selector for the openshift-storage namespace.

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

2. Delete the original pods generated by the DaemonSets.

```
oc delete pod -l app=csi-cephfsplugin -n openshift-storage
oc delete pod -l app=csi-rbdplugin -n openshift-storage
```

CHAPTER 5. ENCRYPTION TOKEN IS DELETED OR EXPIRED

Use this procedure to update the token if the encryption token for your key management system gets deleted or expires.

Prerequisites

- Ensure that you have a new token with the same policy as the deleted or expired token

Procedure

1. Log in to OpenShift Container Platform Web Console.
2. Click **Workloads → Secrets**
3. To update the **ocs-kms-token** used for cluster wide encryption:
   a. Set the **Project** to **openshift-storage**.
   b. Click **ocs-kms-token → Actions → Edit Secret**.
   c. Drag and drop or upload your encryption token file in the **Value** field. The token can either be a file or text that can be copied and pasted.
   d. Click **Save**.
4. To update the **ceph-csi-kms-token** for a given project or namespace with encrypted persistent volumes:
   a. Select the required **Project**.
   b. Click **ceph-csi-kms-token → Actions → Edit Secret**.
   c. Drag and drop or upload your encryption token file in the **Value** field. The token can either be a file or text that can be copied and pasted.
   d. Click **Save**.

NOTE

The token can be deleted only after all the encrypted PVCs using the **ceph-csi-kms-token** have been deleted.

CHAPTER 6. TROUBLESHOOTING ALERTS AND ERRORS IN OPENSHIFT CONTAINER STORAGE

6.1. RESOLVING ALERTS AND ERRORS

Red Hat OpenShift Container Storage can detect and automatically resolve a number of common failure scenarios. However, some problems require administrator intervention.

To know the errors currently firing, check one of the following locations:

- **Monitoring → Alerting → Firing** option
- **Home → Overview → Cluster** tab
- **Storage → Overview → Block and File** tab
- **Storage → Overview → Object** tab

Copy the error displayed and search it in the following section to know its severity and resolution:

Name: **CephMonVersionMismatch**

Message: **There are multiple versions of storage services running.**

Description: **There are {{ $value }} different versions of Ceph Mon components running.**

Severity: Warning

Resolution: Fix

Procedure: Inspect the user interface and log, and verify if an update is in progress.

- If an update in progress, this alert is temporary.
- If an update is not in progress, restart the upgrade process.

Name: **CephOSDVersionMismatch**

Message: **There are multiple versions of storage services running.**

Description: **There are {{ $value }} different versions of Ceph OSD components running.**

Severity: Warning

Resolution: Fix

Procedure: Inspect the user interface and log, and verify if an update is in progress.

- If an update in progress, this alert is temporary.
- If an update is not in progress, restart the upgrade process.

Name: **CephClusterCriticallyFull**

Message: **Storage cluster is critically full and needs immediate expansion**

Description: **Storage cluster utilization has crossed 85%.**

Severity: Crtical

Resolution: Fix

Procedure: Remove unnecessary data or expand the cluster.

Name: **CephClusterNearFull**

Fixed: **Storage cluster is nearing full. Expansion is required.**

Description: **Storage cluster utilization has crossed 75%.**

Severity: Warning

Resolution: Fix

Procedure: Remove unnecessary data or expand the cluster.

Name: **NooBaaBucketErrorState**

Message: **A NooBaa Bucket Is In Error State**

Description: **A NooBaa bucket {{ $labels.bucket_name }} is in error state for more than 6m**

Severity: Warning

Resolution: Workaround

Procedure: Resolving NooBaa Bucket Error State

Name: **NooBaaNamespaceResourceErrorState**

Message: **A NooBaa Namespace Resource Is In Error State**

Description: **A NooBaa namespace resource {{ $labels.namespace_resource_name }} is in error state for more than 5m**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Error State

Name: **NooBaaNamespaceBucketErrorState**

Message: **A NooBaa Namespace Bucket Is In Error State**

Description: **A NooBaa namespace bucket {{ $labels.bucket_name }} is in error state for more than 5m**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Error State

Name: **NooBaaBucketExceedingQuotaState**

Message: **A NooBaa Bucket Is In Exceeding Quota State**

Description: **A NooBaa bucket {{ $labels.bucket_name }} is exceeding its quota - {{ printf "%0.0f" $value }}% used message: A NooBaa Bucket Is In Exceeding Quota State**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Exceeding Quota State

Name: **NooBaaBucketLowCapacityState**

Message: **A NooBaa Bucket Is In Low Capacity State**

Description: **A NooBaa bucket {{ $labels.bucket_name }} is using {{ printf "%0.0f" $value }}% of its capacity**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **NooBaaBucketNoCapacityState**

Message: **A NooBaa Bucket Is In No Capacity State**

Description: **A NooBaa bucket {{ $labels.bucket_name }} is using all of its capacity**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **NooBaaBucketReachingQuotaState**

Message: **A NooBaa Bucket Is In Reaching Quota State**

Description: **A NooBaa bucket {{ $labels.bucket_name }} is using {{ printf "%0.0f" $value }}% of its quota**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **NooBaaResourceErrorState**

Message: **A NooBaa Resource Is In Error State**

Description: **A NooBaa resource {{ $labels.resource_name }} is in error state for more than 6m**

Severity: Warning

Resolution: Workaround

Procedure: Resolving NooBaa Bucket Error State

Name: **NooBaaSystemCapacityWarning100**

Message: **A NooBaa System Approached Its Capacity**

Description: **A NooBaa system approached its capacity, usage is at 100%**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **NooBaaSystemCapacityWarning85**

Message: **A NooBaa System Is Approaching Its Capacity**

Description: **A NooBaa system is approaching its capacity, usage is more than 85%**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **NooBaaSystemCapacityWarning95**

Message: **A NooBaa System Is Approaching Its Capacity**

Description: **A NooBaa system is approaching its capacity, usage is more than 95%**

Severity: Warning

Resolution: Fix

Procedure: Resolving NooBaa Bucket Capacity or Quota State

Name: **CephMdsMissingReplicas**

Message: **Insufficient replicas for storage metadata service.**

Description: `Minimum required replicas for storage metadata service not available.

Might affect the working of storage cluster.`

Severity: Warning

Resolution: Contact Red Hat support

Procedure:

1. Check for alerts and operator status.
2. If the issue cannot be identified, contact Red Hat support.

Name: **CephMgrIsAbsent**

Message: **Storage metrics collector service not available anymore.**

Description: **Ceph Manager has disappeared from Prometheus target discovery.**

Severity: Critical

Resolution: Contact Red Hat support

Procedure:

1. Inspect the user interface and log, and verify if an update is in progress.
   - If an update in progress, this alert is temporary.
   - If an update is not in progress, restart the upgrade process.
2. Once the upgrade is complete, check for alerts and operator status.
3. If the issue persistents or cannot be identified, contact Red Hat support.

Name: **CephNodeDown**

Message: **Storage node {{ $labels.node }} went down**

Description: **Storage node {{ $labels.node }} went down. Please check the node immediately.**

Severity: Critical

Resolution: Contact Red Hat support

Procedure:

1. Check which node stopped functioning and its cause.
2. Take appropriate actions to recover the node. If node cannot be recovered:
   - See Replacing storage nodes for OpenShift Container Storage
   - Contact Red Hat support.

Name: **CephClusterErrorState**

Message: **Storage cluster is in error state**

Description: **Storage cluster is in error state for more than 10m.**

Severity: Critical

Resolution: Contact Red Hat support

Procedure:

1. Check for alerts and operator status.
2. If the issue cannot be identified, download log files and diagnostic information using must-gather.
3. Open a Support Ticket with Red Hat Support with an attachment of the output of must-gather.

Name: **CephClusterWarningState**

Message: **Storage cluster is in degraded state**

Description: **Storage cluster is in warning state for more than 10m.**

Severity: Warning

Resolution: Contact Red Hat support

Procedure:

1. Check for alerts and operator status.
2. If the issue cannot be identified, download log files and diagnostic information using must-gather.
3. Open a Support Ticket with Red Hat Support with an attachment of the output of must-gather.

Name: **CephDataRecoveryTakingTooLong**

Message: **Data recovery is slow**

Description: **Data recovery has been active for too long.**

Severity: Warning

Resolution: Contact Red Hat support

Name: **CephOSDDiskNotResponding**

Message: **Disk not responding**

Description: **Disk device {{ $labels.device }} not responding, on host {{ $labels.host }}.**

Severity: Critical

Resolution: Contact Red Hat support

Name: **CephOSDDiskUnavailable**

Message: **Disk not accessible**

Description: **Disk device {{ $labels.device }} not accessible on host {{ $labels.host }}.**

Severity: Critical

Resolution: Contact Red Hat support

Name: **CephPGRepairTakingTooLong**

Message: **Self heal problems detected**

Description: **Self heal operations taking too long.**

Severity: Warning

Resolution: Contact Red Hat support

Name: **CephMonHighNumberOfLeaderChanges**

Message: **Storage Cluster has seen many leader changes recently.**

Description: **'Ceph Monitor "{{ $labels.job }}": instance {{ $labels.instance }} has seen {{ $value printf "%.2f" }} leader changes per minute recently.'**

Severity: Warning

Resolution: Contact Red Hat support

Name: **CephMonQuorumAtRisk**

Message: **Storage quorum at risk**

Description: **Storage cluster quorum is low.**

Severity: Critical

Resolution: Contact Red Hat support

Name: **ClusterObjectStoreState**

Message: **Cluster Object Store is in unhealthy state. Please check Ceph cluster health**.

Description: **Cluster Object Store is in unhealthy state for more than 15s. Please check Ceph cluster health**.

Severity: Critical

Resolution: Contact Red Hat support

Procedure:

- Check the **CephObjectStore** CR instance.
- Contact Red Hat support.

Name: **CephOSDFlapping**

Message: **Storage daemon osd.x has restarted 5 times in the last 5 minutes. Please check the pod events or Ceph status to find out the cause**.

Description: **Storage OSD restarts more than 5 times in 5 minutes**.

Severity: Critical

Resolution: Contact Red Hat support

6.2. RESOLVING NOOBAA BUCKET ERROR STATE

Procedure

1. Log in to OpenShift Web Console and click **Storage → Overview → Object**
2. In the **Details** card, click the link under **System Name** field.
3. In the left pane, click **Buckets** option and search for the bucket in error state. If the bucket in error state is a namespace bucket, be sure to click the **Namespace Buckets** pane.
4. Click on it's **Bucket Name**. Error encountered in bucket is displayed.
5. Depending on the specific error of the bucket, perform one or both of the following:
    a. For space related errors:

i. In the left pane, click **Resources** option.

ii. Click on the resource in error state.

iii. Scale the resource by adding more agents.

b. For resource health errors:

i. In the left pane, click **Resources** option.

ii. Click on the resource in error state.

iii. Connectivity error means the backing service is not available and needs to be restored.

iv. For access/permissions errors, update the connection's **Access Key** and **Secret Key**.

6.3. RESOLVING NOOBAA BUCKET EXCEEDING QUOTA STATE

To resolve **A NooBaa Bucket Is In Exceeding Quota State** error perform one of the following:

- Cleanup some of the data on the bucket.
- Increase the bucket quota by performing the following steps:
  1. Log in to OpenShift Web Console and click **Storage → Overview → Object**
  2. In the **Details** card, click the link under **System Name** field.
  3. In the left pane, click **Buckets** option and search for the bucket in error state.
  4. Click on it's **Bucket Name**. Error encountered in bucket is displayed.
  5. Click **Bucket Policies → Edit Quota** and increase the quota.

6.4. RESOLVING NOOBAA BUCKET CAPACITY OR QUOTA STATE

Procedure

1. Log in to OpenShift Web Console and click **Storage → Overview → Object**
2. In the **Details** card, click the link under **System Name** field.
3. In the left pane, click **Resources** option and search for the PV pool resource.
4. For the PV pool resource with low capacity status, click on it's **Resource Name**.
5. Edit the pool configuration and increase the number of agents.

6.5. RECOVERING PODS

When a first node (say **NODE1**) goes to NotReady state because of some issue, the hosted pods that are using PVC with ReadWriteOnce (RWO) access mode try to move to the second node (say **NODE2**) but get stuck due to multi-attach error. In such a case, you can recover MON, OSD, and application pods by using the following steps.

Procedure

1. Power off **NODE1** (from AWS or vSphere side) and ensure that **NODE1** is completely down.
2. Force delete the pods on **NODE1** by using the following command:

```
$ oc delete pod <pod-name> --grace-period=0 --force
```

6.6. RECOVERING FROM EBS VOLUME DETACH

When an OSD or MON elastic block storage (EBS) volume where the OSD disk resides is detached from the worker Amazon EC2 instance, the volume gets reattached automatically within one or two minutes. However, the OSD pod gets into a **CrashLoopBackOff** state. To recover and bring back the pod to **Running** state, you must restart the EC2 instance.

6.7. ENABLING AND DISABLING DEBUG LOGS FOR ROOK-CEPH-OPERATOR

Enable the debug logs for the rook-ceph-operator to obtain information about failures that help in troubleshooting issues.

Procedure

Enabling the debug logs

1. Edit the configmap of the rook-ceph-operator.

```
$ oc edit configmap rook-ceph-operator-config
```

2. Add the **ROOK_LOG_LEVEL: DEBUG** parameter in the **rook-ceph-operator-config** yaml file to enable the debug logs for rook-ceph-operator.

```
…
data:
  # The logging level for the operator: INFO | DEBUG
  ROOK_LOG_LEVEL: DEBUG
```

Now, the rook-ceph-operator logs consist of the debug information.

Disabling the debug logs

1. Edit the configmap of the rook-ceph-operator.

```
$ oc edit configmap rook-ceph-operator-config
```

2. Add the **ROOK_LOG_LEVEL: INFO** parameter in the **rook-ceph-operator-config** yaml file to disable the debug logs for rook-ceph-operator.

```
…
data:
  # The logging level for the operator: INFO | DEBUG
  ROOK_LOG_LEVEL: INFO
```

CHAPTER 7. CHECKING FOR LOCAL STORAGE OPERATOR DEPLOYMENTS

OpenShift Container Storage clusters with Local Storage Operator are deployed using local storage devices. To find out if your existing cluster with OpenShift Container Storage was deployed using local storage devices, use the following procedure:

Prerequisites

- OpenShift Container Storage is installed and running in the **openshift-storage** namespace.

Procedure

By checking the storage class associated with your OpenShift Container Storage cluster's persistent volume claims (PVCs), you can tell if your cluster was deployed using local storage devices.

1. Check the storage class associated with OpenShift Container Storage cluster's PVCs with the following command:

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

2. Check the output. For clusters with Local Storage Operator, the PVCs associated with **ocs-deviceset** use the storage class **localblock**. The output looks similar to the following:

```
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS
MODES   STORAGECLASS              AGE
db-noobaa-db-0        Bound    pvc-d96c747b-2ab5-47e2-b07e-1079623748d8   50Gi
RWO           ocs-storagecluster-ceph-rbd   114s
ocs-deviceset-0-0-lzfrd   Bound    local-pv-7e70c77c                       1769Gi     RWO
localblock                2m10s
ocs-deviceset-1-0-7rggl   Bound    local-pv-b19b3d48                       1769Gi     RWO
localblock                2m10s
ocs-deviceset-2-0-znhk8   Bound    local-pv-e9f22cdc                       1769Gi     RWO
localblock                2m10s
```

Additional Resources

- Deploying OpenShift Container Storage using local storage devices on VMware
- Deploying OpenShift Container Storage using local storage devices on Red Hat Virtualization
- Deploying OpenShift Container Storage using local storage devices on bare metal

CHAPTER 8. TROUBLESHOOTING AND DELETING REMAINING RESOURCES DURING UNINSTALL

Occasionally some of the custom resources managed by an operator may remain in "Terminating" status waiting on the finalizer to complete, although you have performed all the required cleanup tasks. In such an event you need to force the removal of such resources. If you do not do so, the resources remain in the "Terminating" state even after you have performed all the uninstall steps.

1. Check if the openshift-storage namespace is stuck in Terminating state upon deletion.

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

Output:

```
NAME                DISPLAY NAME   STATUS
openshift-storage                  Terminating
```

2. Check for the **NamespaceFinalizersRemaining** and **NamespaceContentRemaining** messages in the **STATUS** section of the command output and perform the next step for each of the listed resources.

```
$ oc get project openshift-storage -o yaml
```

Example output :

```
status:
  conditions:
  - lastTransitionTime: "2020-07-26T12:32:56Z"
    message: All resources successfully discovered
    reason: ResourcesDiscovered
    status: "False"
    type: NamespaceDeletionDiscoveryFailure
  - lastTransitionTime: "2020-07-26T12:32:56Z"
    message: All legacy kube types successfully parsed
    reason: ParsedGroupVersions
    status: "False"
    type: NamespaceDeletionGroupVersionParsingFailure
  - lastTransitionTime: "2020-07-26T12:32:56Z"
    message: All content successfully deleted, may be waiting on finalization
    reason: ContentDeleted
    status: "False"
    type: NamespaceDeletionContentFailure
  - lastTransitionTime: "2020-07-26T12:32:56Z"
    message: 'Some resources are remaining: cephobjectstoreusers.ceph.rook.io has
      1 resource instances'
    reason: SomeResourcesRemain
    status: "True"
    type: NamespaceContentRemaining
  - lastTransitionTime: "2020-07-26T12:32:56Z"
    message: 'Some content in the namespace has finalizers remaining:
cephobjectstoreuser.ceph.rook.io
      in 1 resource instances'
```

```
    reason: SomeFinalizersRemain
    status: "True"
    type: NamespaceFinalizersRemaining
```

3. Delete all the remaining resources listed in the previous step.
   For each of the resources to be deleted, do the following:

   a. Get the object kind of the resource which needs to be removed. See the message in the above output.
      Example :

      message: Some content in the namespace has finalizers remaining: cephobjectstoreuser.ceph.rook.io

      Here **cephobjectstoreuser.ceph.rook.io** is the object kind.

   b. Get the Object name corresponding to the object kind.

      ```
      $ oc get  <Object-kind> -n  <project-name>
      ```

      Example :

      ```
      $ oc get cephobjectstoreusers.ceph.rook.io -n openshift-storage
      ```

      Example output:

      ```
      NAME                          AGE
      noobaa-ceph-objectstore-user   26h
      ```

   c. Patch the resources.

      ```
      $ oc patch -n <project-name> <object-kind>/<object-name> --type=merge -p
      '{"metadata": {"finalizers":null}}'
      ```

      Example:

      ```
      $ oc patch -n openshift-storage cephobjectstoreusers.ceph.rook.io/noobaa-ceph-
      objectstore-user \
      --type=merge -p '{"metadata": {"finalizers":null}}'
      ```

      Output:

      ```
      cephobjectstoreuser.ceph.rook.io/noobaa-ceph-objectstore-user patched
      ```

4. Verify that the openshift-storage project is deleted.

   ```
   $ oc get project openshift-storage
   ```

   Output:

   ```
   Error from server (NotFound): namespaces "openshift-storage" not found
   ```

   If the issue persists, reach out to Red Hat Support.

CHAPTER 9. TROUBLESHOOTING CEPHFS PVC CREATION IN EXTERNAL MODE

If you have updated the Red Hat Ceph Storage cluster from a version lower than 4.1.1 to the latest release and is not a freshly deployed cluster, you must manually set the application type for CephFS pool on the Red Hat Ceph Storage cluster to enable CephFS PVC creation in external mode.

1. Check for CephFS pvc stuck in **Pending** status.

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

Example output :

```
NAME                    STATUS    VOLUME
CAPACITY  ACCESS MODES    STORAGECLASS                  AGE
ngx-fs-pxknkcix20-pod     Pending
                    ocs-external-storagecluster-cephfs  28h
[...]
```

2. Check the **describe** output to see the events for respective pvc.
Expected error message is **cephfs_metadata/csi.volumes.default/csi.volume.pvc-xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx: (1) Operation not permitted)**

```
# oc describe pvc ngx-fs-pxknkcix20-pod -n nginx-file
```

Example output:

```
Name:          ngx-fs-pxknkcix20-pod
Namespace:     nginx-file
StorageClass:  ocs-external-storagecluster-cephfs
Status:        Pending
Volume:
Labels:        <none>
Annotations:   volume.beta.kubernetes.io/storage-provisioner: openshift-
storage.cephfs.csi.ceph.com
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    ngx-fs-oyoe047v2bn2ka42jfgg-pod-hqhzf
Events:
 Type     Reason              Age                 From
Message
 ----     ------              ----                ----
-------
 Warning  ProvisioningFailed  107m (x245 over 22h)  openshift-
storage.cephfs.csi.ceph.com_csi-cephfsplugin-provisioner-5f8b66cc96-hvcqp_6b7044af-
c904-4795-9ce5-bf0cf63cc4a4
 (combined from similar events): failed to provision volume with StorageClass "ocs-external-
storagecluster-cephfs": rpc error: code = Internal desc = error (an error (exit status 1)
occurred while
 running rados args: [-m 192.168.13.212:6789,192.168.13.211:6789,192.168.13.213:6789 --
id csi-cephfs-provisioner --keyfile=stripped -c /etc/ceph/ceph.conf -p cephfs_metadata
getomapval
```

```
 csi.volumes.default csi.volume.pvc-1ac0c6e6-9428-445d-bbd6-1284d54ddb47 /tmp/omap-get-186436239 --namespace=csi]) occurred, command output streams is ( error getting omap value
 cephfs_metadata/csi.volumes.default/csi.volume.pvc-1ac0c6e6-9428-445d-bbd6-1284d54ddb47: (1) Operation not permitted)
```

3. Check the settings for the **<cephfs metadata pool name>** (here **cephfs_metadata**) and **<cephfs data pool name>** (here **cephfs_data**). For running the command, you will need **jq** preinstalled in the Red Hat Ceph Storage client node.

```
# ceph osd pool ls detail --format=json | jq '.[] | select(.pool_name| startswith("cephfs")) | .pool_name, .application_metadata' "cephfs_data"
{
  "cephfs": {}
}
"cephfs_metadata"
{
   "cephfs": {}
}
```

4. Set the application type for CephFS pool.

   - Run the following commands on the Red Hat Ceph Storage client node :

     ```
     # ceph osd pool application set <cephfs metadata pool name> cephfs metadata cephfs
     ```

     ```
     # ceph osd pool application set <cephfs data pool name> cephfs data cephfs
     ```

5. Verify if the settings are applied.

```
 # ceph osd pool ls detail --format=json | jq '.[] | select(.pool_name| startswith("cephfs")) | .pool_name, .application_metadata' "cephfs_data"
 {
   "cephfs": {
     "data": "cephfs"
  }
 }
 "cephfs_metadata"
 {
   "cephfs": {
     "metadata": "cephfs"
   }
 }
```

6. Check the CephFS PVC status again. The PVC should now be in **Bound** state.

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

Example output :

```
NAME                     STATUS    VOLUME
CAPACITY  ACCESS MODES    STORAGECLASS                  AGE
ngx-fs-pxknkcix20-pod     Bound     pvc-1ac0c6e6-9428-445d-bbd6-1284d54ddb47
```

```
1Mi        RWO            ocs-external-storagecluster-cephfs   29h
[...]
```

CHAPTER 10. RESTORING THE MONITOR PODS IN OPENSHIFT CONTAINER STORAGE

Restore the monitor pods if all three of them go down, and when OpenShift Container Storage is not able to recover the monitor pods automatically.

Procedure

1. Scale down the **rook-ceph-operator** and **ocs operator** deployments.

```
# oc scale deployment rook-ceph-operator --replicas=0 -n openshift-storage
```

```
# oc scale deployment ocs-operator --replicas=0 -n openshift-storage
```

2. Create a backup of all deployments in the **openshift-storage** namespace.

```
# mkdir backup
```

```
# cd backup
```

```
# oc project openshift-storage
```

```
# for d in $(oc get deployment|awk -F' ' '{print $1}'|grep -v NAME); do echo $d;oc get deployment $d -o yaml > oc_get_deployment.${d}.yaml; done
```

3. Patch the OSD deployments to remove the **livenessProbe** parameter, and run it with the command parameter as **sleep**.

```
# for i in $(oc get deployment -l app=rook-ceph-osd -oname);do oc patch ${i} -n openshift-storage --type='json' -p '[{"op":"remove", "path":"/spec/template/spec/containers/0/livenessProbe"}]' ; oc patch ${i} -n openshift-storage -p '{"spec": {"template": {"spec": {"containers": [{"name": "osd", "command": ["sleep", "infinity"], "args": []}]}}}}' ; done
```

4. Retrieve the **monstore** cluster map from all the OSDs.

   a. Create the **recover_mon.sh** script.

```
#!/bin/bash
ms=/tmp/monstore

rm -rf $ms
mkdir $ms

for osd_pod in $(oc get po -l app=rook-ceph-osd -oname -n openshift-storage); do

  echo "Starting with pod: $osd_pod"

  podname=$(echo $osd_pod|sed 's/pod\///g')
  oc exec $osd_pod -- rm -rf $ms
  oc cp $ms $podname:$ms
```

```
  rm -rf $ms
  mkdir $ms

  echo "pod in loop: $osd_pod ; done deleting local dirs"

  oc exec $osd_pod -- ceph-objectstore-tool --type bluestore --data-path
/var/lib/ceph/osd/ceph-$(oc get $osd_pod -ojsonpath='{
.metadata.labels.ceph_daemon_id }') --op update-mon-db --no-mon-config --mon-store-
path $ms
  echo "Done with COT on pod: $osd_pod"

  oc cp $podname:$ms $ms

  echo "Finished pulling COT data from pod: $osd_pod"
done
```

b. Run the **recover_mon.sh** script.

```
# chmod +x recover_mon.sh
```

```
# ./recover_mon.sh
```

5. Patch the MON deployments, and run it with the command parameter as **sleep**.

a. Edit the MON deployments.

```
# for i in $(oc get deployment -l app=rook-ceph-mon -oname);do oc patch ${i} -n
openshift-storage -p '{"spec": {"template": {"spec": {"containers": [{"name": "mon",
"command": ["sleep", "infinity"], "args": []}]}}}}'; done
```

b. Patch the MON deployments to increase the **initialDelaySeconds**.

```
# oc get deployment rook-ceph-mon-a -o yaml | sed "s/initialDelaySeconds:
10/initialDelaySeconds: 2000/g" | oc replace -f -
```

```
# oc get deployment rook-ceph-mon-b -o yaml | sed "s/initialDelaySeconds:
10/initialDelaySeconds: 2000/g" | oc replace -f -
```

```
# oc get deployment rook-ceph-mon-c -o yaml | sed "s/initialDelaySeconds:
10/initialDelaySeconds: 2000/g" | oc replace -f -
```

6. Copy the previously retrieved **monstore** to the **mon-a** pod.

```
# oc cp /tmp/monstore/ $(oc get po -l app=rook-ceph-mon,mon=a -oname |sed
's/pod\///g'):/tmp/
```

7. Navigate into the MON pod and change the ownership of the retrieved **monstore**.

```
# oc rsh $(oc get po -l app=rook-ceph-mon,mon=a -oname)
```

```
# chown -R ceph:ceph /tmp/monstore
```

8. Copy the keyring template file before rebuilding the **mon db**.

```
# oc rsh $(oc get po -l app=rook-ceph-mon,mon=a -oname)
```

```
# cp /etc/ceph/keyring-store/keyring /tmp/keyring
```

```
# cat /tmp/keyring
 [mon.]
   key = AQCleqldWqm5IhAAgZQbEzoShkZV42RiQVffnA==
   caps mon = "allow *"
 [client.admin]
   key = AQCmAKld8J05KxAArOWeRAw63gAwwZO5o75ZNQ==
   auid = 0
   caps mds = "allow *"
   caps mgr = "allow *"
   caps mon = "allow *"
   caps osd = "allow *”
```

9. Identify the keyring of all the other Ceph daemons (MGR, MDS, RGW, Crash, CSI and CSI provisioners) from its respective secrets.

```
# oc get secret rook-ceph-mds-ocs-storagecluster-cephfilesystem-a-keyring -ojson  | jq
.data.keyring | xargs echo | base64 -d

[mds.ocs-storagecluster-cephfilesystem-a]
key = AQB3r8VgAtr6OhAAVhhXpNKqRTuEVdRoxG4uRA==
caps mon = "allow profile mds"
caps osd = "allow *"
caps mds = "allow"
```

Example keyring file, **/etc/ceph/ceph.client.admin.keyring**:

```
[mon.]
 key = AQDxTF1hNgLTNxAAi51cCojs01b4I5E6v2H8Uw==
 caps mon = "allow "
[client.admin]
        key = AQDxTF1hpzguOxAA0sS8nN4udoO35OEbt3bqMQ==
        caps mds = "allow " caps mgr = "allow *" caps mon = "allow *" caps osd = "allow *"
[mds.ocs-storagecluster-cephfilesystem-a] key =
AQCKTV1horgjARAA8aF/BDh/4+eG4RCNBCl+aw== caps mds = "allow" caps mon = "allow
profile mds" caps osd = "allow *" [mds.ocs-storagecluster-cephfilesystem-b] key =
AQCKTV1hN4gKLBAA5emIVq3ncV7AMEM1c1RmGA== caps mds = "allow" caps mon =
"allow profile mds" caps osd = "allow *" [client.rgw.ocs.storagecluster.cephobjectstore.a] key
= AQCOkdBixmpiAxAA4X7zjn6SGTI9c1MBflszYA== caps mon = "allow rw" caps osd =
"allow rwx" [mgr.a] key = AQBOTV1hGYOEORAA87471+eIZLZtptfkcHvTRg== caps mds =
"allow *" caps mon = "allow profile mgr" caps osd = "allow *" [client.crash] key =
AQBOTV1htO1aGRAAe2MPYcGdiAT+Oo4CNPSF1g== caps mgr = "allow rw" caps mon =
"allow profile crash" [client.csi-cephfs-node] key =
AQBOTV1hiAtuBBAAaPPBVgh1AqZJlDeHWdoFLw== caps mds = "allow rw" caps mgr =
"allow rw" caps mon = "allow r" caps osd = "allow rw tag cephfs *=" [client.csi-cephfs-
provisioner] key = AQBNTV1hHu6wMBAAzNXZv36aZJuE1iz7S7GfeQ== caps mgr = "allow
rw" caps mon = "allow r" caps osd = "allow rw tag cephfs metadata="
[client.csi-rbd-node]
 key = AQBNTV1h+LnkIRAAWnpIN9bUAmSHOvJ0EJXHRw==
```

```
 caps mgr = "allow rw"
 caps mon = "profile rbd"
 caps osd = "profile rbd"
[client.csi-rbd-provisioner]
 key = AQBNTV1hMNcsExAAvA3gHB2qaY33LOdWCvHG/A==
 caps mgr = "allow rw"
 caps mon = "profile rbd"
 caps osd = "profile rbd"
```

IMPORTANT

- For **client.csi** related keyring, refer to the previous keyring file output and add the default **caps** after fetching the key from its respective OpenShift Container Storage secret.
- OSD keyring is added automatically post recovery.

10. Navigate into the **mon-a** pod, and verify that the **monstore** has **monmap**.

    a. Navigate into the **mon-a** pod.

    ```
    # oc rsh $(oc get po -l app=rook-ceph-mon,mon=a -oname)
    ```

    b. Verify that the **monstore** has **monmap**.

    ```
    # ceph-monstore-tool /tmp/monstore get monmap -- --out /tmp/monmap
    ```

    ```
    # monmaptool /tmp/monmap --print
    ```

11. Optional: If the **monmap** is missing then create a new **monmap**.

    ```
    # monmaptool --create --add <mon-a-id> <mon-a-ip> --add <mon-b-id> <mon-b-ip> --add
    <mon-c-id> <mon-c-ip> --enable-all-features --clobber /root/monmap --fsid <fsid>
    ```

    <mon-a-id>

    Is the ID of the **mon-a** pod.

    <mon-a-ip>

    Is the IP address of the **mon-a** pod.

    <mon-b-id>

    Is the ID of the **mon-b** pod.

    <mon-b-ip>

    Is the IP address of the **mon-b** pod.

    <mon-c-id>

    Is the ID of the **mon-c** pod.

    <mon-c-ip>

    Is the IP address of the **mon-c** pod.

    <fsid>

    Is the file system ID.

12. Verify the **monmap**.

```
# monmaptool /root/monmap --print
```

13. Import the **monmap**.

IMPORTANT

Use the previously created **keyring** file.

```
# ceph-monstore-tool /tmp/monstore rebuild -- --keyring /tmp/keyring --monmap
/root/monmap
```

```
# chown -R ceph:ceph /tmp/monstore
```

14. Create a backup of the old **store.db** file.

```
# mv /var/lib/ceph/mon/ceph-a/store.db /var/lib/ceph/mon/ceph-a/store.db.corrupted
```

```
# mv /var/lib/ceph/mon/ceph-b/store.db /var/lib/ceph/mon/ceph-b/store.db.corrupted
```

```
# mv /var/lib/ceph/mon/ceph-c/store.db /var/lib/ceph/mon/ceph-c/store.db.corrupted
```

15. Copy the rebuild **store.db** file to the **monstore** directory.

```
# mv /tmp/monstore/store.db /var/lib/ceph/mon/ceph-a/store.db
```

```
# chown -R ceph:ceph /var/lib/ceph/mon/ceph-a/store.db
```

16. After rebuilding the **monstore** directory, copy the **store.db** file from local to the rest of the MON pods.

```
# oc cp $(oc get po -l app=rook-ceph-mon,mon=a -oname | sed
's/pod\///g'):/var/lib/ceph/mon/ceph-a/store.db /tmp/store.db
```

```
# oc cp /tmp/store.db $(oc get po -l app=rook-ceph-mon,mon=<id> -oname | sed
's/pod\///g'):/var/lib/ceph/mon/ceph-<id>
```

<id>

Is the ID of the MON pod

17. Navigate into the rest of the MON pods and change the ownership of the copied **monstore**.

```
# oc rsh $(oc get po -l app=rook-ceph-mon,mon=<id> -oname)
```

```
# chown -R ceph:ceph /var/lib/ceph/mon/ceph-<id>/store.db
```

<id>

Is the ID of the MON pod

18. Revert the patched changes.

    - For MON deployments:

      ```
      # oc replace --force -f <mon-deployment.yaml>
      ```

      <mon-deployment.yaml>

      Is the MON deployment yaml file

    - For OSD deployments:

      ```
      # oc replace --force -f <osd-deployment.yaml>
      ```

      <osd-deployment.yaml>

      Is the OSD deployment yaml file

    - For MGR deployments:

      ```
      # oc replace --force -f <mgr-deployment.yaml>
      ```

      <mgr-deployment.yaml>

      Is the MGR deployment yaml file

      IMPORTANT

      Ensure that the MON, MGR and OSD pods are up and running.

19. Scale up the **rook-ceph-operator** and **ocs-operator** deployments.

    ```
    # oc -n openshift-storage scale deployment ocs-operator --replicas=1
    ```

Verification steps

1. Check the Ceph status to confirm that CephFS is running.

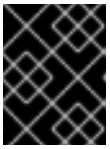
   ```
   # ceph -s
   ```

   Example output:

   ```
   cluster:
     id:     f111402f-84d1-4e06-9fdb-c27607676e55
     health: HEALTH_ERR
             1 filesystem is offline
             1 filesystem is online with fewer MDS than max_mds
             3 daemons have recently crashed

   services:
     mon: 3 daemons, quorum b,c,a (age 15m)
     mgr: a(active, since 14m)
     mds: ocs-storagecluster-cephfilesystem:0
     osd: 3 osds: 3 up (since 15m), 3 in (since 2h)
   ```

```
data:
  pools:   3 pools, 96 pgs
  objects: 500 objects, 1.1 GiB
  usage:   5.5 GiB used, 295 GiB / 300 GiB avail
  pgs:     96 active+clean
```

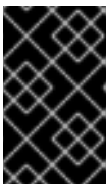


IMPORTANT

If the filesystem is offline or MDS service is missing, you need to restore the CephFS. For more information, see Section 10.1, "Restoring the CephFS".

2. Check the Multicloud Object Gateway (MCG) status. It should be active, and the backingstore and bucketclass should be in **Ready** state.

```
noobaa status -n openshift-storage
```



IMPORTANT

If the MCG is not in the active state, and the backingstore and bucketclass not in the **Ready** state, you need to restart all the MCG related pods. For more information, see Section 10.2, "Restoring the Multicloud Object Gateway" .

10.1. RESTORING THE CEPHFS

If the filesystem is offline or MDS service is missing you need to restore the CephFS.

Procedure

1. Scale down the **rook-ceph-operator** and **ocs operator** deployments.

```
# oc scale deployment rook-ceph-operator --replicas=0 -n openshift-storage
```

```
# oc scale deployment ocs-operator --replicas=0 -n openshift-storage
```

2. Patch the MDS deployments to remove the **livenessProbe** parameter and run it with the command parameter as **sleep**.

```
# for i in $(oc get deployment -l app=rook-ceph-mds -oname);do oc patch ${i} -n openshift-storage --type='json' -p '[{"op":"remove", "path":"/spec/template/spec/containers/0/livenessProbe"}]' ; oc patch ${i} -n openshift-storage -p '{"spec": {"template": {"spec": {"containers": [{"name": "mds", "command": ["sleep", "infinity"], "args": []}]}}}}' ; done
```

3. Recover the CephFS.

```
# ceph fs reset ocs-storagecluster-cephfilesystem --yes-i-really-mean-it
```

If the **reset** command fails, force create the default filesystem with the data and metadata pools, and then reset it.

NOTE

The **reset** command might fail if the **cephfilesystem** is missing.

```
# ceph fs new ocs-storagecluster-cephfilesystem ocs-storagecluster-cephfilesystem-metadata ocs-storagecluster-cephfilesystem-data0 --force
```

```
# ceph fs reset ocs-storagecluster-cephfilesystem --yes-i-really-mean-it
```

4. Replace the MDS deployments.

```
# oc replace --force -f oc_get_deployment.rook-ceph-mds-ocs-storagecluster-cephfilesystem-a.yaml
```

```
# oc replace --force -f oc_get_deployment.rook-ceph-mds-ocs-storagecluster-cephfilesystem-b.yaml
```

5. Scale up the **rook-ceph-operator** and **ocs-operator** deployments.

```
# oc scale deployment ocs-operator --replicas=1 -n openshift-storage
```

6. Check the CephFS status.

```
# ceph fs status
```

The status should be active.

IMPORTANT

- If the application pods attached to the deployments which were using the CephFS Persistent Volume Claims (PVCs) get stuck in **CreateContainerError** state post restoring the CephFS, restart the application pods.

  ```
  # oc -n <namespace> delete pods <cephfs-app-pod>
  ```

  <namespace>

  Is the project namespace

  <cephfs-app-pod>

  Is the name of the CephFS application pod

- If new CephFS or RBD PVCs are not getting bound, restart all the pods related to Ceph CSI.

10.2. RESTORING THE MULTICLOUD OBJECT GATEWAY

If the Multicloud Object Gateway (MCG) is not in the active state, and the backingstore and bucketclass is not in the **Ready** state, you need to restart all the MCG related pods, and check the MCG status to confirm that the MCG is back up and running.

Procedure

1. Restart all the pods related to the MCG.

```
# oc delete pods <noobaa-operator> -n openshift-storage
```

```
# oc delete pods <noobaa-core> -n openshift-storage
```

```
# oc delete pods <noobaa-endpoint> -n openshift-storage
```

```
# oc delete pods <noobaa-db> -n openshift-storage
```

<noobaa-operator>

Is the name of the MCG operator

<noobaa-core>

Is the name of the MCG core pod

<noobaa-endpoint>

Is the name of the MCG endpoint

<noobaa-db>

Is the name of the MCG db pod

2. If the RADOS Object Gateway (RGW) is configured, restart the pod.

```
# oc delete pods <rgw-pod> -n openshift-storage
```

<rgw-pod>

Is the name of the RGW pod

CHAPTER 11. CHANGING RESOURCES FOR THE OPENSHIFT CONTAINER STORAGE COMPONENTS

When you install OpenShift Container Storage, it comes with pre-defined resources that the OpenShift Container Storage pods can consume. In some situations with higher I/O load, it might be required to increase these limits.

- To change the CPU and memory resources on the rook-ceph pods, see Section 11.1, “Changing the CPU and memory resources on the rook-ceph pods”.
- To tune the resources for the Multicloud Object Gateway (MCG), see Section 11.2, “Tuning the resources for the MCG”.

11.1. CHANGING THE CPU AND MEMORY RESOURCES ON THE ROOK-CEPH PODS

When you install OpenShift Container Storage, it comes with pre-defined CPU and memory resources for the rook-ceph pods. You can manually increase these values according to the requirements.

You can change the CPU and memory resources on the following pods:

- **mgr**
- **mds**
- **rgw**

The following example illustrates how to change the CPU and memory resources on the rook-ceph pods. In this example, the existing MDS pod values of **cpu** and **memory** are increased from **1** and **4Gi** to **2** and **8Gi** respectively.

1. Edit the storage cluster:

   ```
   # oc edit storagecluster -n openshift-storage <storagecluster_name>
   ```

   <storagecluster_name>

   Specify the name of the storage cluster.

   Example 11.1. Example

   ```
   # oc edit storagecluster -n openshift-storage ocs-storagecluster
   ```

2. Add the following lines to the storage cluster Custom Resource (CR):

   ```
   spec:
     resources:
       mds:
         limits:
           cpu: 2
           memory: 8Gi
   ```

```
      requests:
        cpu: 2
        memory: 8Gi
```

3. Save the changes and exit the editor.

4. Alternatively, run the **oc patch** command to change the CPU and memory value of the **mds** pod:

```
# oc patch -n openshift-storage storagecluster <storagecluster_name>
    --type merge \
    --patch '{"spec": {"resources": {"mds": {"limits": {"cpu": "2","memory": "8Gi"},"requests":
{"cpu": "2","memory": "8Gi"}}}}}'
```

<storagecluster_name>

Specify the name of the storage cluster.

Example 11.2. Example

```
# oc patch -n openshift-storage storagecluster ocs-storagecluster \
    --type merge \
    --patch '{"spec": {"resources": {"mds": {"limits": {"cpu": "2","memory": "8Gi"},"requests":
{"cpu": "2","memory": "8Gi"}}}}}'
```

11.2. TUNING THE RESOURCES FOR THE MCG

The default configuration for the Multicloud Object Gateway (MCG) is optimized for low resource consumption and not performance. For more information on how to tune the resources for the MCG, see the Red Hat Knowledgebase solution Performance tuning guide for Multicloud Object Gateway (NooBaa).