Red Hat OpenShift Data Foundation 4.11

OpenShift Data Foundation 故障排除

有关 OpenShift Data Foundation 故障排除的说明

Last Updated: 2023-10-05

Red Hat OpenShift Data Foundation 4.11 OpenShift Data Foundation 故障排除

有关 OpenShift Data Foundation 故障排除的说明

法律通告

摘要

阅读本文档，了解有关 Red Hat OpenShift Data Foundation 故障排除的说明。

目录

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。请告诉我们如何让它更好。提供反馈：

- 关于特定内容的简单评论：
  1. 请确定您使用 *Multi-page HTML* 格式查看文档。另外，确定 **Feedback** 按钮出现在文档页的右上方。
  2. 用鼠标指针高亮显示您想评论的文本部分。
  3. 点在高亮文本上弹出的 **Add Feedback**。
  4. 按照显示的步骤操作。
- 要提交更复杂的反馈，请创建一个 Bugzilla ticket：
  1. 进入 Bugzilla 网站。
  2. 在 **Component** 部分中，选择 **文档**。
  3. 在 **Description** 中输入您要提供的信息。包括文档相关部分的链接。
  4. 点 **Submit Bug**。

第 1 章 概述

OpenShift Data Foundation 故障排除旨在帮助管理员了解如何排除故障并修复其 Red Hat OpenShift Data Foundation 集群。

大多数故障排除任务都侧重于修复或临时解决方案。本文档根据管理员可能遇到的错误分为若干章节：

- 第 2 章 *使用 must-gather 下载日志文件和诊断信息* 如何在 OpenShift Data Foundation 中使用 must-gather 实用程序。
- 第 3 章 *故障排除所需的常见日志* 如何获取 OpenShift Data Foundation 所需的日志文件。
- 第 6 章 *对 OpenShift Data Foundation 中的警报和错误进行故障排除* 如何识别遇到的错误并执行所需的操作。

警告

红帽不支持在 OpenShift Data Foundation 集群中运行 Ceph 命令（除非由红帽支持或红帽文档表示），因为在运行错误的命令时可能会导致数据丢失。在这种情况下，红帽支持团队只能提供商业合理的工作，在出现数据丢失时可能无法恢复所有数据。

第 2 章 使用 MUST-GATHER 下载日志文件和诊断信息

如果 Red Hat OpenShift Data Foundation 无法自动解决问题，请使用 **must-gather** 工具收集日志文件和诊断信息，以便您或红帽支持可以审核问题并确定解决方案。

重要

当将 Red Hat OpenShift Data Foundation 部署为外部模式时，**must-gather** 仅从 OpenShift Data Foundation 集群收集日志，且不会从外部 Red Hat Ceph Storage 集群收集调试数据和日志。要从外部 Red Hat Ceph Storage 集群收集调试日志，请参阅 Red Hat Ceph Storage 故障排除指南并联系您的 Red Hat Ceph Storage 管理员。

先决条件

- 可选：如果 OpenShift Data Foundation 在断开连接的环境中部署，请确保将独立的 **must-gather** 镜像镜像(mirror)到断开连接的环境中可用的镜像 registry。

```
$ oc image mirror registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.11 <local-registry>/odf4/ocs-must-gather-rhel8:v4.11 [--registry-config=<path-to-the-registry-config>] [--insecure=true]
```

<local-registry>

是本地镜像 registry 可用于断开连接的 OpenShift Container Platform 集群。

<path-to-the-registry-config>

是 registry 凭证的路径，默认为 **~/.docker/config.json**。

--insecure

仅在镜像 registry 不安全时才添加此标志。

如需更多信息，请参阅红帽知识库解决方案：

- 如何在 Redhat Openshift registry 间镜像镜像
- 私有 registry 不安全时镜像 OpenShift 镜像存储库失败

流程

- 从连接到 OpenShift Data Foundation 集群的客户端运行 **must-gather** 命令：

```
$ oc adm must-gather --image=registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.11 --dest-dir=<directory-name>
```

<directory-name>

是要将数据写入的目录的名称。

重要

对于断开连接的环境部署，将 **--image** 参数中的镜像替换为镜像的 **must-gather** 镜像。

```
$ oc adm must-gather --image=<local-registry>/odf4/ocs-must-gather-rhel8:v4.11 --dest-dir=<directory-name>
```

<local-registry>

是本地镜像 registry 可用于断开连接的 OpenShift Container Platform 集群。

这会在指定目录中收集以下信息：

- 所有与 Red Hat OpenShift Data Foundation 集群相关的自定义资源(CR)及其命名空间。
- 所有 Red Hat OpenShift Data Foundation 相关 pod 的 Pod 日志。
- 某些标准 Ceph 命令的输出，如状态、集群运行状况等。

命令变体

- 如果一个或多个 master 节点没有处于 **Ready** 状态，请使用 **--node-name** 指定一个状态为 **Ready** 的 master 节点，以便可以安全地调度 **must-gather** pod。

```
$ oc adm must-gather --image=registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.11 --dest-dir=_<directory-name>_ --node-name=_<node-name>_
```

- 如果要从特定时间收集信息：
  - 要为收集的日志指定相对时间段（例如在 5 秒内或在 2 天内），添加 **/usr/bin/gather since=<duration>**：

```
$ oc adm must-gather --image=registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.11 --dest-dir=_<directory-name>_ /usr/bin/gather since=<duration>
```

  - 要指定在以后的一个特定时间收集日志，添加 **/usr/bin/gather since-time=<rfc3339-timestamp>**：

```
$ oc adm must-gather --image=registry.redhat.io/odf4/ocs-must-gather-rhel8:v4.11 --dest-dir=_<directory-name>_ /usr/bin/gather since-time=<rfc3339-timestamp>
```

按如下方式替换这些命令中的示例值：

<node-name>

如果一个或多个 master 节点没有处于 **Ready** 状态，使用这个参数指定一个仍然处于 **Ready** 状态的 master 节点名称。这可避免调度错误，确保 **must-gather** pod 没有调度到未就绪的 master 节点上。

<directory-name>

must-gather 收集的信息的目录。

<duration>

指定收集信息的时长（相对时长），例如 **5h**（代表从 5 小时以前开始）。

<rfc3339-timestamp>

指定收集信息的时常（RFC 3339 时间戳），例如 **2020-11-10T04:00:00+00:00**（代表从 2020 年 11 月 11 日 4am UTC 开始）。

第 3 章 故障排除所需的常见日志

其中列出了一些用于对 OpenShift Data Foundation 进行故障排除的常用日志，以及用于生成这些日志的命令。

- 为特定 pod 生成日志：

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

- 为 Ceph 或 OpenShift Data Foundation 集群生成日志：

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

重要

目前，rook-ceph-operator 日志不提供有关故障的任何信息，这在故障排除中可作为限制，请参阅为 rook-ceph-operator 启用和禁用 debug 日志。

- 为 cephfs 或 rbd 等插件 pod 生成日志，以检测 app-pod 挂载中的任何问题：

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

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

  - 为 CSI pod 中的所有容器生成日志：

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

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

- 为 cephfs 或 rbd provisioner pod 生成日志，以检测 PVC 不处于 **BOUND** 状态的问题：

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

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

  - 为 CSI pod 中的所有容器生成日志：

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

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

- 使用 cluster-info 命令生成 OpenShift Data Foundation 日志：

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

- 使用 Local Storage Operator 时，可以使用 cluster-info 命令生成日志：

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

- 检查 OpenShift Data Foundation 操作器日志和事件。
  - 检查 Operator 日志：

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

    <ocs-operator>

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

  - 检查 Operator 事件 ：

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

- 获取 OpenShift Data Foundation 操作器版本和渠道。

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

  输出示例：

  ```
  NNAME                         DISPLAY                       VERSION   REPLACES   PHASE
  mcg-operator.v4.11.0          NooBaa Operator               4.11.0               Succeeded
  ocs-operator.v4.11.0          OpenShift Container Storage   4.11.0               Succeeded
  odf-csi-addons-operator.v4.11.0   CSI Addons                    4.11.0               Succeeded
  odf-operator.v4.11.0          OpenShift Data Foundation     4.11.0               Succeeded
  ```

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

  输出示例：

  ```
  NAME                                              PACKAGE                SOURCE
  CHANNEL
  mcg-operator-stable-4.11-redhat-operators-openshift-marketplace   mcg-operator
  redhat-operators   stable-4.11
  ocs-operator-stable-4.11-redhat-operators-openshift-marketplace   ocs-operator
  redhat-operators   stable-4.11
  odf-csi-addons-operator                                odf-csi-addons-operator   redhat-operators
  stable-4.11
  odf-operator                                           odf-operator              redhat-operators   stable-
  4.11
  ```

- 确认已创建了安装计划。

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

- 在更新 OpenShift Data Foundation 后，验证组件的镜像。
  - 检查您要在其上验证镜像运行的组件 pod 的节点。

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

    例如：

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

输出示例：

```
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 是 node-name。

- 检查镜像 ID。

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

<node-name>

是您要验证镜像运行的组件 pod 的节点名称。

```
# chroot /host
```

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

例如：

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

记录 **IMAGEID**，并将其映射到 Rook Ceph Operator 页面中的 **Digest** ID。

其它资源

- 使用 must-gather

第 4 章 部署后覆盖 OPENSHIFT DATA FOUNDATION 的集群范围默认节点选择器

当将集群范围的默认节点选择器用于 OpenShift Data Foundation 时，CSI daemonset 生成的 pod 只能在与选择器匹配的节点上启动。要能够从与选择器不匹配的节点使用 OpenShift Data Foundation，请在命令行界面中执行以下步骤来覆盖**集群范围的默认节点选择器** ：

流程

1. 为 openshift-storage 命名空间指定一个空白节点选择器。

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

2. 删除 DaemonSet 生成的原始 pod。

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

第 5 章 加密令牌已删除或过期

如果密钥管理系统的加密令牌被删除或过期，请使用这个流程更新令牌。

先决条件

- 确保您有一个与已删除或过期令牌相同的策略的新令牌

流程

1. 登录 OpenShift Container Platform Web 控制台。
2. 点 **Workloads → Secrets**
3. 更新用于集群范围加密的 **ocs-kms-token** ：
   a. 将 **Project** 设置为 **openshift-storage**。
   b. 点 **ocs-kms-token → Actions → Edit Secret**。
   c. 在 **Value** 字段中拖放或上传您的加密令牌文件。令牌可以是可复制和粘贴的文件或文本。
   d. 点击 **Save**。
4. 为带有加密持久性卷的给定项目或命名空间更新 **ceph-csi-kms-token** ：
   a. 选择所需的项目。
   b. 点 **ceph-csi-kms-token → Actions → Edit Secret**。
   c. 在 **Value** 字段中拖放或上传您的加密令牌文件。令牌可以是可复制和粘贴的文件或文本。
   d. 点击 **Save**。

注意

只有在所有使用 **ceph-csi-kms-token** 的加密 PVC 已被删除后，才能删除令牌。

第 6 章 对 OPENSHIFT DATA FOUNDATION 中的警报和错误进行故障排除

6.1. 解决警报和错误

Red Hat OpenShift Data Foundation 可以检测并自动解决许多常见的故障情形。但是，有些问题需要管理员介入。

要了解当前触发的错误，请查看以下位置之一：

- **Observe → Alerting → Firing** 选项
- **Home → Overview → Cluster** 标签页
- **Storage → Data Foundation → Storage System →** *storage system* 链接，在弹出的 **→ Overview → Block and File** 标签页
- **Storage → Data Foundation → Storage System →** Storage *system* 链接，在弹出 **→ Overview → Object** 标签页

复制显示的错误并在以下部分搜索它以了解其严重性和解决方案：

Name: **CephMonVersionMismatch**

Message: **运行多个存储服务版本。**

Description：**{{ $value }} 运行的 Ceph Mon 组件的不同版本。**

严重性: 警告

解决方案 ：修复

流程 ：检查用户界面和日志，并验证更新是否进行中。

- 如果更新正在进行，则此警报是临时的。
- 如果更新没有进行，重启升级过程。

名称:**CephOSDVersionMismatch**

Message: **运行多个存储服务版本。**

Description：**{{ $value }} 运行的 Ceph OSD 组件的不同版本。**

严重性: 警告

解决方案 ：修复

流程 ：检查用户界面和日志，并验证更新是否进行中。

- 如果更新正在进行，则此警报是临时的。
- 如果更新没有进行，重启升级过程。

名称 ：**CephClusterCriticallyFull**

消息 ：**存储集群几乎已满，需要立即扩展**

描述 ：**存储集群利用率已超过 85%。**

严重性: 关键

解决方案 ：修复

流程 ：删除不必要的数据或扩展集群。

名称 ：**CephClusterNearFull**

修复了:Storage 集群已接近满。需要进行扩展。

描述 ：**存储集群利用率已超过 75%**

严重性: 警告

解决方案 ：修复

流程 ：删除不必要的数据或扩展集群。

Name:**NooBaaBucketErrorState**

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

Description **: NooBaa bucket {{ $labels.bucket_name }}** 处于错误状态，超过 **6m**

严重性: 警告

解决方案 ：临时解决方案

步骤 ：解决 NooBaa Bucket Error State

名称:**NooBaaNamespaceResourceErrorState**

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

描述 : NooBaa 命名空间资源 {{ $labels.namespace_resource_name }} 处于错误状态，表示 **5m** 的错误状态

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Error State

名称:**NooBaaNamespaceBucketErrorState**

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

Description **: NooBaa 命名空间存储桶 {{ $labels.bucket_name }} 处于错误状态，超过 5m**

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Error State

名称:**NooBaaBucketExceedingQuotaState**

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

Description **: NooBaa bucket {{ $labels.bucket_name }} 超过其配额 - {{ printf "%0.0f" $value }}% 使用的消息： NooBaa Bucket Is In Exceeding Quota State**

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Exceeding Quota State

Name:**NooBaaBucketLowCapacityState**

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

Description **: NooBaa bucket {{ $labels.bucket_name }} 正在为其容量使用 {{ printf "%0.0f" $value }}%**

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

Name:**NooBaaBucketNoCapacityState**

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

描述 : NooBaa 存储桶 {{ $labels.bucket_name }} 使用其所有容量

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

Name:**NooBaaBucketReachingQuotaState**

消息 ：NooBaa Bucket Is In Reaching Quota State

Description **: NooBaa bucket {{ $labels.bucket_name }} 正在为其配额使用 {{ printf "%0.0f" $value }}%**

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

Name:**NooBaaResourceErrorState**

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

描述 : NooBaa resource {{ $labels.resource_name }} 处于错误状态，超过 6m

严重性: 警告

解决方案 ：临时解决方案

步骤 ：解决 NooBaa Bucket Error State

Name:**NooBaaSystemCapacityWarning100**

消息 ：NooBaa System Approached Its Capacity

描述 ：NooBaa 系统接近其容量，使用量为 100%

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

Name:**NooBaaSystemCapacityWarning85**

Message **: NooBaa System Is Approaching it Capacity**

描述 ：NooBaa 系统接近其容量，使用时间超过 85%

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

名称 ：NooBaaSystemCapacityWarning95

Message **: NooBaa System Is Approaching it Capacity**

描述 ：NooBaa 系统接近其容量，使用时间超过 95%

严重性: 警告

解决方案 ：修复

步骤 ：解决 NooBaa Bucket Capacity 或 Quota State

Name:**CephMdsMissingReplicas**

Message ：用于存储元数据服务的不计副本。

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

可能会影响存储集群的工作。

严重性: 警告

解决方案 ：请联系红帽支持

流程 ：

1. 检查警报和操作器状态。
2. 如果无法识别该问题，请联系红帽支持团队。

名称 ：CephMgrIsAbsent

Message:**Storage 指标收集器服务不再可用。**

描述 : Ceph Manager 从 Prometheus 目标发现中消失。

严重级别: Critical

解决方案 ：请联系红帽支持

流程 ：

1. 检查用户界面并记录，并验证更新是否正在进行。
   - 如果更新正在进行，则此警报是临时的。
   - 如果更新没有进行，重启升级过程。
2. 升级完成后，检查警报和 Operator 状态。
3. 如果问题持久或无法识别，请联系红帽支持。

名称 ：CephNodeDown

Message: **Storage node {{ $labels.node }} 停机**

描述:Storage node {{ $labels.node }} 停机。请立即检查节点。

严重级别: Critical

解决方案 ：请联系红帽支持

流程 ：

1. 检查哪个节点停止正常运行，并检查其原因。
2. 采取适当的操作来恢复节点。如果无法恢复节点：
   - 请参阅为 Red Hat OpenShift Data Foundation 替换存储节点
   - 联系红帽支持部门.

名称:CephClusterErrorState

Message:**Storage cluster 处于错误状态**

描述:Storage cluster 处于错误状态 10m。

严重级别: Critical

解决方案 ：请联系红帽支持

流程 ：

1. 检查警报和操作器状态。
2. 如果无法识别该问题，请使用 must-gather 下载日志文件和诊断信息。
3. 向红帽支持创建一个支持问题单，并附加 must-gather 的输出。

名称 ：CephClusterWarningState

Message:**Storage cluster 处于 degraded 状态**

描述:Storage cluster 处于 warning 状态，表示 10m 以上的警告状态。

严重性: 警告

解决方案 ：请联系红帽支持

流程 ：

1. 检查警报和操作器状态。
2. 如果无法识别该问题，请使用 must-gather 下载日志文件和诊断信息。
3. 向红帽支持创建一个支持问题单，并附加 must-gather 的输出。

名称 ：CephDataRecoveryTakingTooLong

Message: **Data recovery is slow**

描述 ：数据恢复时间过长。

严重性: 警告

解决方案 ：请联系红帽支持

名称 ：CephOSDDiskNotResponding

Message:**Disk not respond**

描述 : 磁盘设备 {{ $labels.device }} 未响应，在主机 {{ $labels.host }} 上。

严重级别: Critical

解决方案 ：请联系红帽支持

名称 ：CephOSDDiskUnavailable

Message:**Disk not access**

描述:**磁盘设备 {{ $labels.device }} 无法在主机 {{ $labels.host }} 上访问。**

严重级别: Critical

解决方案 ：请联系红帽支持

名称 ：CephPGRepairTakingTooLong

Message:**检测到的自助修复问题**

描述 ：执行自助服务修复操作用时过长。

严重性: 警告

解决方案 ：请联系红帽支持

Name:**CephMonHighNumberOfLeaderChanges**

Message:**Storage Cluster 最近看到很多领导变化。**

描述:**'Ceph Monitor "{{ $labels.job }}": instance {{ $labels.instance }} 已看到 {{ $value printf "%.2f" }} leader 每分钟更改。 '**

严重性: 警告

解决方案 ：请联系红帽支持

名称 ：**CephMonQuorumAtRisk**

消息 ：**存储仲裁的风险**

描述 ：**存储群集仲裁较低。**

严重级别: Critical

解决方案 ：请联系红帽支持

名称 ：**ClusterObjectStoreState**

Message:**Cluster Object Store 处于不健康状态。Please check Ceph cluster health**.

描述:**Cluster Object Store 处于不健康状态，代表超过 15。Please check Ceph cluster health**.

严重级别: Critical

解决方案 ：请联系红帽支持

流程 ：

- 检查 **CephObjectStore** CR 实例。
- 联系红帽支持部门.

名称 ：**CephOSDFlapping**

Message:**Storage daemon osd.x 在最后 5 分钟内重启 5 次。Please check the pod events or Ceph status to find out the cause**.

描述 ：**Storage OSD 在 5 分钟内重新启动超过 5 次。**

严重级别: Critical

解决方案 ：请联系红帽支持

名称 ：**OdfPoolMirroringImageHealth**

Message:**Mirroring image(PV)位于池 <pool-name> 中，超过 1m。Mirroring might not work as expected.**

描述 ：对一个或多个应用程序失败。

严重性: 警告

解决方案 ：请联系红帽支持

名称 ：**OdfMirrorDaemonStatus**

消息 ：**Mirror 守护进程不健康。**

描述 ：对整个集群进行灾难恢复失败。Mirror daemon is in unhealthy status for more than 1m.Mirroring on this cluster is not working as expected.

严重级别: Critical

解决方案 ：请联系红帽支持

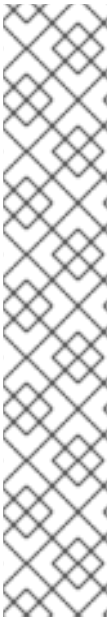
6.2. 解决集群健康问题

Red Hat Ceph Storage 可以在 OpenShift Data Foundation 用户界面中引发该显示的一系列有限健康消息。它们定义为具有唯一标识符的健康检查。标识符是一个制表伪可读字符串，旨在使工具能够理解健康检查，并以反应其含义的方式呈现它们。有关更多信息和故障排除，请单击下面的健康代码。

健康代码	描述
MON_DISK_LOW	一个或多个 Ceph 监控器在磁盘空间上较低。

6.2.1. MON_DISK_LOW

如果将 monitor 数据库存储为百分比的文件系统中的可用空间下降到 **mon_data_avail_warn** 下，则会触发此警报（默认：15%）。这可能表明系统上的某些其他进程或用户正在填满监控器使用的相同文件系统。也可能表明监控器的数据库比较大。



注意

文件系统的路径因您的 mon 部署而异。您可以找到在 **storagecluster.yaml** 中部署 mon 的路径。

路径示例：

- 通过 PVC 路径部署的 mon: **/var/lib/ceph/mon**
- 通过 hostpath 部署 mon: **/var/lib/rook/mon**

若要清除空间，请查看文件系统中的高使用量文件并选择要删除的文件。要查看文件，请运行：

```
# du -a <path-in-the-mon-node> |sort -n -r |head -n10
```

将 **<path-in-the-mon-node>** 替换为部署 mons 的文件系统的路径。

6.3. 解决 NOOBAA BUCKET 错误状态

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。

2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
3. 单击 **Object** 选项卡。
4. 在**详细信息**卡中，单**系统名称**字段下的链接。
5. 在左侧窗格中，单击 **Buckets** 选项并搜索处于错误状态的存储桶。如果处于错误状态的存储桶是一个命名空间存储桶，请确定点 **Namespace Buckets** 窗格。
6. 点它的 **Bucket Name**。此时会显示存储桶中遇到的错误。
7. 根据存储桶的具体错误，执行以下操作之一或两者：
   a. 对于与空间相关的错误：
      i. 在左侧窗格中，点 **Resources** 选项。
      ii. 单击处于错误状态的资源。
      iii. 通过添加更多代理来缩放资源。
   b. 对于资源健康错误：
      i. 在左侧窗格中，点 **Resources** 选项。
      ii. 单击处于错误状态的资源。
      iii. 连接错误意味着后备服务不可用，需要恢复。
      iv. 如需访问/权限错误，请更新连接的**访问密钥**和**机密密钥**。

6.4. 解决 NOOBAA BUCKET EXCEEDING QUOTA STATE 问题

要解决 **A NooBaa Bucket Is In Exceeding Quota State**错误，请执行以下操作之一：

- 清理存储桶上的一些数据。
- 执行以下步骤增加存储桶配额：
  1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
  2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
  3. 单击 **Object** 选项卡。
  4. 在**详细信息**卡中，单**系统名称**字段下的链接。
  5. 在左侧窗格中，单击 **Buckets** 选项并搜索处于错误状态的存储桶。
  6. 点其 **Bucket Name**。此时会显示存储桶中遇到的错误。
  7. 点 **Bucket Policies → Edit Quota** 并增加配额。

6.5. 解决 NOOBAA BUCKET CAPACITY 或 QUOTA STATE 问题

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
3. 单击 **Object** 选项卡。
4. 在**详细信息**卡中，单**系统名称**字段下的链接。
5. 在左侧窗格中，点 **Resources** 选项，再搜索 PV 池资源。
6. 对于具有低容量状态的 PV 池资源，请单击 **Resource Name**。
7. 编辑池配置并增加代理数量。

6.6. 恢复 POD

当第一个节点(例如 **NODE1**)因为出现问题而变为 NotReady 状态时，使用 ReadWriteOnce(RWO)访问模式的 PVC 的托管 pod 会尝试移到第二个节点（例如 **NODE2**），但由于 multi-attach 错误而卡住。在这种情况下，您可以通过下列步骤恢复 MON、OSD 和应用容器集：

流程

1. 关闭 **NODE1** （从 AWS 或 vSphere 端）并确保 **NODE1** 完全关闭。
2. 使用以下命令，强制删除 **NODE1** 上的 pod：

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

6.7. 从 EBS 卷分离中恢复

当 OSD 磁盘所驻留的 OSD 或 MON 弹性块存储(EBS)卷与工作程序 Amazon EC2 实例分离时，该卷会在一两分钟内自动重新附加。但是，OSD 容器集进入 **CrashLoopBackOff** 状态。若要将 pod 恢复并恢复为 **Running** 状态，您必须重新启动 EC2 实例。

6.8. 为 ROOK-CEPH-OPERATOR 启用和禁用 DEBUG 日志

为 rook-ceph-operator 启用 debug 日志，以获取有助于对问题进行故障排除的失败信息。

流程

启用 debug 日志

1. 编辑 rook-ceph-operator 的 configmap。

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

2. 在 **rook-ceph-operator-config** yaml 文件中添加 **ROOK_LOG_LEVEL: DEBUG** 参数，为 rook-ceph-operator 启用调试日志。

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

现在，rook-ceph-operator 日志由 debug 信息组成。

禁用 debug 日志

1. 编辑 rook-ceph-operator 的 configmap。

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

2. 在 **rook-ceph-operator-config** yaml 文件中添加 **ROOK_LOG_LEVEL: INFO** 参数，以禁用 rook-ceph-operator 的调试日志。

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

第 7 章 检查 LOCAL STORAGE OPERATOR 部署

使用本地存储 Operator 的 Red Hat OpenShift Data Foundation 集群是使用本地存储设备部署的。要查找您的 OpenShift Data Foundation 的现有集群是否使用本地存储设备进行了部署，请使用以下步骤：

先决条件

- OpenShift Data Foundation 在 **openshift-storage** 命名空间上安装并运行。

流程

通过检查与 OpenShift Data Foundation 集群的持久性卷声明(PVC)关联的存储类，您可以确定您的集群是否使用本地存储设备部署。

1. 使用以下命令，检查与 OpenShift Data Foundation 集群 PVC 关联的存储类：

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

2. 检查输出。对于使用 Local Storage Operator 的集群，与 **ocs-deviceset** 关联的 PVC 使用存储类 **localblock**。输出结果类似如下：

```
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
```

其它资源

- 使用 VMware 上的本地存储设备部署 OpenShift Data Foundation
- 使用 Red Hat Virtualization 上的本地存储设备部署 OpenShift Data Foundation
- 使用裸机上的本地存储设备部署 OpenShift Data Foundation
- 使用 IBM Power 上的本地存储设备部署 OpenShift Data Foundation

第 8 章 删除失败或不需要的 CEPH 对象存储设备

失败或不需要的 Ceph OSD （对象存储设备）会影响存储基础架构的性能。因此，为了提高存储集群的可靠性和弹性，您必须删除失败或不需要的 Ceph OSD。

如果您有故障或不需要的 Ceph OSD 来删除：

1. 验证 Ceph 健康状态。
   有关更多信息，请参阅：验证 Ceph 集群是否健康。
2. 根据 OSD 的调配，移除失败或不需要的 Ceph OSD。
   请参阅：
   - 在 动态置备的 Red Hat OpenShift Data Foundation 中删除失败的或不需要的 Ceph OSD 。
   - 使用本地存储设备移除失败的或不需要的 Ceph OSD。

如果使用本地磁盘，您可以在删除旧 OSD 后重复使用这些磁盘。

8.1. 验证 CEPH 集群是否健康

存储健康状况在 **Block** 和 **File** 和 **Object** 仪表板上可见。

流程

1. 在 OpenShift Web 控制台中，点 **Storage → Data Foundation**。
2. 在 **Overview** 选项卡的 **Status** 卡中，点 **Storage System**，然后点弹出框中的存储系统链接。
3. 在 **Block and File** 选项卡的 **Status** 卡中，验证 *Storage Cluster* 是否具有绿色勾号。
4. 在 **Details** 卡中，验证是否显示集群信息。

8.2. 在动态置备的 RED HAT OPENSHIFT DATA FOUNDATION 中删除失败的或不需要的 CEPH OSD

按照流程中的步骤，在动态置备的 Red Hat OpenShift Data Foundation 中删除失败或不需要的 Ceph OSD。

重要

只有红帽支持团队才支持缩减集群。

警告

- 当 Ceph 组件没有处于健康状态时，删除 OSD 可能会导致数据丢失。
- 同时删除两个或多个 OSD 会导致数据丢失。

先决条件

- 检查 Ceph 是否健康。如需更多信息，请参阅验证 Ceph 集群是否健康。
- 确保没有触发警报，或者所有重建过程都在进行中。

流程

1. 缩减 OSD 部署。

```
# oc scale deployment rook-ceph-osd-<osd-id> --replicas=0
```

2. 获取要删除的 Ceph OSD 的 **osd-prepare** pod。

```
# oc get deployment rook-ceph-osd-<osd-id> -oyaml | grep ceph.rook.io/pvc
```

3. 删除 **osd-prepare** pod。

```
# oc delete -n openshift-storage pod rook-ceph-osd-prepare-<pvc-from-above-command>-<pod-suffix>
```

4. 从集群移除出现故障的 OSD。

```
# failed_osd_id=<osd-id>

# oc process -n openshift-storage ocs-osd-removal -p FAILED_OSD_IDS=$<failed_osd_id> | oc create -f -
```

其中，**FAILED_OSD_ID** 是 pod 名称中紧接在 **rook-ceph-osd** 前缀后面的整数。

5. 通过检查日志来验证 OSD 是否已成功移除。

```
# oc logs -n openshift-storage ocs-osd-removal-$<failed_osd_id>-<pod-suffix>
```

6. 可选：如果您遇到 **cephosd:osd.0 is not ok to destroy to destroy to destroy** from the **ocs-osd-removal-job** pod in OpenShift Container Platform 的错误，请参阅 对 **cephosd:osd.0** 错误进行故障排除，同时删除失败或不需要的 Ceph OSD。

7. 删除 OSD 部署。

```
# oc delete deployment rook-ceph-osd-<osd-id>
```

验证步骤

- 要检查 OSD 是否已成功删除，请运行：

```
# oc get pod -n openshift-storage ocs-osd-removal-$<failed_osd_id>-<pod-suffix>
```

此命令必须将状态返回为 **Completed**。

8.3. 使用本地存储设备移除失败的或不需要的 CEPH OSD

您可以按照以下步骤删除使用本地存储设备置备的失败或不需要的 Ceph。

重要

只有红帽支持团队才支持缩减集群。

警告

- 当 Ceph 组件没有处于健康状态时，删除 OSD 可能会导致数据丢失。
- 同时删除两个或多个 OSD 会导致数据丢失。

先决条件

- 检查 Ceph 是否健康。如需更多信息，请参阅验证 Ceph 集群是否健康。
- 确保没有触发警报，或者所有重建过程都在进行中。

流程

1. 总之，通过将 OSD 部署上的副本扩展到 0 来标记 OSD 停机。如果 OSD 已因为失败而停机，您可以跳过这一步。

```
# oc scale deployment rook-ceph-osd-<osd-id> --replicas=0
```

2. 从集群移除出现故障的 OSD。

```
# failed_osd_id=<osd_id>

# oc process -n openshift-storage ocs-osd-removal -p FAILED_OSD_IDS=$<failed_osd_id> |
oc create -f -
```

其中，**FAILED_OSD_ID** 是 pod 名称中紧接在 **rook-ceph-osd** 前缀后面的整数。

3. 通过检查日志来验证 OSD 是否已成功移除。

```
# oc logs -n openshift-storage ocs-osd-removal-$<failed_osd_id>-<pod-suffix>
```

4. 可选：如果您遇到 **cephosd:osd.0 is not ok to destroy to destroy to destroy** from the **ocs-osd-removal-job** pod in OpenShift Container Platform 的错误，请参阅 对 **cephosd:osd.0** 错误进行故障排除，同时删除失败或不需要的 Ceph OSD。

5. 删除与故障 OSD 关联的持久性卷声明(PVC)资源。

   a. 获取与故障 OSD 关联的 **PVC**。

```
# oc get -n openshift-storage -o yaml deployment rook-ceph-osd-<osd-id> | grep
ceph.rook.io/pvc
```

   b. 获取与 **PVC 关联的持久性卷** (PV)。

```
# oc get -n openshift-storage pvc <pvc-name>
```

c. 获取失败的设备名称。

```
# oc get pv <pv-name-from-above-command> -oyaml | grep path
```

d. 获取与故障 OSD 关联的 **prepare-pod**。

```
# oc describe -n openshift-storage pvc ocs-deviceset-0-0-nvs68 | grep Mounted
```

e. 在删除关联的 PVC 前，删除 **osd-prepare pod**。

```
# oc delete -n openshift-storage pod <osd-prepare-pod-from-above-command>
```

f. 删除与故障 OSD 关联的 **PVC**。

```
# oc delete -n openshift-storage pvc <pvc-name-from-step-a>
```

6. 从 **LocalVolume 自定义资源** (CR)中删除失败的设备条目。

a. 使用失败设备登录到节点。

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

b. 为失败的设备名称记录 /dev/disk/by-id/<id>。

```
# ls -alh /mnt/local-storage/localblock/
```

7. 可选：如果是，Local Storage Operator 用于置备 OSD，使用 {osd-id} 登录机器并删除设备符号链接。

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

a. 获取故障设备名称的 OSD 符号链接。

```
# ls -alh /mnt/local-storage/localblock
```

b. 删除 符号链接。

```
# rm /mnt/local-storage/localblock/<failed-device-name>
```

8. 删除与 OSD 关联的 PV。

```
# oc delete pv <pv-name>
```

验证步骤

- 要检查 OSD 是否已成功删除，请运行：

```
#oc get pod -n openshift-storage ocs-osd-removal-$<failed_osd_id>-<pod-suffix>
```

此命令必须将状态返回为 **Completed**。

8.4. 对 CEPHOSD:OSD.0 错误进行故障排除，在删除失败或不需要的 CEPH OSD 时无法销毁

如果您收到 **cephosd:osd.0 is NOT ok to destroy** to destroy from the **ocs-osd-removal-job** pod in OpenShift Container Platform 的错误，使用 **FORCE_OSD_REMOVAL** 选项运行 OSD 移除作业，以将 OSD 移到销毁状态。

```
# oc process -n openshift-storage ocs-osd-removal -p FORCE_OSD_REMOVAL=true -p
FAILED_OSD_IDS=$<failed_osd_id> | oc create -f -
```

注意

只有在所有 PG 都处于 active 状态时，才必须使用 **FORCE_OSD_REMOVAL** 选项。如果没有，PG 必须完成回填或进一步调查，以确保它们处于活动状态。

第 9 章 卸载过程中的故障排除和删除剩余的资源

有时，由 Operator 管理的一些自定义资源可能会处于 "Terminating" 状态，等待终结器完成，尽管您执行了所有必要的清理任务。在这种情况下，您需要强制删除这些资源。如果不这样做，资源仍会处于"Terminating"状态，即使您执行了所有卸载步骤。

1. 检查 openshift-storage 命名空间在删除时是否处于 Terminating 状态。

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

输出：

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

2. 在命令输出的 **STATUS** 部分检查 **NamespaceFinalizersRemaining** 和 **NamespaceContentRemaining** 信息，并对列出的每个资源执行下一步。

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

输出示例：

```
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. 删除上一步中列出的所有剩余资源。

对于要删除的每个资源，请执行以下操作：

a. 获取需要删除的资源的对象类型。查看以上输出中的消息。
示例：

Message: 命名空间中的一些内容使终结器为 cephobjectstoreuser.ceph.rook.io

其中 **cephobjectstoreuser.ceph.rook.io** 是对象类型。

b. 获取与对象类型对应的对象名称。

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

示例：

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

输出示例：

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

c. 修补资源。

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

例如：

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

输出：

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

4. 验证 openshift-storage 项目是否已删除。

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

输出：

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

如果问题仍然存在，请联系红帽支持团队。

第 10 章 对外部模式的 CEPHFS PVC 创建进行故障排除

如果您已将红帽 Ceph 存储集群从低于 4.1.1 的版本更新为最新版本，且不是全新部署的集群，您必须在红帽 Ceph 存储集群中手动设置 CephFS 池的应用类型，以外部模式启用 CephFS PVC 创建。

1. 检查 CephFS pvc 处于 **Pending** 状态。

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

输出示例：

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

2. 检查 **describe** 输出，以查看相应 pvc 的事件。
预期的错误消息为 **cephfs_metadata/csi.volumes.default/csi.volume.pvc-xxxxxx-xxxx-xxxx-xxxx-xxxxxxx:(1)Operation not permitted）**

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

输出示例：

```
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. 检查 **<cephfs metadata pool name>** (这里是 **cephfs_metadata**) 和 **<cephfs data pool name>** (这里是 **cephfs_data**)。为了运行命令，需要在 Red Hat Ceph Storage 客户端节点中预安装 **jq**。

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

4. 设置 CephFS 池的应用类型。

   - 在 Red Hat Ceph Storage 客户端节点中运行以下命令：

     ```
     # 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. 验证是否应用了设置。

```
 # 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. 再次检查 CephFS PVC 状态。PVC 现在处于 **Bound** 状态。

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

输出示例：

```
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
[...]
```

第 11 章 在 OPENSHIFT DATA FOUNDATION 中恢复 MONITOR POD

如果所有三个 Pod 都停机，并且 OpenShift Data Foundation 无法自动恢复 monitor pod，则恢复 monitor pod。

流程

1. 缩减 **rook-ceph-operator** 和 **ocs operator** 部署。

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

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

2. 在 **openshift-storage** 命名空间中创建所有部署的备份。

```
# 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. 修补 OSD 部署以移除 **livenessProbe** 参数，再以 命令参数作为 **sleep** 状态运行它。

```
# 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. 从所有 OSD 检索 **monstore** 集群映射。

   a. 创建 **restore_mon.sh** 脚本。

```
#!/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. 运行 **restore_mon.sh** 脚本。

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

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

5. 修补 MON 部署，并使用命令参数作为 **sleep** 状态运行它。

a. 编辑 MON 部署。

```
# 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. 修补 MON 部署，以增加 **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. 将之前检索到的 **monstore** 复制到 **mon-a** pod。

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

7. 导航到 MON 容器集，再更改检索到的 **monstore** 的所有权。

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

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

8. 在重建 **mon db** 之前复制密钥环模板文件。

```
# 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. 从对应的机密中识别所有其他 Ceph 守护进程（MGR、MDS、RGW、Crash、CSI 和 CSI 置备程序）的密钥环。

```
# 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"
```

keyring 文件示例： **/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"
```

重要

- 对于 **client.csi** 相关的密钥环，请参阅前面的密钥环文件输出，并在从其相应的 OpenShift Data Foundation secret 获取密钥后添加默认 **大写字母**。
- OSD 密钥环会在恢复后自动添加。

10. 进入 **mon-a** pod，验证 **monstore** 具有 **monmap**。

    a. 进入到 **mon-a** 容器集。

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

    b. 验证 **monstore** 有 **monmap**。

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

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

11. 可选：如果没有 **monmap**，则创建新的 **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>

    是 **mon-a** pod 的 ID。

    <mon-a-ip>

    是 **mon-a** pod 的 IP 地址。

    <mon-b-id>

    是 **mon-b** pod 的 ID。

    <mon-b-ip>

    是 **mon-b** pod 的 IP 地址。

    <mon-c-id>

    是 **mon-c** pod 的 ID。

    <mon-c-ip>

    是 **mon-c** pod 的 IP 地址。

    <fsid>

    是文件系统 ID。

12. 验证 **monmap**。

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

13. 导入 **monmap**。

重要

使用之前创建的 **keyring** 文件。

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

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

14. 创建旧 **store.db** 文件的备份。

```
# 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. 将重新构建 **store.db** 文件复制到 **monstore** 目录。

```
# 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. 在重建了 **monstore** 目录后，将 **store.db** 文件从本地 复制到 MON 容器集的其余部分。

```
# 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>

是 MON Pod 的 ID

17. 前往 MON 容器集的其余部分，再更改复制的 **monstore** 的所有权。

```
# 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>

是 MON Pod 的 ID

18. 恢复补丁的更改。

- 对于 MON 部署：

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

-

<mon-deployment.yaml>

是 MON 部署 yaml 文件

- 对于 OSD 部署：

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

<osd-deployment.yaml>

是 OSD 部署 yaml 文件

- 对于 MGR 部署：

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

<mgr-deployment.yaml>

是 MGR 部署 yaml 文件

重要

确保 MON、MGR 和 OSD 容器集已启动并在运行。

19. 扩展 **rook-ceph-operator** 和 **ocs-operator** 部署。

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

验证步骤

1. 检查 Ceph 状态，以确认 CephFS 正在运行。

```
# ceph -s
```

输出示例：

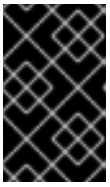
```
cluster:
  id:     f111402f-84d1-4e06-9fdb-c27607676e55
  health: HEALTH_ERR
          1 filesystem is offline
          1 filesystem is online with fewer MDS than max_mds
          3 daemons have recently crashed

  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
```

2. 检查 Multicloud 对象网关(MCG)状态。它应该处于活跃状态，后备存储和 bucketclass 应为 **Ready** 状态。

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



重要

如果 MCG 不在活跃状态，且后备存储和存储桶类没有处于 **Ready** 状态，则需要重启所有 MCG 相关 pod。如需更多信息，请参阅 第 11.1 节 “恢复 Multicloud 对象网关”。

11.1. 恢复 MULTICLOUD 对象网关

如果 Multicloud Object Gateway(MCG)没有处于活跃状态，且后备store 和 bucketclass 不在 **Ready** 状态，您需要重启所有 MCG 相关 pod，并检查 MCG 状态以确认 MCG 是否恢复并正在运行。

流程

1. 重启与 MCG 相关的所有 pod。

```
# 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>

是 MCG operator 的名称

<noobaa-core>

是 MCG 内核 pod 的名称

<noobaa-endpoint>

是 MCG 端点的名称

<noobaa-db>

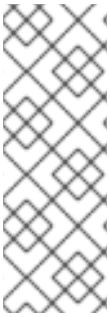
是 MCG db pod 的名称

2. 如果配置了 RADOS 对象网关(RGW)，请重新启动容器集。

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

<rgw-pod>

是 RGW pod 的名称



注意

在 OpenShift Container Platform 4.11 中，在恢复后，RBD PVC 无法在应用程序 pod 上挂载。因此，您需要重启托管应用容器集的节点。要获取托管应用程序 pod 的节点名称，请运行以下命令：

```
# oc get pods <application-pod> -n <namespace> -o yaml | grep nodeName
  nodeName: node_name
```

第 12 章 在 OPENSHIFT DATA FOUNDATION 中恢复 CEPH-MONITOR 仲裁

在某些情况下，**ceph-mons** 可能会丢失仲裁。如果 **mons** 无法再次形成仲裁，则需要一个手动过程来再次进入仲裁。唯一的要求是，至少有一个 **mon** 必须健康。以下步骤从仲裁中删除不健康状态的 **mons**，并可让您使用单个 **mon** 重新组成仲裁，然后将仲裁回到原始大小。

例如，如果您有三个 **mons** 并失去了仲裁，您需要从仲裁中删除两个有问题的 **mons**，通知可以正常工作的 **mon** 它是仲裁中唯一的 **mon**，然后重启这个可以正常工作的 **mon**。

流程

1. 停止 **rook-ceph-operator**，以便在修改 **monmap** 时不通过 **mons** 失败。

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

2. 注入一个新的 **monmap**。

警告

您必须非常仔细注入 **monmap**。如果运行不正确，您的集群可以被永久销毁。Ceph **monmap** 来跟踪 **mon** 仲裁。**monmap** 被更新为仅包含健康的 mon。在本例中，健康的 mon 是 **rook-ceph-mon-b**，而不健康的 **mons** 为 **rook-ceph-mon-a** 和 **rook-ceph-mon-c**。

a. 备份当前的 **rook-ceph-mon-b** 部署：

```
# oc -n openshift-storage get deployment rook-ceph-mon-b -o yaml > rook-ceph-mon-b-deployment.yaml
```

b. 打开 YAML 文件，并从 **mon** 容器复制**命令**和**参数**（请参见以下示例中的容器列表）。这是 **monmap** 更改所需要的。

```
[...]
  containers:
  - args:
    - --fsid=41a537f2-f282-428e-989f-a9e07be32e47
    - --keyring=/etc/ceph/keyring-store/keyring
    - --log-to-stderr=true
    - --err-to-stderr=true
    - --mon-cluster-log-to-stderr=true
    - '--log-stderr-prefix=debug '
    - --default-log-to-file=false
    - --default-mon-cluster-log-to-file=false
    - --mon-host=$(ROOK_CEPH_MON_HOST)
    - --mon-initial-members=$(ROOK_CEPH_MON_INITIAL_MEMBERS)
    - --id=b
    - --setuser=ceph
    - --setgroup=ceph
```

```
    - --foreground
    - --public-addr=10.100.13.242
    - --setuser-match-path=/var/lib/ceph/mon/ceph-b/store.db
    - --public-bind-addr=$(ROOK_POD_IP)
    command:
    - ceph-mon
[...]
```

c. 清理复制的 **command** 和 **args** 字段以形成过去的命令，如下所示：

```
# ceph-mon \
    --fsid=41a537f2-f282-428e-989f-a9e07be32e47 \
    --keyring=/etc/ceph/keyring-store/keyring \
    --log-to-stderr=true \
    --err-to-stderr=true \
    --mon-cluster-log-to-stderr=true \
    --log-stderr-prefix=debug \
    --default-log-to-file=false \
    --default-mon-cluster-log-to-file=false \
    --mon-host=$ROOK_CEPH_MON_HOST \
    --mon-initial-members=$ROOK_CEPH_MON_INITIAL_MEMBERS \
    --id=b \
    --setuser=ceph \
    --setgroup=ceph \
    --foreground \
    --public-addr=10.100.13.242 \
    --setuser-match-path=/var/lib/ceph/mon/ceph-b/store.db \
    --public-bind-addr=$ROOK_POD_IP
```

注意

确保删除括起了 **--log-stderr-prefix** 标记的单引号，以及包括 **ROOK_CEPH_MON_MON_MON_MON_HOST**、**ROOK_CEPH_MON_MON_CEPH_MON_INITIAL_MEMBERS** 和 **ROOK_POD_IP** 变量的括号。

d. 修补 **rook-ceph-mon-b** 部署，在不删除 **mon** 的情况下停止这个 **mon** 工作。

```
# oc -n openshift-storage patch deployment rook-ceph-mon-b  --type='json' -p
'[{"op":"remove", "path":"/spec/template/spec/containers/0/livenessProbe"}]'

# oc -n openshift-storage patch deployment rook-ceph-mon-b -p '{"spec": {"template":
{"spec": {"containers": [{"name": "mon", "command": ["sleep", "infinity"], "args": []}]}}}}'
```

e. 在 **mon-b** pod 上执行以下步骤：

i. 连接到健康 **mon** 的 pod 并运行以下命令：

```
# oc -n openshift-storage exec -it <mon-pod> bash
```

ii. 设置变量。

```
# monmap_path=/tmp/monmap
```

iii. 将 **monmap** 提取到一个文件，从健康的 **mon** 部署中粘贴 ceph **mon** 命令并添加 **--extract-monmap=${monmap_path}** 标记。

```
# ceph-mon \
    --fsid=41a537f2-f282-428e-989f-a9e07be32e47 \
    --keyring=/etc/ceph/keyring-store/keyring \
    --log-to-stderr=true \
    --err-to-stderr=true \
    --mon-cluster-log-to-stderr=true \
    --log-stderr-prefix=debug \
    --default-log-to-file=false \
    --default-mon-cluster-log-to-file=false \
    --mon-host=$ROOK_CEPH_MON_HOST \
    --mon-initial-members=$ROOK_CEPH_MON_INITIAL_MEMBERS \
    --id=b \
    --setuser=ceph \
    --setgroup=ceph \
    --foreground \
    --public-addr=10.100.13.242 \
    --setuser-match-path=/var/lib/ceph/mon/ceph-b/store.db \
    --public-bind-addr=$ROOK_POD_IP \
    --extract-monmap=${monmap_path}
```

iv. 检查 **monmap** 的内容。

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

v. 从 **monmap** 中删除错误的 **mons**。

```
# monmaptool ${monmap_path} --rm <bad_mon>
```

在本例中，我们移除了 **mon0** 和 **mon2** ：

```
# monmaptool ${monmap_path} --rm a
# monmaptool ${monmap_path} --rm c
```

vi. 把修改过的 **monmap** 注入到健康的 **mon**中，粘贴 ceph **mon** 命令并添加 **--inject-monmap=${monmap_path}** 标记：

```
# ceph-mon \
    --fsid=41a537f2-f282-428e-989f-a9e07be32e47 \
    --keyring=/etc/ceph/keyring-store/keyring \
    --log-to-stderr=true \
    --err-to-stderr=true \
    --mon-cluster-log-to-stderr=true \
    --log-stderr-prefix=debug \
    --default-log-to-file=false \
    --default-mon-cluster-log-to-file=false \
    --mon-host=$ROOK_CEPH_MON_HOST \
    --mon-initial-members=$ROOK_CEPH_MON_INITIAL_MEMBERS \
    --id=b \
    --setuser=ceph \
    --setgroup=ceph \
    --foreground \
```

```
--public-addr=10.100.13.242 \
--setuser-match-path=/var/lib/ceph/mon/ceph-b/store.db \
--public-bind-addr=$ROOK_POD_IP \
--inject-monmap=${monmap_path}
```

vii. 退出 shell 以继续。

3. 编辑 Rook **configmaps**。

a. 编辑 operator 用来跟踪 **mons** 的 **configmap**。

```
# oc -n openshift-storage edit configmap rook-ceph-mon-endpoints
```

b. 验证在数据元素中，您可以看到如下三个 **mon** （具体取决于您的 **moncount**，可能会更多）：

```
data: a=10.100.35.200:6789;b=10.100.13.242:6789;c=10.100.35.12:6789
```

c. 从列表中删除有问题的 **mons**，以使用一个好的 **mon** 结束。例如：

```
data: b=10.100.13.242:6789
```

d. 保存文件并退出。

e. 现在，您需要使用用于 **mons** 和其他组件的 **Secret**。

i. 为变量 **good_mon_id** 设置一个值。
例如：

```
# good_mon_id=b
```

ii. 您可以使用 **oc patch** 命令来修补 **rook-ceph-config** secret，并更新两个键/值对 **mon_host** 和 **mon_initial_members**。

```
# mon_host=$(oc -n openshift-storage get svc rook-ceph-mon-b -o
jsonpath='{.spec.clusterIP}')

# oc -n openshift-storage patch secret rook-ceph-config -p '{"stringData":
{"mon_host": "[v2:"${mon_host}":3300,v1:"${mon_host}":6789]",
"mon_initial_members": """${good_mon_id}"""}}'
```

注意

如果使用 **hostNetwork: true**，则需要将 **mon_host** 变量替换为代表 **mon** 固定到的节点 IP(**nodeSelector**)。这是因为在那个 "mode" 中创建了 **rook-ceph-mon-*** 服务。

4. 重新启动 **mon**。
您需要使用原始 **ceph-mon** 命令重启好的 **mon** pod，以获取这些更改。

a. 在 **mon** 部署 YAML 文件的备份中使用 **oc replace** 命令：

```
# oc replace --force -f rook-ceph-mon-b-deployment.yaml
```

注意

选项 **--force** 删除部署并创建新部署。

b. 验证集群的状态。
状态应该在仲裁中显示 **mon**。如果状态正常，您的集群应该再次处于健康状态。

5. 删除不再预期在仲裁中的两个 mon 部署。
例如：

```
# oc delete deploy <rook-ceph-mon-1>
# oc delete deploy <rook-ceph-mon-2>
```

在本例中，要删除的部署有 **rook-ceph-mon-a** 和 **rook-ceph-mon-c**。

6. 重启 Operator。

a. 再次启动 rook 运算符，以恢复监控集群的健康状况。

注意

忽略多个资源已存在的错误是安全的。

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

根据 **mon** 数量，Operator 会自动添加更多 **mons** 来再次增加仲裁大小。

第 13 章 启用 RED HAT OPENSHIFT DATA FOUNDATION 控制台插件

如果在安装 OpenShift Data Foundation Operator 后不自动启用 console 插件选项。console 插件提供一个包含在 Web 控制台中的自定义接口。您可以从图形用户界面(GUI)或命令行界面启用 console 插件选项。

先决条件

- 您有管理访问权限来访问 OpenShift Web 控制台。
- OpenShift Data Foundation Operator 在 **openshift-storage** 命名空间上安装并运行。

流程

从用户界面

1. 在 OpenShift Web 控制台中，点 **Operators → Installed Operators** 查看所有已安装的 Operator。
2. 确保所选 **项目** 为 **openshift-storage**。
3. 单击 **OpenShift Data Foundation** operator。
4. 启用 console 插件选项。
   a. 在 **Details** 选项卡中，单击 **Console 插件** 下的铅笔图标。
   b. 选择 **Enable**，然后单击 **Save**。

使用命令行界面

- 执行以下命令启用 console 插件选项：

```
$ oc patch console.operator cluster -n openshift-storage --type json -p '[{"op": "add", "path": "/spec/plugins", "value": ["odf-console"]}]'
```

验证步骤

- 启用 console 插件选项后，显示一条带有消息的弹出窗口，**Web 控制台更新**会出现在 GUI 中。点这个弹出窗口中的 **Refresh web console** 来反映控制台的更改。
  - 在 Web 控制台中，导航到 **Storage** 并验证 **Data Foundation** 是否可用。

第 14 章 更改 OPENSHIFT DATA FOUNDATION 组件的资源

安装 OpenShift Data Foundation 时，它附带了 OpenShift Data Foundation Pod 可消耗的预定义资源。在某些情况下，可能需要提高 I/O 负载。

- 要更改 rook-ceph pod 上的 CPU 和内存资源，请参阅 第 14.1 节 “更改 rook-ceph pod 上的 CPU 和内存资源”。
- 要调整 Multicloud 对象网关(MCG)的资源，请参阅 第 14.2 节 “为 MCG 调整资源”。

14.1. 更改 ROOK-CEPH POD 上的 CPU 和内存资源

安装 OpenShift Data Foundation 时，它附带了 rook-ceph Pod 的预定义 CPU 和内存资源。您可以根据要求手动增加这些值。

您可以更改以下 pod 中的 CPU 和内存资源：

- **mgr**
- **mds**
- **RGW**

以下示例演示了如何更改 rook-ceph Pod 上的 CPU 和内存资源。在本例中，**cpu** 和 **memory** 的现有 MDS pod 值会分别从 **1** 和 **4Gi** 增加到 **2** 和 **8Gi**。

1. 编辑存储集群：

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

<storagecluster_name>
指定存储集群的名称。
例如：

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

2. 将下面几行添加到存储集群自定义资源(CR)中：

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

3. 保存更改并退出编辑器。
4. 或者，运行 **oc patch** 命令更改 **mds** pod 的 CPU 和内存值：

```
# 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>

指定存储集群的名称。
例如：

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

14.2. 为 MCG 调整资源

Multicloud 对象网关(MCG)的默认配置针对低资源消耗和不性能进行了优化。有关如何调整 MCG 资源的更多信息，请参阅 用于多云对象网关(NooBaa)的红帽知识库解决方案性能调整指南。

第 15 章 使用 OVS-MULTITENANT 插件访问 ODF-CONSOLE，方法是手动启用全局 POD 网络

在 OpenShift Container Platform 中，当 **ovs-multitenant** 插件用于软件定义型网络 (SDN) 时，来自不同项目的 pod 无法将数据包发送到不同项目的 pod 和服务的数据包。默认情况下，pod 无法在命名空间或项目之间进行通信，因为项目的 pod 网络不是全局的。

要访问 odf-console，**openshift-console** 命名空间中的 OpenShift 控制台 pod 需要与 **openshift-storage** 命名空间中的 OpenShift Data Foundation odf-console 连接。这只有在手动模拟全局 pod 网络时才可能。

问题

- 当 OpenShift Container Platform 中使用 `ovs-multitenant` 插件时，odf-console 插件会失败，并显示以下信息：

```
GET request for "odf-console" plugin failed: Get "https://odf-console-service.openshift-storage.svc.cluster.local:9001/locales/en/plugin__odf-console.json": context deadline exceeded (Client.Timeout exceeded while awaiting headers)
```

分辨率

- 使 OpenShift Data Foundation 项目的 pod 网络全局化：

```
$ oc adm pod-network make-projects-global openshift-storage
```