Red Hat Ceph Storage 7

操作指南

Red Hat Ceph Storage 的操作任务

Last Updated: 2024-06-14

Red Hat Ceph Storage 7 操作指南

Red Hat Ceph Storage 的操作任务

法律通告

摘要

本文档论述了如何为 Red Hat Ceph Storage 执行操作任务。 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 信息

目录

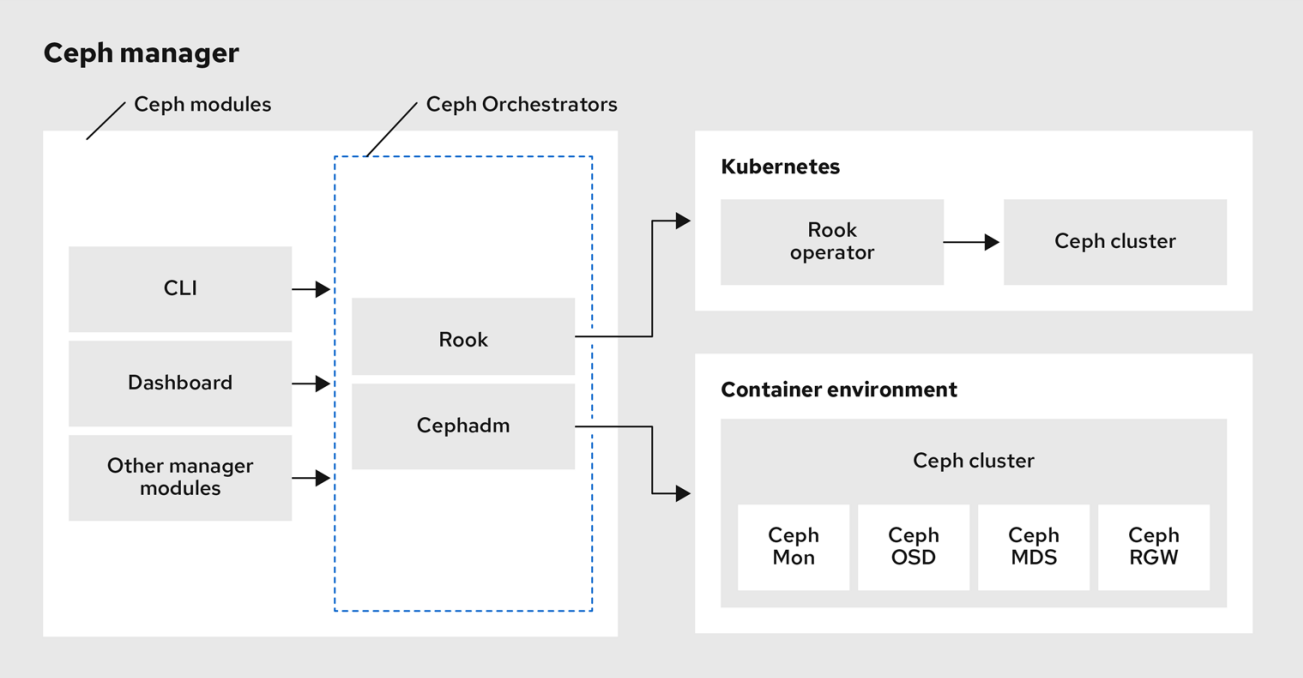
第 1 章 CEPH 编排器简介

作为存储管理员，您可以将 Ceph 编排器与 Cephadm 实用程序搭配使用，能够发现设备并在 Red Hat Ceph Storage 集群中创建服务。

1.1. 使用 CEPH ORCHESTRATOR

Red Hat Ceph Storage Orchestrators 是经理模块，主要充当 Red Hat Ceph Storage 集群和部署工具（如 Rook 和 Cephadm）以实现统一体验。它们也与 Ceph 命令行界面和 Ceph 控制面板集成。

以下是 Ceph Orchestrator 的工作流图：



153_Ceph_0421

Red Hat Ceph Storage Orchestrators 类型

Red Hat Ceph Storage Orchestrators 有三个主要类型：

- **编配器 CLI** ：它们是 Orchestrators 中使用的常用 API，它包括一组可以实施的命令。这些 API 还提供通用命令行界面(CLI)，以使用外部编排服务编排 **ceph-mgr** 模块。以下是用于 Ceph Orchestrator 的命名法：
  - 主机：这是物理主机的主机名，而不是容器内的 pod 名称、DNS 名称、容器名称或主机名。
  - 服务类型 ：服务的类型，如 nfs、mds、osd、mon、rgw 和 mgr。
  - 服务：由 Ceph 存储集群提供的功能服务，如监控服务、管理器服务、OSD 服务、Ceph Object Gateway 服务和 NFS 服务。
  - 守护进程：由一个或多个主机（如 Ceph 对象网关服务）部署的服务的特定实例，可在三个不同的主机上运行不同的 Ceph 对象网关守护进程。
- **Cephadm Orchestrator** - 这是一个 Ceph Orchestrator 模块，它不依赖于外部工具，如 Rook 或 Ansible，而是通过建立 SSH 连接并发出显式管理命令来管理集群中的节点。此模块适用于第一天和第二天操作。

使用 Cephadm Orchestrator 是在不利用 Ansible 等部署框架的情况下安装 Ceph 存储集群的建议方法。其理念是，为管理器守护进程提供对 SSH 配置和密钥的访问，这些密钥可以连接到集群中的所有节点，以执行任何管理操作，如创建存储设备清单、部署和替换 OSD 或启动和停止 Ceph 守护进程。此外，Cephadm Orchestrator 将部署由 **systemd** 管理的容器镜像，以允许独立升级共同放置服务。

此编配器还突出显示一个工具，它封装了所有必要的操作，以根据当前主机上的服务管理容器镜像部署，包括引导运行 Ceph Monitor 和 Ceph Manager 的最小集群的命令。

- **Rook Orchestrator** - Rook 是一个编配工具，它使用 Kubernetes Rook 操作器来管理在 Kubernetes 集群内运行的 Ceph 存储集群。rook 模块提供 Ceph 的 Orchestrator 框架和 Rook 间的集成。Rook 是 Kubernetes 的一个开源云原生存储 operator。
  Rook 遵循 "operator" 模型，其中在 Kubernetes 中定义的自定义资源定义(CRD)对象用来描述 Ceph 存储集群及其所需状态，而 rook 操作器守护进程在控制循环中运行，并将当前集群状态与所需的状态进行比较，并采取措施来融合。描述 Ceph 所需状态的主要对象是 Ceph 存储集群 CRD，它包括关于 OSD 应该使用哪些设备的信息、应运行多少个 monitor，以及应使用的 Ceph 版本。Rook 定义了几个其他 CRD 来描述 RBD 池、CephFS 文件系统等。

  Rook Orchestrator 模块在 **ceph-mgr** 守护进程中运行并实现 Ceph orchestration API，它更改 Kubernetes 中的 Ceph 存储集群来描述所需的集群状态。Rook 集群的 **ceph-mgr** 守护进程作为 Kubernetes pod 运行，因此 rook 模块可以在没有显式配置的情况下连接到 Kubernetes API。

第 2 章 使用 CEPH ORCHESTRATOR 管理服务

作为存储管理员，在安装 Red Hat Ceph Storage 集群后，您可以使用 Ceph 编排器监控和管理存储集群中的服务。服务是一组配置在一起的守护进程。

本节涵盖以下管理信息：

- *Ceph 编排器的放置规格。*
- *使用命令行界面部署 Ceph 守护进程。*
- *使用命令行界面在主机子集上部署 Ceph 守护进程。*
- *Ceph 编排器的服务规格。*
- *使用服务规格部署 Ceph 守护进程。*
- *使用服务规格部署 Ceph 文件系统镜像守护进程。*

2.1. CEPH 编排器的放置规格

您可以使用 Ceph 编排器部署 **osds**、**mons**、**mgrs**、**mds** 和 **rgw** 服务。红帽建议使用放置规格部署服务。您需要知道要使用 Ceph 编排器部署服务所需的守护进程的位置和数量。放置规格可以作为命令行参数或 **yaml** 文件中的服务规格传递。

使用放置规格部署服务的方法有两种：

- 直接在命令行界面中使用放置规格。例如，如果要在主机上部署三个监视器，请运行以下命令在 **host01**、**host02** 和 **host03** 上部署三个 monitor。

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
  ```

- 使用 YAML 文件中的放置规格。例如，如果您要在所有主机上部署 **node-exporter**，您可以在 **yaml** 文件中指定以下内容：

  示例

  ```
  service_type: node-exporter
  placement:
  host_pattern: '*'
  extra_entrypoint_args:
    - "--collector.textfile.directory=/var/lib/node_exporter/textfile_collector2"
  ```

2.2. 使用命令行界面部署 CEPH 守护进程

通过使用 Ceph 编排器，您可以使用 **ceph orch** 命令部署诸如 Ceph Manager、Ceph 监控器、Ceph OSD、监控堆栈等守护进程。放置规格通过 Orchestrator 命令传递为 **--placement** 参数。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- 主机添加到存储集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 使用以下方法之一在主机上部署守护进程：

   - **方法 1：** 指定守护进程的数量和主机名：

     语法

     ```
     ceph orch apply SERVICE_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
     ```

     示例

     ```
     [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
     ```

   - **方法 2：** 向主机添加标签，然后使用标签部署守护进程：

     a. 向主机添加标签：

     语法

     ```
     ceph orch host label add HOSTNAME_1 LABEL
     ```

     示例

     ```
     [ceph: root@host01 /]# ceph orch host label add host01 mon
     ```

     b. 使用标签部署守护进程：

     语法

     ```
     ceph orch apply DAEMON_NAME label:LABEL
     ```

     示例

     ```
     ceph orch apply mon label:mon
     ```

   - **方法 3**：向主机添加标签并使用 **--placement** 参数进行部署：

     a. 向主机添加标签：

     语法

     ```
     ceph orch host label add HOSTNAME_1 LABEL
     ```

示例

```
[ceph: root@host01 /]# ceph orch host label add host01 mon
```

b. 使用标签放置规格部署守护进程：

语法

```
ceph orch apply DAEMON_NAME --placement="label:LABEL"
```

示例

```
ceph orch apply mon --placement="label:mon"
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ceph orch ps --service_name=SERVICE_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  [ceph: root@host01 /]# ceph orch ps --service_name=mon
  ```

其它资源

- 请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 添加主机*。

2.3. 使用命令行界面在主机子集上部署 CEPH 守护进程

您可以使用 **--placement** 选项在主机的子集上部署守护进程。您可以使用要部署的守护进程的名称来指定放置规格中的守护进程数量。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出您要在其上部署 Ceph 守护进程的主机：

   示例

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

3. 部署守护进程：

   语法

   ```
   ceph orch apply SERVICE_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1 _HOST_NAME_2 HOST_NAME_3"
   ```

   示例

   ```
   ceph orch apply mgr --placement="2 host01 host02 host03"
   ```

   在本例中，**mgr** 守护进程仅部署到两个主机上。

验证

- 列出主机：

  示例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

其它资源

- 请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 列出主机*。

2.4. CEPH 编排器的服务规格

服务规格是一个数据结构，它指定用于部署 Ceph 服务的服务属性和配置设置。以下是多文档 YAML 文件 **cluster.yaml** 的示例，用于指定服务规格：

示例

```
service_type: mon
placement:
  host_pattern: "mon*"
---
service_type: mgr
placement:
  host_pattern: "mgr*"
---
```

```
service_type: osd
service_id: default_drive_group
placement:
  host_pattern: "osd*"
data_devices:
  all: true
```

以下列表定义了服务规格的属性参数，如下所示：

- **service_type** ：服务的类型：
  - Ceph 服务，如 mon、crash、mds、mgr、osd、rbd 或 rbd-mirror。
  - Ceph 网关，如 nfs 或 rgw。
  - 监控堆栈，如 Alertmanager、Prometheus、Grafana 或 Node-exporter。
  - 用于自定义容器的容器。
- **service_id** ：服务的唯一名称。
- **placement** ：用于定义部署守护进程的位置和方式。
- **unmanaged** ：如果设置为 **true**，则 Orchestrator 将无法部署或删除与该服务关联的任何守护进程。

Orchestrators 的无状态服务

无状态服务是一种不需要状态信息的服务。例如，要启动 **rgw** 服务，不需要额外的信息才能启动或运行该服务。**rgw** 服务不创建有关此状态的信息，从而提供相应的功能。无论 **rgw** 服务何时启动，其状态都是相同的。

2.5. 禁用自动管理守护进程

您可以将 Cephadm 服务标记为 **受管** 或 **非受管**，而无需编辑和重新应用服务规格。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。

流程

- 使用以下命令为服务设置 **非受管** ：

  语法

  ```
  ceph orch set-unmanaged SERVICE_NAME
  ```

  示例

  ```
  [root@host01 ~]# ceph orch set-unmanaged grafana
  ```

- 使用以下命令为服务设置 **受管** ：

语法

```
ceph orch set-managed SERVICE_NAME
```

示例

```
[root@host01 ~]# ceph orch set-managed mon
```

2.6. 使用服务规格部署 CEPH 守护进程

通过 Ceph 编排器，您可以使用 YAML 文件中的服务规格部署守护进程，如 ceph Manager、Ceph 监控、Ceph OSD、监控堆栈等。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。

流程

1. 创建 **yaml** 文件：

   示例

   ```
   [root@host01 ~]# touch mon.yaml
   ```

2. 此文件可以通过两种不同的方式进行配置：

   - 编辑该文件，在放置规格中包含主机详情：

     语法

     ```
     service_type: SERVICE_NAME
     placement:
       hosts:
         - HOST_NAME_1
         - HOST_NAME_2
     ```

     示例

     ```
     service_type: mon
     placement:
       hosts:
         - host01
         - host02
         - host03
     ```

   - 编辑该文件，在放置规格中包含标签详情：

     语法

```
service_type: SERVICE_NAME
placement:
  label: "LABEL_1"
```

示例

```
service_type: mon
placement:
  label: "mon"
```

3. 可选：在部署服务时，您还可以在服务规格文件中使用额外的容器参数，如 CPU、CA 证书和其他文件：

示例

```
extra_container_args:
  - "-v"
  - "/etc/pki/ca-trust/extracted:/etc/pki/ca-trust/extracted:ro"
  - "--security-opt"
  - "label=disable"
  - "cpus=2"
  - "--collector.textfile.directory=/var/lib/node_exporter/textfile_collector2"
```

注意

Red Hat Ceph Storage 支持使用额外的参数来启用 Cephadm 部署的 node-exporter 中的额外指标。

4. 将 YAML 文件挂载到容器中的一个目录下：

示例

```
[root@host01 ~]# cephadm shell --mount mon.yaml:/var/lib/ceph/mon/mon.yaml
```

5. 进入该目录：

示例

```
[ceph: root@host01 /]# cd /var/lib/ceph/mon/
```

6. 使用服务规格部署 Ceph 守护进程：

语法

```
ceph orch apply -i FILE_NAME.yaml
```

示例

```
[ceph: root@host01 mon]# ceph orch apply -i mon.yaml
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

其它资源

- 请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 列出主机*。

2.7. 使用服务规格部署 CEPH 文件系统镜像守护进程

Ceph 文件系统 (CephFS) 支持使用 CephFS 镜像守护进程 (**cephfs-mirror**) 将快照异步复制到远程 CephFS 文件系统。快照同步将快照数据复制到远程 CephFS，并在远程目标上创建一个新的快照，其名称相同。利用 Ceph 编排器，您可以使用 YAML 文件中的服务规格部署 **cephfs-mirror**。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 已创建 CephFS。

流程

1. 创建 **yaml** 文件：

   示例

   ```
   [root@host01 ~]# touch mirror.yaml
   ```

2. 编辑该文件使其包含以下内容：

   语法

   ```
   service_type: cephfs-mirror
   service_name: SERVICE_NAME
   placement:
     hosts:
   ```

```
    - HOST_NAME_1
    - HOST_NAME_2
    - HOST_NAME_3
```

示例

```
service_type: cephfs-mirror
service_name: cephfs-mirror
placement:
  hosts:
    - host01
    - host02
    - host03
```

3. 将 YAML 文件挂载到容器中的一个目录下：

   示例

   ```
   [root@host01 ~]# cephadm shell --mount mirror.yaml:/var/lib/ceph/mirror.yaml
   ```

4. 进入该目录：

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/
   ```

5. 使用服务规格部署 **cephfs-mirror** 守护进程：

   示例

   ```
   [ceph: root@host01 /]# ceph orch apply -i mirror.yaml
   ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=cephfs-mirror
  ```

其它资源

- 有关 **cephfs-mirror** 守护进程的更多信息，请参阅 *Ceph 文件系统* 镜像。

第 3 章 使用 CEPH ORCHESTRATOR 管理主机

作为存储管理员，您可以在后端中将 Ceph Orchestrator 与 Cephadm 搭配使用，以添加、列出和删除现有 Red Hat Ceph Storage 集群中的主机。

您还可以向主机添加标签。标签是自由格式的，没有具体含义。每一主机可以有多个标签。例如，将 **mon** 标签应用到部署了监控守护进程的所有主机，**mgr** 用于部署有管理器守护进程的 mgr，同时用于 Ceph 对象网关的 **rgw**，等等。

标记存储集群中的所有主机有助于简化系统管理任务，允许您快速识别每个主机上运行的守护进程。此外，您可以使用 Ceph 编配器或 YAML 文件在具有特定主机标签的主机上部署或删除守护进程。

本节涵盖了以下管理任务：

- *使用 Ceph 编排器添加主机。*
- *使用 Ceph Orchestrator 添加多个主机。*
- *使用 Ceph 编排器列出主机。*
- *向主机 添加标签。*
- *从主机中删除标签。*
- *使用 Ceph 编排器删除主机。*
- *使用 Ceph 编排器将主机置于维护模式*。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 新主机的 IP 地址应在 **/etc/hosts** 文件中更新。

3.1. 使用 CEPH ORCHESTRATOR 添加主机

您可以将 Ceph Orchestrator 与后端中的 Cephadm 搭配使用，将主机添加到现有的 Red Hat Ceph Storage 集群中。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对存储集群中所有节点的根级别访问权限。
- 将节点注册到 CDN 并附加订阅。
- 具有 sudo 的 Ansible 用户，对存储集群中所有节点的 **ssh** 访问和免密码访问。

流程

1. 从 Ceph 管理节点，登录 Cephadm shell：

   示例

```
[root@host01 ~]# cephadm shell
```

2. 将集群的公共 SSH 密钥提取到文件夹：

语法

```
ceph cephadm get-pub-key > ~/PATH
```

示例

```
[ceph: root@host01 /]# ceph cephadm get-pub-key > ~/ceph.pub
```

3. 将 Ceph 集群的公共 SSH 密钥复制到新主机上的 root 用户的 **authorized_keys** 文件中：

语法

```
ssh-copy-id -f -i ~/PATH root@HOST_NAME_2
```

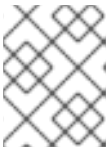
示例

```
[ceph: root@host01 /]# ssh-copy-id -f -i ~/ceph.pub root@host02
```

4. 从 Ansible 管理节点，将新主机添加到 Ansible 清单文件。该文件的默认位置为 **/usr/share/cephadm-ansible/hosts**。以下示例显示了典型的清单文件的结构：

示例

```
host01
host02
host03

[admin]
host00
```



注意

如果您之前已将新主机添加到 Ansible 清单文件，并在主机上运行 preflight playbook，请跳至第 6 步。

5. 使用 **--limit** 选项运行 preflight playbook：

语法

```
ansible-playbook -i INVENTORY_FILE cephadm-preflight.yml --extra-vars "ceph_origin=rhcs"
--limit NEWHOST
```

示例

```
[ceph-admin@admin cephadm-ansible]$ ansible-playbook -i hosts cephadm-preflight.yml --
extra-vars "ceph_origin=rhcs" --limit host02
```

preflight playbook 在新主机上安装 **podman**、**lvm2**、**chronyd** 和 **cephadm**。安装完成后，**cephadm** 驻留在 **/usr/sbin/** 目录中。

6. 从 Ceph 管理节点，登录 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

7. 使用 **cephadm** 编配器将主机添加到存储集群中：

   语法

   ```
   ceph orch host add HOST_NAME IP_ADDRESS_OF_HOST [--
   label=LABEL_NAME_1,LABEL_NAME_2]
   ```

   --label 选项是可选的，这会在添加主机时添加标签。您可以向主机添加多个标签。

   示例

   ```
   [ceph: root@host01 /]# ceph orch host add host02 10.10.128.70 --labels=mon,mgr
   ```

验证

- 列出主机：

  示例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

其它资源

- 请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 列出主机*。
- 有关 **cephadm-preflight** playbook 的更多信息，请参阅 *Red Hat Ceph Storage 安装指南中的 运行 preflight playbook* 部分。
- 请参阅 *Red Hat Ceph Storage 安装指南中的将 Red Hat Ceph Storage 节点注册到 CDN 并附加订阅* 部分。
- 请参阅 *Red Hat Ceph Storage 安装指南中的 创建带有 sudo 访问权限的 Ansible 用户* 部分。

3.2. 使用 CEPH ORCHESTRATOR 添加多个主机

您可以通过 Ceph Orchestrator 使用 YAML 文件格式的服务规格同时将多个主机添加到 Red Hat Ceph Storage 集群。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

流程

1. 创建 **hosts.yaml** 文件：

   示例

   ```
   [root@host01 ~]# touch hosts.yaml
   ```

2. 编辑 **hosts.yaml** 文件使其包含以下详情：

   示例

   ```
   service_type: host
   addr: host01
   hostname: host01
   labels:
   - mon
   - osd
   - mgr
   ---
   service_type: host
   addr: host02
   hostname: host02
   labels:
   - mon
   - osd
   - mgr
   ---
   service_type: host
   addr: host03
   hostname: host03
   labels:
   - mon
   - osd
   ```

3. 将 YAML 文件挂载到容器中的一个目录下：

   示例

   ```
   [root@host01 ~]# cephadm shell --mount hosts.yaml:/var/lib/ceph/hosts.yaml
   ```

4. 进入该目录：

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/
   ```

5. 使用服务规格部署主机：

   语法

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   示例

```
[ceph: root@host01 hosts]# ceph orch apply -i hosts.yaml
```

验证

- 列出主机：

  示例

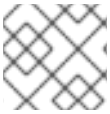
  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

其它资源

- 请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 列出主机*。

3.3. 使用 CEPH ORCHESTRATOR 列出主机

您可以使用 Ceph 编排器列出 Ceph 集群的主机。



注意

主机 *STATUS* 为空，在 **ceph orch host ls** 命令的输出中。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到存储集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出集群的主机：

   示例

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

   您将看到主机的 *STATUS* 是空白的，这是预期的行为。

3.4. 为主机添加标签

使用 Ceph 编排器向主机添加标签。标签可用于指定守护进程的放置。

几个标签示例是 **mgr**、**mon** 和 **osd**，基于主机上部署的服务。每一主机可以有多个标签。

您还可以添加以下主机标签，它们对 **cephadm** 具有特殊含义，它们以 _ 开头：

- **_no_schedule**: 此标签会阻止 **cephadm** 调度或部署主机上的守护进程。如果它被添加到已包含 Ceph 守护进程的现有主机中，它会导致 **cephadm** 在其他位置移动这些守护进程，除了自动移除的 OSD 除外。当添加带有 **_no_schedule** 标签的主机时，不会在其上部署守护进程。当守护进程在删除主机前排空时，在该主机上设置了 **_no_schedule** 标签。

- **_no_autotune_memory** ：此标签不会在主机上自动微调内存。即使为 host 上的一个或多个守护进程启用了 **osd_memory_target_autotune** 选项，也会防止守护进程内存被调优。

- **_admin**: 默认情况下，**_admin** 标签应用于存储集群中的 bootstrapped 主机， **client.admin** 密钥被设置为使用 **ceph orch client-keyring {ls|set|rm}** 功能分发到该主机。将此标签添加到其他主机通常会导致 **cephadm** 在 **/etc/ceph** 目录中部署配置和密钥环文件。

先决条件

- 已安装并引导的存储集群。
- 对存储集群中所有节点的根级别访问权限。
- 主机添加到存储集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 为主机添加标签：

   语法

   ```
   ceph orch host label add HOSTNAME LABEL
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 mon
   ```

验证

- 列出主机：

  示例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

3.5. 从主机中删除标签

您可以使用 Ceph 编配器从主机移除标签。

先决条件

- 已安装并引导的存储集群。
- 对存储集群中所有节点的根级别访问权限。

流程

1. 启动 **cephadm** shell：

```
[root@host01 ~]# cephadm shell
[ceph: root@host01 /]#
```

2. 删除标签。

语法

```
ceph orch host label rm HOSTNAME LABEL
```

示例

```
[ceph: root@host01 /]# ceph orch host label rm host02 mon
```

验证

- 列出主机：

示例

```
[ceph: root@host01 /]# ceph orch host ls
```

3.6. 使用 CEPH ORCHESTRATOR 删除主机

您可以使用 Ceph 编排器删除 Ceph 集群的主机。所有守护进程都会使用 **drain** 选项删除，该选项添加了 **_no_schedule** 标签，以确保您无法部署任何守护进程或集群完成这个操作。

重要

如果您要删除 bootstrap 主机，请确保在删除主机前将 admin 密钥环和配置文件复制到存储集群中的另一主机上。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到存储集群中。
- 部署所有服务。
- Cephadm 部署在必须移除服务的节点上。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 获取主机详情：

   示例

   ```
   [ceph: root@host01 /]# ceph orch host ls
   ```

3. 排空主机中的所有守护进程：

   语法

   ```
   ceph orch host drain HOSTNAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch host drain host02
   ```

   _no_schedule 标签自动应用到阻止部署的主机。

4. 检查移除 OSD 的状态：

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

   当 OSD 上没有剩余的放置组(PG)时，该 OSD 会停用并从存储集群中移除。

5. 检查所有守护进程是否已从存储集群中移除：

   语法

   ```
   ceph orch ps HOSTNAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch ps host02
   ```

6. 删除主机：

   语法

   ```
   ceph orch host rm HOSTNAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch host rm host02
   ```

其它资源

- 如需更多信息，请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 添加主机*的内容。
- 如需更多信息，请参阅 *Red Hat Ceph Storage Operations Guide* 中的*使用 Ceph Orchestrator 列出主机*的内容。

3.7. 使用 CEPH 编排器将主机置于维护模式

您可以使用 Ceph Orchestrator 将主机置于维护模式和停用状态。**ceph orch host maintenance enter** 命令停止 **systemd 目标**，这会导致主机上所有 Ceph 守护进程停止。类似地，**ceph orch host maintenance exit** 命令重新启动 **systemd 目标**，Ceph 守护进程会自行重启。

当主机被置于维护模式时，编配器采用以下工作流：

1. 运行 **orch host ok-to-stop** 命令确认删除主机不会影响数据可用性。
2. 如果主机有 Ceph OSD 守护进程，它会将 **noout** 应用到主机子树，以防止在计划的维护插槽期间触发数据迁移。
3. 停止 Ceph 目标，从而停止所有守护进程。
4. 禁用主机上的 **ceph 目标**，以防止重新引导来自动启动 Ceph 服务。

退出维护会反转上述序列。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 添加至集群的主机。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 您可以将主机置于维护模式，或者将其置于维护模式：
   - 将主机置于维护模式：

     语法

     ```
     ceph orch host maintenance enter HOST_NAME [--force]
     ```

     示例

     ```
     [ceph: root@host01 /]# ceph orch host maintenance enter host02 --force
     ```

--force 标志允许用户绕过警告，但不允许警报。

- 将主机从维护模式中放置：

  语法

  ```
  ceph orch host maintenance exit HOST_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch host maintenance exit host02
  ```

验证

- 列出主机：

  示例

  ```
  [ceph: root@host01 /]# ceph orch host ls
  ```

第 4 章 使用 CEPH ORCHESTRATOR 管理 MONITOR

作为存储管理员，您可以使用放置规格部署额外的 monitor，使用服务规格添加 monitor，将监控器添加到子网配置，并将监控器添加到特定的主机。除此之外，您还可以使用 Ceph Orchestrator 删除 monitor。

默认情况下，一般的 Red Hat Ceph Storage 集群在不同主机上部署有三个或五个监控守护进程。

如果集群中有五个或更多节点，红帽建议部署五个监控器。

注意

在使用 OSP director 部署 Ceph 时，红帽建议部署三个 monitor。

Ceph 会在集群增加时自动部署监控器守护进程，并在集群缩小时自动扩展后端监控守护进程。是否可以平稳地执行这个自动扩大和缩减取决于正确的子网配置。

如果您的 monitor 节点或整个集群都位于单个子网中，则 Cephadm 会在向集群添加新主机时自动添加最多五个 monitor 守护进程。Cephadm 在新主机上自动配置监控器守护进程。新主机与存储集群中引导的主机位于同一个子网中。

Cephadm 还可以部署和缩放 monitor，以响应存储集群大小的变化。

4.1. CEPH MONITOR

Ceph Monitor 是轻量型进程，维护存储集群映射的主副本。所有 Ceph 客户端都会联系 Ceph 监控器，并检索存储集群映射的当前副本，使客户端能够绑定到池并读写数据。

Ceph 监控程序使用 Paxos 协议的一种变体来就存储集群之间的映射和其他重要信息建立共识。由于 Paxos 的性质，Ceph 需要大多数 monitor 能够建立仲裁，从而建立共识。

重要

对于生产环境集群，需要在独立的主机上至少有三个监控器才能获得红帽的支持。

红帽建议部署奇数个监控器。奇数的 Ceph 监控器具有比偶数个监控器更高的故障恢复能力。例如，若要在双监视器部署上维护仲裁，Ceph 无法容忍任何故障；对于四个监视器，可以容忍一个失败，对于五个监视器，可以容忍两个失败。这就是建议为奇数的原因。总结一下，Ceph 需要大多数监控器正在运行，并能够相互通信，另外两个是三个，共三，共 4 个，以此类推。

对于多节点 Ceph 存储集群的初始部署，红帽需要至少三个监视器，当需要多于三个 monitor 的情况，每次需要增加 2 个。

由于 Ceph 监控是轻量级的，因此可以在与 OpenStack 节点相同的主机上运行。但是，红帽建议在独立主机上运行 monitor。

重要

红帽仅在容器化环境中支持并置 Ceph 服务。

从存储集群中移除 monitor 时，请考虑 Ceph Monitor 使用 Paxos 协议来建立关于主存储集群映射的共识。您必须有足够的数量的 Ceph 监控器来建立仲裁。

其它资源

其它资源

- 有关所有支持的 Ceph 配置，请参阅 Red Hat Ceph Storage *支持的配置*知识库文章。

4.2. 配置 MONITOR 选择策略

monitor 选择策略标识了网络分割并处理失败。您可以使用三种不同模式配置选择监控策略：

1. **classic** - 默认默认，它是最低等级的监控，根据两个站点之间的选举模块进行投票。
2. **disallow** - 此模式可让您将 monitor 标记为禁止，在这种情况下，他们会参与仲裁并服务客户端，但不能是选择的领导者。这样，您可以将 monitor 添加到禁止的领导列表中。如果 monitor 在 disallowed 列表中，它将始终被推迟到另一个 monitor。
3. **connectivity** - 这个模式主要用于解决网络差异。它会根据 ping 检查其对等点提供的 ping 评估连接分数，并选出最连接的、可靠监控成为领导机。这个模式旨在处理网络分割，如果您的集群在多个数据中心间扩展或存在影响，则可能会出现这种情况。这个模式包含连接分数评级，并以最佳分数选择监控器。如果需要将特定的 monitor 成为领导，请配置选择策略，使特定的 monitor 是排名中的第一个 monitor，其等级为 **0**。

红帽建议保持在 经典（**classic**） 模式，除非您需要其他模式的功能。

在构造集群前，将以下命令的 **election_strategy** 更改为 **classic**, **disallow**, 或 **connectivity**：

语法

```
ceph mon set election_strategy {classic|disallow|connectivity}
```

4.3. 使用命令行界面部署 CEPH 监控守护进程

Ceph 编排器默认部署一个监控器守护进程。您可以通过在命令行界面中使用 **放置** 规格来部署额外的监控守护进程。要部署不同数量的 monitor 守护进程，请指定不同的数字。如果您不指定应当部署管理器守护进程的主机，Ceph 编配器会随机选择主机，并将管理器守护进程部署到主机上。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 部署 Ceph 监控守护进程有四个不同的方法：

方法 1

- 使用放置规格在主机上部署监控器：

注意

红帽建议您使用 **--placement** 选项部署到特定主机上。

语法

```
ceph orch apply mon --placement="HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
```

示例

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01 host02 host03"
```

注意

务必将 bootstrap 节点包含为命令中的第一个节点。

重要

不要将监视器单独添加为 **ceph orch apply mon** supersedes，也不会将 monitor 添加到所有主机。例如，如果您运行以下命令，第一个命令在 **host01** 上创建 monitor。然后，第二个命令会取代 host1 上的监控器，并在 **host02** 上创建监控器。然后，第三个命令会取代 **host02** 上的监控器，并在 **host03** 上创建监控器。最后，只有第三个主机上有一个监控器。

```
# ceph orch apply mon host01
# ceph orch apply mon host02
# ceph orch apply mon host03
```

方法 2

- 使用放置规格，通过标签在特定主机上部署特定数量的监控器：

  a. 向主机添加标签：

  语法

  ```
  ceph orch host label add HOSTNAME_1 LABEL
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch host label add host01 mon
  ```

  b. 部署守护进程：

  语法

  ```
  ceph orch apply mon --placement="HOST_NAME_1:mon HOST_NAME_2:mon
  HOST_NAME_3:mon"
  ```

  示例

```
[ceph: root@host01 /]# ceph orch apply mon --placement="host01:mon host02:mon
host03:mon"
```

方法 3

- 使用放置规格在特定主机上部署特定数量的监控器：

  语法

  ```
  ceph orch apply mon --placement="NUMBER_OF_DAEMONS HOST_NAME_1
  HOST_NAME_2 HOST_NAME_3"
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply mon --placement="3 host01 host02 host03"
  ```

方法 4

- 在存储集群的主机上随机部署监控器守护进程：

  语法

  ```
  ceph orch apply mon NUMBER_OF_DAEMONS
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply mon 3
  ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

4.4. 使用服务规格部署 CEPH 监控守护进程

Ceph 编排器默认部署一个监控器守护进程。您可以使用服务规格（如 YAML 格式文件）部署额外的监控守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

流程

1. 创建 **mon.yaml** 文件：

   示例

   ```
   [root@host01 ~]# touch mon.yaml
   ```

2. 编辑 **mon.yaml** 文件，使其包含以下详情：

   语法

   ```
   service_type: mon
   placement:
     hosts:
       - HOST_NAME_1
       - HOST_NAME_2
   ```

   示例

   ```
   service_type: mon
   placement:
     hosts:
       - host01
       - host02
   ```

3. 将 YAML 文件挂载到容器中的一个目录下：

   示例

   ```
   [root@host01 ~]# cephadm shell --mount mon.yaml:/var/lib/ceph/mon/mon.yaml
   ```

4. 进入该目录：

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mon/
   ```

5. 部署监控器守护进程：

   语法

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   示例

```
[ceph: root@host01 mon]# ceph orch apply -i mon.yaml
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

4.5. 使用 CEPH 编排器在特定网络中部署监控器守护进程

Ceph 编排器默认部署一个监控器守护进程。您可以为每个 monitor 明确指定 IP 地址或 CIDR 网络，并控制放置每个 monitor 的位置。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 禁用自动监控器部署：

   示例

   ```
   [ceph: root@host01 /]# ceph orch apply mon --unmanaged
   ```

3. 在特定网络上的主机上部署监控器：

   语法

   ```
   ceph orch daemon add mon HOST_NAME_1:IP_OR_NETWORK
   ```

示例

```
[ceph: root@host01 /]# ceph orch daemon add mon host03:10.1.2.123
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

4.6. 使用 CEPH ORCHESTRATOR 删除 MONITOR 守护进程

要从主机中删除 monitor 守护进程，您只能在其他主机上重新部署 monitor 守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 至少一个在主机上部署的 monitor 守护进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 运行 **ceph orch apply** 命令来部署所需的监控器守护进程：

   语法

   ```
   ceph orch apply mon "NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_3"
   ```

   如果要从 **host02** 中删除 monitor 守护进程，您可以在其他主机上重新部署 monitor。

   示例

```
[ceph: root@host01 /]# ceph orch apply mon “2 host01 host03”
```

验证

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mon
  ```

其它资源

- 如需更多信息，请参阅 *Red Hat Ceph Storage Operations 指南中的使用命令行界面部分部署 Ceph 监控守护进程*。
- 如需更多信息，请参阅 *Red Hat Ceph Storage Operations 指南中的使用服务规格部署 Ceph 监控守护进程*。

4.7. 从不健康的存储集群中移除 CEPH MONITOR

您可以从不健康的存储集群中删除 **ceph-mon** 守护进程。不健康的存储集群是，其持续具有没有处于 **active + clean** 状态的放置组。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- Ceph 监控节点的根级别访问权限.
- 至少一个运行 Ceph Monitor 节点。

流程

1. 识别 Surviving 监控器并登录到主机：

   语法

   ```
   ssh root@MONITOR_ID
   ```

   示例

   ```
   [root@admin ~]# ssh root@host00
   ```

2. 登录到每个 Ceph Monitor 主机并停止所有 Ceph Monitor：

   语法

```
cephadm unit --name DAEMON_NAME.HOSTNAME stop
```

示例

```
[root@host00 ~]# cephadm unit --name mon.host00 stop
```

3. 设置适合扩展守护进程维护的环境，并以交互方式运行守护进程：

语法

```
cephadm shell --name DAEMON_NAME.HOSTNAME
```

示例

```
[root@host00 ~]# cephadm shell --name mon.host00
```

4. 提取 **monmap** 文件的副本：

语法

```
ceph-mon -i HOSTNAME --extract-monmap TEMP_PATH
```

示例

```
[ceph: root@host00 /]# ceph-mon -i host01 --extract-monmap /tmp/monmap

2022-01-05T11:13:24.440+0000 7f7603bd1700 -1 wrote monmap to /tmp/monmap
```

5. 删除非可见的 Ceph 监控器：

语法

```
monmaptool TEMPORARY_PATH --rm HOSTNAME
```

示例

```
[ceph: root@host00 /]# monmaptool /tmp/monmap --rm host01
```

6. 将 surviving monitor map 与已删除 monitor 注入 surviving Ceph Monitor:

语法

```
ceph-mon -i HOSTNAME --inject-monmap TEMP_PATH
```

示例

```
[ceph: root@host00 /]# ceph-mon -i host00 --inject-monmap /tmp/monmap
```

7. 仅启动 Surviving 监控器：

语法

```
cephadm unit --name DAEMON_NAME.HOSTNAME start
```

示例

```
[root@host00 ~]# cephadm unit --name mon.host00 start
```

8. 验证 monitor 形成仲裁：

示例

```
[ceph: root@host00 /]# ceph -s
```

9. 可选：在 **/var/lib/ceph/*CLUSTER_FSID*/mon.*HOSTNAME*** 目录中归档已删除的 Ceph Monitor 的数据目录。

第 5 章 使用 CEPH 编排器管理 MANAGERS

作为存储管理员，您可以使用 Ceph 编排器部署额外的管理器守护进程。在 bootstrap 过程中，Cephadm 会在 bootstrap 节点上自动安装管理器守护进程。

通常，您应该在运行 Ceph 监控守护进程的每个主机上设置 Ceph Manager，以实现相同的可用性级别。

默认情况下，**ceph-mgr** 实例首先由 Ceph 监控器激活，另一些都是备用管理器。不需要在 **ceph-mgr** 守护进程之间应该有一个仲裁。

如果活动守护进程无法向监控器发送 beacon 超过 **mon mgr beacon grace**，则它会被一个待机替换。

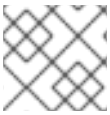
如果要预先进行故障转移，您可以使用 **ceph mgr fail MANAGER_NAME** 命令将 **ceph-mgr** 守护进程明确标记为失败。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。

5.1. 使用 CEPH ORCHESTRATOR 部署管理器守护进程

Ceph 编配器默认部署两个管理器守护进程。您可以通过在命令行界面中使用 **放置** 规格来部署额外的 manager 守护进程。要部署不同数量的管理器守护进程，请指定不同的数字。如果不指定应当部署 Manager 守护进程的主机，Ceph 编排器随机选择主机，并将 Manager 守护进程部署到其中。



注意

确保您的部署在每个部署中至少有三个 Ceph 管理器。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 您可以通过两种不同的方式部署管理器守护进程：

方法 1

- 使用特定主机组中的放置规格部署管理器守护进程：

注意

红帽建议您使用 **--placement** 选项部署到特定主机上。

语法

```
ceph orch apply mgr --placement=" HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
```

示例

```
[ceph: root@host01 /]# ceph orch apply mgr --placement="host01 host02 host03"
```

方法 2

- 在存储集群的主机上随机部署管理器守护进程：

  语法

  ```
  ceph orch apply mgr NUMBER_OF_DAEMONS
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply mgr 3
  ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mgr
  ```

5.2. 使用 CEPH ORCHESTRATOR 删除 MANAGER 守护进程

要从主机中删除管理器守护进程，您只需在其他主机上重新部署守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 主机上至少部署一个管理器守护进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 运行 **ceph orch apply** 命令以重新部署所需的管理器守护进程：

   语法

   ```
   ceph orch apply mgr "NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_3"
   ```

   如果要从 **host02** 中删除管理器守护进程，您可以在其他主机上重新部署 manager 守护进程。

   示例

   ```
   [ceph: root@host01 /]# ceph orch apply mgr "2 host01 host03"
   ```

验证

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mgr
  ```

其它资源

- 如需更多信息，请参阅使用 *Red Hat Ceph Storage 操作指南* 中的*使用 Ceph Orchestrator 部署管理器守护进程*。

5.3. 使用 CEPH MANAGER 模块

使用 **ceph mgr module ls** 命令查看可用的模块以及当前启用的模块。

使用 **ceph mgr module 启用或禁用模块启用 MODULE** 命令，或者 **ceph mgr module 会分别禁用 MODULE** 命令。

如果启用了模块，则活跃的 **ceph-mgr** 守护进程将加载并执行它。对于提供服务的模块（如 HTTP 服务器），则模块可能会在加载时发布其地址。要查看此类模块的地址，请运行 **ceph mgr services** 命令。

有些模块还可能实施特殊的待机模式，该模式在备用 **ceph-mgr** 守护进程和活动守护进程上运行。这可让服务提供的模块，在客户端尝试连接到待机时将其客户端重定向到活跃守护进程。

以下是启用 dashboard 模块的示例：

```
[ceph: root@host01 /]# ceph mgr module enable dashboard

[ceph: root@host01 /]# ceph mgr module ls
MODULE
balancer              on (always on)
crash                 on (always on)
devicehealth          on (always on)
orchestrator          on (always on)
pg_autoscaler         on (always on)
progress              on (always on)
rbd_support           on (always on)
status                on (always on)
telemetry             on (always on)
volumes               on (always on)
cephadm               on
dashboard             on
iostat                on
nfs                   on
prometheus            on
restful               on
alerts                -
diskprediction_local  -
influx                -
insights              -
k8sevents             -
localpool             -
mds_autoscaler        -
mirroring             -
osd_perf_query        -
osd_support           -
rgw                   -
rook                  -
selftest              -
snap_schedule         -
stats                 -
telegraf              -
test_orchestrator     -
zabbix                -

[ceph: root@host01 /]# ceph mgr services
{
        "dashboard": "http://myserver.com:7789/",
        "restful": "https://myserver.com:8789/"
}
```

集群首次启动时，它使用 **mgr_initial_modules** 设置覆盖要启用哪些模块。但是，通过集群的其余部分忽略此设置：只将其用于 bootstrap。例如，在第一次启动 monitor 守护进程前，您可以在 **ceph.conf** 文件中添加类似如下的部分：

```
[mon]
    mgr initial modules = dashboard balancer
```

如果模块实施注释行 hook，命令可作为普通 Ceph 命令访问，Ceph 会自动将模块命令合并到标准 CLI 界面中，并将其正确路由到模块：

```
[ceph: root@host01 /]# ceph <command | help>
```

您可以在上述命令中使用以下配置参数：

表 5.1. 配置参数

配置	描述	类型	default
mgr 模块路径	从中加载模块的路径。	字符串	**"<library dir>/mgr"**
mgr 数据	加载守护进程数据的路径（如密钥环）	字符串	**"/var/lib/ceph/mgr/$cluster-$id"**
mgr tick period	Manager beacons to monitor 和其它定期检查之间的秒数。	整数	**5**
mon mgr beacon grace	最后一个 beacon 后的时长应被视为管理器失败。	整数	**30**

5.4. 使用 CEPH MANAGER 负载均衡器模块

balancer 是 Ceph Manager(**ceph-mgr**)的一个模块，用于优化 OSD 之间放置组(PG)放置，从而实现平衡的分发（可自动或监管方式）。

目前无法禁用 balancer 模块。它只能关闭自定义配置。

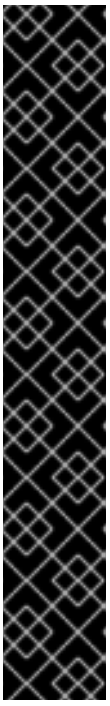
模式

目前支持的负载均衡器模式有两种：

- CRUSH **-compat** ：CRUSH compat 模式使用 Ceph Luminous 中引入的兼容 **weight-set** 功能来管理 CRUSH 层次结构中设备的备用权重集合。普通权重应保持设置为设备的大小，以反映您要存储在设备上的数据数量。然后，负载均衡器会优化 **weight-set** 值，以较小的增量调整它们，以实现与目标分布匹配的发行版。由于 PG 放置是一种伪随机进程，因此放置有自然变化；通过优化权重，平衡平衡器的作用是自然变化。
  这个模式与旧的客户端完全向后兼容。当 OSDMap 和 CRUSH map 与旧客户端共享时，平衡器会将优化的 weightsff 显示为实际权重。

  此模式的主要限制是，如果层次结构的子树共享任何 OSD，则均衡器无法处理具有不同放置规则的多个 CRUSH 层次结构。由于此配置使得在共享 OSD 上管理空间利用率比较困难，因此通常不建议这样做。因此，这个限制通常不是问题。

- **upmap**: 从 Luminous 开始，OSDMap 可以存储各个 OSD 的显式映射，如普通的 CRUSH 放置计算例外。这些 **upmap** 条目提供对 PG 映射的精细控制。此 CRUSH 模式将优化各个 PG 的放置，以实现均衡的分发。在大多数情况下，此分布为"完美"，每个 OSD +/-1 PG 上相等的 PG 数量，因为它们可能无法均匀划分。



重要

要允许使用这个功能，您必须使用以下命令告知集群只需要支持 luminous 或更新的客户端：

```
[ceph: root@host01 /]# ceph osd set-require-min-compat-client luminous
```

如果任何 pre-luminous 客户端或守护进程连接到 monitor，则此命令会失败。

由于一个已知问题，内核 CephFS 客户端会将自身报告为 jewel 客户端。要临时解决这个问题，请使用 **--yes-i-really-mean-it** 标志：

```
[ceph: root@host01 /]# ceph osd set-require-min-compat-client luminous --
yes-i-really-mean-it
```

您可以检查哪些客户端版本被用于：

```
[ceph: root@host01 /]# ceph features
```

5.4.1. 使用容量平衡 Red Hat Ceph 集群

使用容量平衡 Red Hat Ceph 存储集群。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

流程

1. 检查是否启用了 balancer 模块：

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module enable balancer
   ```

2. 打开 balancer 模块：

   示例

   ```
   [ceph: root@host01 /]# ceph balancer on
   ```

3. 要更改模式，请使用以下命令：默认模式是 **upmap** ：

   示例

   ```
   [ceph: root@host01 /]# ceph balancer mode crush-compat
   ```

   或

   示例

   ```
   [ceph: root@host01 /]# ceph balancer mode upmap
   ```

4. 检查负载均衡器的当前状态。

示例

```
[ceph: root@host01 /]# ceph balancer status
```

自动平衡

默认情况下，在打开 balancer 模块时会使用自动平衡：

示例

```
[ceph: root@host01 /]# ceph balancer on
```

您可以使用以下方法再次关闭负载均衡器：

示例

```
[ceph: root@host01 /]# ceph balancer off
```

这使用 **crush-compat** 模式，与旧的客户端向后兼容，并随着时间的推移对数据分布进行小更改，以确保 OSD 平等地使用。

节流

如果集群已降级，则没有对 PG 分发的调整，例如，如果 OSD 失败，系统尚未修复自身。

当集群处于健康状态时，负载均衡器会节流到其更改，使得 PG 百分比被错误或需要移动，默认低于 5%。可以使用 **target_max_misplaced_ratio** 设置调整这个百分比。例如，将阈值增加到 7%：

示例

```
[ceph: root@host01 /]# ceph config-key set mgr target_max_misplaced_ratio .07
```

对于自动平衡：

- 在自动负载均衡器运行之间将休眠的秒数：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/sleep_interval 60
```

- 将一天的时间设置为以 HHMM 格式开始自动平衡：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/begin_time 0000
```

- 将当天的时间设置为以 HHMM 格式完成自动平衡：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/end_time 2359
```

- 限制本周或更高版本的自动平衡。使用与 crontab 相同的约定，**0** 为 Sunday，**1** 为 Monday，以此类推：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/begin_weekday 0
```

- 限制本周或更早版本自动平衡。这使用与 crontab 相同的约定，**0** 为 Sunday，**1** 为 Monday，以此类推：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/end_weekday 6
```

- 定义自动平衡仅限于的池 ID。此默认值是一个空字符串，表示所有池都是 balanced。可以使用 **ceph osd pool ls detail** 命令获取数字池 ID：

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/balancer/pool_ids 1,2,3
```

监控的优化

balancer 操作分为几个不同的阶段：

1. 构建 **计划**。
2. 评估数据分发的质量，针对当前的 PG 分发，或在执行一个**计划（plan）**后生成的 PG 分发。
3. 执行**计划**。
   - 评估和评分当前发行版：

     示例

     ```
     [ceph: root@host01 /]# ceph balancer eval
     ```

   - 评估单个池的发布：

     语法

     ```
     ceph balancer eval POOL_NAME
     ```

     示例

     ```
     [ceph: root@host01 /]# ceph balancer eval rbd
     ```

   - 查看更多评估详情：

     示例

     ```
     [ceph: root@host01 /]# ceph balancer eval-verbose ...
     ```

- 使用当前配置模式生成计划：

  语法

  ```
  ceph balancer optimize PLAN_NAME
  ```

  使用自定义计划名称替换 *PLAN_NAME*。

  示例

  ```
  [ceph: root@host01 /]# ceph balancer optimize rbd_123
  ```

- 查看计划的内容：

  语法

  ```
  ceph balancer show PLAN_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph balancer show rbd_123
  ```

- 要丢弃旧计划：

  语法

  ```
  ceph balancer rm PLAN_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph balancer rm rbd_123
  ```

- 要查看当前记录的计划，请使用 status 命令：

  ```
  [ceph: root@host01 /]# ceph balancer status
  ```

- 要计算执行计划后结果的分发质量：

  语法

  ```
  ceph balancer eval PLAN_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph balancer eval rbd_123
  ```

- 执行计划：

  语法

  ```
  ceph balancer execute PLAN_NAME
  ```

示例

```
[ceph: root@host01 /]# ceph balancer execute rbd_123
```

注意

只有预期会改进发布时才执行计划。执行后，计划将被丢弃。

5.4.2. 使用读取负载均衡器平衡 Red Hat Ceph 集群 [技术预览]

重要

read Balancer 只是一个技术预览功能，仅适用于 Red Hat Ceph Storage 7.0。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。如需了解更多详细信息，请参阅红帽技术预览功能的支持范围。

如果您有未平衡的主 OSD，您可以使用 **osdmaptool** 中内置的离线优化器进行更新。

红帽建议您在运行负载均衡器前运行容量负载均衡器以确保最佳结果。

按照流程中的步骤，使用读取负载均衡器平衡集群：

先决条件

- 一个正在运行的和容量平衡的 Red Hat Ceph Storage 集群。
- 红帽建议您运行容量负载均衡器，以便在运行读取负载均衡器前平衡每个 OSD 上的容量，以确保最佳结果。运行以下命令来平衡容量：

  1. 获取 osdmap 的最新副本。

     ```
     [ceph: root@host01 /]# ceph osd  getmap -o map
     ```

  2. 运行 upmap balancer。

     ```
     [ceph: root@host01 /]# ospmaptool map –upmap out.txt
     ```

  3. 文件 out.txt 包含所提议的解决方案。
     此流程中的命令是运行的常规 Ceph CLI 命令，以将更改应用到集群。

     如果 out.txt 文件中有任何建议，请运行以下命令。

     ```
     [ceph: root@host01 /]# source out.txt
     ```

     如需更多信息，请参阅使用容量平衡 IBM Ceph 集群

流程

1. 检查每个池的 **read_balance_score**:

```
[ceph: root@host01 /]# ceph osd pool ls detail
```

如果 **read_balance_score** 高于 1，则您的池具有未平衡的 Primary OSD。

对于同构集群，最佳分数为 [Ceil{(PG/Number of OSDs)}/(PG/Number of OSDs)]/[(PG/Number of OSDs 的数量)/(PG/Number of OSDs)]。例如，如果您有一个具有 32 个 PG 和 10 个 OSD 的池，则 (PG/Number of OSDs 的数量) = 32/10 = 3.2。因此，如果所有设备都相同，则最佳分数是 3.2 的静默值(PG/Number of OSDs 的数量)是 4/3.2 = 1.25。如果您在具有 64 个 PG 的同一系统中有另一个池，则最佳分数为 7/6.4 =1.09375

输出示例：

```
$ ceph osd pool ls detail
pool 1 '.mgr' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 1
pgp_num 1 autoscale_mode on last_change 17 flags hashpspool stripe_width 0
pg_num_max 32 pg_num_min 1 application mgr read_balance_score 3.00
pool 2 'cephfs.a.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num
16 pgp_num 16 autoscale_mode on last_change 55 lfor 0/0/25 flags hashpspool stripe_width
0 pg_autoscale_bias 4 pg_num_min 16 recovery_priority 5 application cephfs
read_balance_score 1.50
pool 3 'cephfs.a.data' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num
128 pgp_num 128 autoscale_mode on last_change 27 lfor 0/0/25 flags hashpspool,bulk
stripe_width 0 application cephfs read_balance_score 1.31
```

2. 获取 **osdmap** 的最新副本：

```
[ceph: root@host01 /]# ceph osd getmap -o om
```

输出示例：

```
got osdmap epoch 56
```

3. 运行 optimizer：
文件 **out.txt** 包含所提议的解决方案。

```
[ceph: root@host01 /]# osdmaptool om --read out.txt --read-pool _POOL_NAME_ [--vstart]
```

输出示例：

```
$ osdmaptool om --read out.txt --read-pool cephfs.a.meta
./bin/osdmaptool: osdmap file 'om'
writing upmap command output to: out.txt
---------- BEFORE ------------
 osd.0 | primary affinity: 1 | number of prims: 4
 osd.1 | primary affinity: 1 | number of prims: 8
 osd.2 | primary affinity: 1 | number of prims: 4

read_balance_score of 'cephfs.a.meta': 1.5

---------- AFTER ------------
 osd.0 | primary affinity: 1 | number of prims: 5
 osd.1 | primary affinity: 1 | number of prims: 6
 osd.2 | primary affinity: 1 | number of prims: 5
```

```
read_balance_score of 'cephfs.a.meta': 1.13


num changes: 2
```

4. 文件 **out.txt** 包含所提议的解决方案。
此流程中的命令是运行正常的 Ceph CLI 命令，以便对集群应用更改。如果您在 vstart 集群中工作，您可以传递 **--vstart** 参数，以便 CLI 命令使用 **./bin/ 前缀进行格式化。**

```
[ceph: root@host01 /]# source out.txt
```

输出示例：

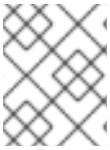
```
$ cat out.txt
ceph osd pg-upmap-primary 2.3 0
ceph osd pg-upmap-primary 2.4 2

$ source out.txt
change primary for pg 2.3 to osd.0
change primary for pg 2.4 to osd.2
```

注意

如果您第一次运行 **ceph osd pg-upmap-primary** 命令，您可能会收到如下警告：

```
Error EPERM: min_compat_client luminous < reef, which is required for pg-upmap-primary. Try 'ceph osd set-require-min-compat-client reef' before using the new interface
```

在本例中，运行推荐的命令 **ceph osd set-require-min-compat-client reef**，并调整集群的 min-compact-client。

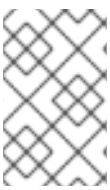


注意

如果放置组(PG)数量添加到集群中或从集群中删除任何 OSD，则请考虑重新检查分数并重新运行负载均衡器，因为这些操作可能会显著影响对池的读取负载均衡器的影响。

5.5. 使用 CEPH MANAGER 警报模块

您可以使用 Ceph 管理器警报模块通过电子邮件发送关于 Red Hat Ceph Storage 集群健康状况的简单警报消息。



注意

这个模块并不是一个可靠的监控解决方案。作为 Ceph 集群本身一部分运行的事实是，在 **ceph-mgr** 守护进程出现故障时，它完全限制会防止警报被发送。但是，对于没有监控架构的环境中存在的一个独立的集群非常有用。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- Ceph 监控节点的根级别访问权限.

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 启用警报模块：

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module enable alerts
   ```

3. 确保启用了 alert 模块：

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module ls | more
   {
       "always_on_modules": [
           "balancer",
           "crash",
           "devicehealth",
           "orchestrator",
           "pg_autoscaler",
           "progress",
           "rbd_support",
           "status",
           "telemetry",
           "volumes"
       ],
       "enabled_modules": [
           "alerts",
           "cephadm",
           "dashboard",
           "iostat",
           "nfs",
           "prometheus",
           "restful"
       ]
   ```

4. 配置简单邮件传输协议(SMTP)：

   语法

   ```
   ceph config set mgr mgr/alerts/smtp_host SMTP_SERVER
   ceph config set mgr mgr/alerts/smtp_destination RECEIVER_EMAIL_ADDRESS
   ceph config set mgr mgr/alerts/smtp_sender SENDER_EMAIL_ADDRESS
   ```

   示例

   -

```
[ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_host smtp.example.com
[ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_destination
example@example.com
[ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_sender
example2@example.com
```

5. 可选： 默认情况下，警报模块使用 SSL 和端口 465。

   语法

   ```
   ceph config set mgr mgr/alerts/smtp_port PORT_NUMBER
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_port 587
   ```

   在配置警报时不要设置 **smtp_ssl** 参数。

6. 向 SMTP 服务器进行身份验证：

   语法

   ```
   ceph config set mgr mgr/alerts/smtp_user USERNAME
   ceph config set mgr mgr/alerts/smtp_password PASSWORD
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_user admin1234
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_password admin1234
   ```

7. 可选：默认情况下，SMTP **From** 名称是 **Ceph**。要更改它，请设置 **smtp_from_name** 参数：

   语法

   ```
   ceph config set mgr mgr/alerts/smtp_from_name CLUSTER_NAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/smtp_from_name 'Ceph Cluster Test'
   ```

8. 可选：默认情况下，警报模块会每分钟检查存储集群的健康状况，并在集群健康状况有变化时发送消息。要更改频率，请设置 **interval** 参数：

   语法

   ```
   ceph config set mgr mgr/alerts/interval INTERVAL
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph config set mgr mgr/alerts/interval "5m"
   ```

在本例中，间隔设置为 5 分钟。

9. 可选：立即发送警报：

示例

```
[ceph: root@host01 /]# ceph alerts send
```

其它资源

- 如需有关 *Ceph 健康消息* 的更多信息，请参阅 *Red Hat Ceph Storage 故障排除指南中的* Ceph 集群的健康消息部分。

5.6. 使用 CEPH 管理器 CRASH 模块

通过使用 Ceph 管理器 crash 模块，您可以收集有关守护进程 crashdumps 的信息，并将其存储在 Red Hat Ceph Storage 集群中，以便进一步分析。

默认情况下，守护进程崩溃转储在 **/var/lib/ceph/crash** 中转储。您可以使用选项 **crash dir** 进行配置。崩溃目录按时间、日期和随机生成的 UUID 命名，包含元数据文件 **meta** 和最新的日志文件，其 **crash_id** 相同。

您可以使用 **ceph-crash.service** 自动提交这些崩溃，并在 Ceph 监控器中保留。 **ceph-crash.service** 监视 **crashdump** 目录，并使用 **ceph crash post** 上传它们。

RECENT_CRASH heath 消息是 Ceph 集群中最常见的运行状况消息之一。此健康消息表示，一个或多个 Ceph 守护进程最近崩溃，且崩溃尚未存档或被管理员确认。这可能表示软件错误、硬件问题（如磁盘失败）或其它问题。选项 **mgr/crash/warn_recent_interval** 控制最近一次表示的时间周期，默认为两周。您可以运行以下命令来禁用警告：

示例

```
[ceph: root@host01 /]# ceph config set mgr/crash/warn_recent_interval 0
```

选项 **mgr/crash/retain_interval** 控制您要保留崩溃报告的周期，然后再自动清除崩溃报告。这个选项的默认值是一年。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

流程

1. 确定启用了 crash 模块：

示例

```
[ceph: root@host01 /]# ceph mgr module ls | more
{
    "always_on_modules": [
        "balancer",
        "crash",
        "devicehealth",
        "orchestrator_cli",
```

```
        "progress",
        "rbd_support",
        "status",
        "volumes"
    ],
    "enabled_modules": [
        "dashboard",
        "pg_autoscaler",
        "prometheus"
    ]
```

2. 保存崩溃转储：元数据文件是存储在 crash dir 中作为 **meta** 的 JSON blob。您可以调用 ceph 命令 **-i -** 选项，该选项会从 stdin 读取。

   示例

   ```
   [ceph: root@host01 /]# ceph crash post -i meta
   ```

3. 列出所有新的以及归档的崩溃信息的时间戳或 UUID 崩溃 ID：

   示例

   ```
   [ceph: root@host01 /]# ceph crash ls
   ```

4. 列出所有新崩溃信息的时间戳或 UUID 崩溃 ID：

   示例

   ```
   [ceph: root@host01 /]# ceph crash ls-new
   ```

5. 列出所有新崩溃信息的时间戳或 UUID 崩溃 ID：

   示例

   ```
   [ceph: root@host01 /]# ceph crash ls-new
   ```

6. 列出按年龄分组的保存崩溃信息的摘要：

   示例

   ```
   [ceph: root@host01 /]# ceph crash stat
   8 crashes recorded
   8 older than 1 days old:
   2022-05-20T08:30:14.533316Z_4ea88673-8db6-4959-a8c6-0eea22d305c2
   2022-05-20T08:30:14.590789Z_30a8bb92-2147-4e0f-a58b-a12c2c73d4f5
   2022-05-20T08:34:42.278648Z_6a91a778-bce6-4ef3-a3fb-84c4276c8297
   2022-05-20T08:34:42.801268Z_e5f25c74-c381-46b1-bee3-63d891f9fc2d
   2022-05-20T08:34:42.803141Z_96adfc59-be3a-4a38-9981-e71ad3d55e47
   2022-05-20T08:34:42.830416Z_e45ed474-550c-44b3-b9bb-283e3f4cc1fe
   2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d
   2022-05-24T19:58:44.315282Z_1847afbc-f8a9-45da-94e8-5aef0738954e
   ```

7. 查看保存崩溃的详情：

语法

```
ceph crash info CRASH_ID
```

示例

```
[ceph: root@host01 /]# ceph crash info 2022-05-24T19:58:42.549073Z_b2382865-ea89-
4be2-b46f-9a59af7b7a2d
{
    "assert_condition": "session_map.sessions.empty()",
    "assert_file": "/builddir/build/BUILD/ceph-16.1.0-486-g324d7073/src/mon/Monitor.cc",
    "assert_func": "virtual Monitor::~Monitor()",
    "assert_line": 287,
    "assert_msg": "/builddir/build/BUILD/ceph-16.1.0-486-g324d7073/src/mon/Monitor.cc: In
function 'virtual Monitor::~Monitor()' thread 7f67a1aeb700 time 2022-05-
24T19:58:42.545485+0000\n/builddir/build/BUILD/ceph-16.1.0-486-
g324d7073/src/mon/Monitor.cc: 287: FAILED
ceph_assert(session_map.sessions.empty())\n",
    "assert_thread_name": "ceph-mon",
    "backtrace": [
        "/lib64/libpthread.so.0(+0x12b30) [0x7f679678bb30]",
        "gsignal()",
        "abort()",
        "(ceph::__ceph_assert_fail(char const*, char const*, int, char const*)+0x1a9)
[0x7f6798c8d37b]",
        "/usr/lib64/ceph/libceph-common.so.2(+0x276544) [0x7f6798c8d544]",
        "(Monitor::~Monitor()+0xe30) [0x561152ed3c80]",
        "(Monitor::~Monitor()+0xd) [0x561152ed3cdd]",
        "main()",
        "__libc_start_main()",
        "_start()"
    ],
    "ceph_version": "16.2.8-65.el8cp",
    "crash_id": "2022-07-06T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d",
    "entity_name": "mon.ceph-adm4",
    "os_id": "rhel",
    "os_name": "Red Hat Enterprise Linux",
    "os_version": "8.5 (Ootpa)",
    "os_version_id": "8.5",
    "process_name": "ceph-mon",
    "stack_sig":
"957c21d558d0cba4cee9e8aaf9227b3b1b09738b8a4d2c9f4dc26d9233b0d511",
    "timestamp": "2022-07-06T19:58:42.549073Z",
    "utsname_hostname": "host02",
    "utsname_machine": "x86_64",
    "utsname_release": "4.18.0-240.15.1.el8_3.x86_64",
    "utsname_sysname": "Linux",
    "utsname_version": "#1 SMP Wed Jul 06 03:12:15 EDT 2022"
}
```

8. 删除比 *KEEP* days 旧的已保存的崩溃：其中 *KEEP* 必须是一个整数。

语法

```
ceph crash prune KEEP
```

-

示例

```
[ceph: root@host01 /]# ceph crash prune 60
```

9. 对崩溃报告进行归档，使其不再被视为 **RECENT_CRASH** 健康检查，且不会出现在 **crash ls-new** 输出中。它会出现在 **crash ls** 中。

语法

```
ceph crash archive CRASH_ID
```

示例

```
[ceph: root@host01 /]# ceph crash archive 2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d
```

10. 记录所有崩溃报告：

示例

```
[ceph: root@host01 /]# ceph crash archive-all
```

11. 删除崩溃转储：

语法

```
ceph crash rm CRASH_ID
```

示例

```
[ceph: root@host01 /]# ceph crash rm 2022-05-24T19:58:42.549073Z_b2382865-ea89-4be2-b46f-9a59af7b7a2d
```

其它资源

- 如需有关 *Ceph 健康消息* 的更多信息，请参阅 *Red Hat Ceph Storage 故障排除指南中的* Ceph 集群的健康消息部分。

5.7. TELEMETRY 模块

遥测模块发送有关存储集群的数据，以帮助了解 Ceph 的使用方式以及操作过程中遇到的问题。在公共仪表板上视觉化数据，以查看报告集群数量、其总容量和 OSD 数量以及版本分布趋势的摘要统计。

Channels

遥测报告分为不同的频道，每种频道都有不同类型的信息。启用遥测后，您可以打开或关闭单个频道。

以下是四个不同的频道：

- **Basic** - 默认为 **on**。此频道提供有关集群的基本信息，其中包括以下信息：

- 集群的容量。
- 监视器、管理器、OSD、MDS、对象网关或其他守护进程的数量。
- 当前正在使用的软件版本。
- RADOS 池和 Ceph 文件系统的数量和类型。
- 从默认值（而不是其值）更改的配置选项的名称。

- **crash** - 默认为 **on**。这个频道提供有关守护进程崩溃的信息，其中包括以下信息：
  - 守护进程的类型。
  - 守护进程的版本。
  - 操作系统、操作系统分发和内核版本。
  - 标识崩溃的 Ceph 代码中的位置的堆栈追踪。
- **设备** - 默认 **位于**。此频道提供有关设备指标的信息，其中包括匿名 SMART 指标。
- **Ident** - 默认为 **off**。此频道为用户提供了集群相关的标识信息，如集群描述和联系电子邮件地址。
- **perf** - 默认为 **off**。此频道提供集群的各种性能指标，可用于以下内容：
  - 显示集群整体健康状况。
  - 识别工作负载模式。
  - 对延迟、节流、内存管理和其他类似问题进行故障排除。
  - 通过守护进程监控集群性能。

报告的数据不包含任何敏感数据，如池名称、对象名称、对象内容、主机名或设备序列号。

它包含集群如何部署、Ceph 版本、主机分发和其他参数的计数器和统计信息，可帮助项目更好地了解 Ceph 的使用方式。

数据安全，并发送到 https://telemetry.ceph.com。

启用遥测

在启用频道前，请确保遥测在 **上**。

- 启用遥测：

```
ceph telemetry on
```

启用和禁用频道

- 启用或禁用单个频道：

```
ceph telemetry enable channel basic
ceph telemetry enable channel crash
ceph telemetry enable channel device
```

```
ceph telemetry enable channel ident
ceph telemetry enable channel perf

ceph telemetry disable channel basic
ceph telemetry disable channel crash
ceph telemetry disable channel device
ceph telemetry disable channel ident
ceph telemetry disable channel perf
```

- 启用或禁用多个频道：

```
ceph telemetry enable channel basic crash device ident perf
ceph telemetry disable channel basic crash device ident perf
```

- 启用或禁用所有频道：

```
ceph telemetry enable channel all
ceph telemetry disable channel all
```

报告示例

- 要随时查看报告的数据，请生成示例报告：

```
ceph telemetry show
```

- 如果 Telemetry **已关闭**，请预览示例报告：

```
ceph telemetry preview
```

为具有数百个 OSD 或更多 OSD 的存储集群生成示例报告需要更长的时间。

- 为了保护您的隐私，设备报告是单独生成的，主机名和设备序列号等数据被匿名化。设备遥测发送到不同的端点，且不会将设备数据与特定集群相关联。要查看设备报告，请运行以下命令：

```
ceph telemetry show-device
```

- 如果 Telemetry **已关闭**，请预览示例设备报告：

```
ceph telemetry preview-device
```

- 获取 **上带有** telemetry 的两个报告的单一输出：

```
ceph telemetry show-all
```

- 通过遥测(**off**)获取两个报告的一个输出：

```
ceph telemetry preview-all
```

- 根据频道生成示例报告：

语法

```
ceph telemetry show CHANNEL_NAME
```

- 根据频道生成示例报告的预览：

  语法

  ```
  ceph telemetry preview CHANNEL_NAME
  ```

集合

集合是频道中收集的数据的不同方面。

- 列出集合：

  ```
  ceph telemetry collection ls
  ```

- 查看您注册的集合与新的可用集合之间的区别：

  ```
  ceph telemetry diff
  ```

- 注册最新的集合：

  语法

  ```
  ceph telemetry on
  ceph telemetry enable channel CHANNEL_NAME
  ```

Interval（间隔）

默认情况下，模块编译并每 24 小时发送一次新的报告。

- 调整间隔：

  语法

  ```
  ceph config set mgr mgr/telemetry/interval INTERVAL
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/interval 72
  ```

  在示例中，报告每三天生成一次(72 小时)。

Status

- 查看当前配置：

  ```
  ceph telemetry status
  ```

手动发送遥测

- 根据临时发送遥测数据：

```
ceph telemetry send
```

如果遥测被禁用，请将 **--license shared-1-0** 添加到 **ceph telemetry send** 命令。

通过代理发送遥测

- 如果集群无法直接连接到配置的遥测端点，您可以配置 HTTP/HTTPs 代理服务器：

  语法

  ```
  ceph config set mgr mgr/telemetry/proxy PROXY_URL
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/proxy https://10.0.0.1:8080
  ```

  您可以在命令中包含用户传递：

  示例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/proxy https://10.0.0.1:8080
  ```

联系和描述

- 可选：在报告中添加联系人和描述：

  语法

  ```
  ceph config set mgr mgr/telemetry/contact '_CONTACT_NAME_'
  ceph config set mgr mgr/telemetry/description '_DESCRIPTION_'
  ceph config set mgr mgr/telemetry/channel_ident true
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/contact 'John Doe
  <john.doe@example.com>'
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/description 'My first Ceph cluster'
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/channel_ident true
  ```

  如果启用了 **ident** 标志，其详情不会在领导板中显示。

Leaderboard

- 在公共仪表板上参与领导板：

  示例

  ```
  [ceph: root@host01 /]# ceph config set mgr mgr/telemetry/leaderboard true
  ```

  领导板显示有关存储集群的基本信息。此板包括存储容量和 OSD 数量。

禁用遥测

- 随时禁用遥测：

  示例

  ```
  ceph telemetry off
  ```

第 6 章 使用 CEPH ORCHESTRATOR 管理 OSD

作为存储管理员，您可以使用 Ceph 编排器管理 Red Hat Ceph Storage 集群的 OSD。

6.1. CEPH OSD

当 Red Hat Ceph Storage 集群启动并运行时，您可以在运行时将 OSD 添加到存储集群中。

Ceph OSD 通常由一个存储驱动器和一个 **ceph-osd** 守护进程和一个节点中的相关日志组成。如果节点有多个存储驱动器，则为每个驱动器映射一个 **ceph-osd** 守护进程。

红帽建议定期检查集群的容量，以查看它是否达到其存储容量的上限。当存储集群达到其 **近满（near full）**比率时，添加一个或多个 OSD 来扩展存储集群的容量。

当您要缩小 Red Hat Ceph Storage 集群大小或替换硬件时，您还可以在运行时移除 OSD。如果节点有多个存储驱动器，您可能还需要为该驱动器删除其中一个 **ceph-osd** 守护进程。通常，最好检查存储集群的容量，以查看您是否达到其容量的上限。在删除 OSD 后，确保存储集群没有达到**接近全满**比率。

重要

在添加 OSD 前，不要让存储集群达到**全满**比率。在存储集群达到**接近满**比率后发生 OSD 故障可能会导致存储集群超过**全满**比率。Ceph 会阻止写入访问来保护数据，直到您解决存储容量问题。在删除 OSD 前，需要仔细考虑它对 **full** 比率的影响。

6.2. CEPH OSD 节点配置

配置 Ceph OSD 及其支持硬件，类似于使用 OSD 的池的存储策略。Ceph 优先选择池中的统一硬件，以实现一致的性能配置集。为了获得最佳性能，请考虑使用相同类型或大小的驱动器的 CRUSH 层次结构。

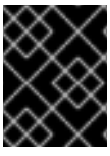
如果您添加了 dissimilar 大小的驱动器，请相应地调整它们的权重。将 OSD 添加到 CRUSH map 时，请考虑新 OSD 的权重。硬盘驱动器容量增长约 40%，因此较新的 OSD 节点可能会比存储集群中的旧节点更长的硬盘驱动器，即它们可能具有更大的权重。

在进行新安装前，请参阅 *安装指南中的安装 Red Hat Ceph Storage 的要求*。
https://access.redhat.com/documentation/zh-cn/red_hat_ceph_storage/7/html-single/installation_guide/#requirements-for-installing-red-hat-ceph-storage

6.3. 自动调优 OSD 内存

*OSD 守护进程根据 **osd_memory_target** 配置选项调整内存消耗。选项 **osd_memory_target** 根据系统中可用的 RAM 来设置 OSD 内存。*

*如果 Red Hat Ceph Storage 部署在不与其他服务共享内存的专用节点上，**cephadm** 会自动根据 RAM 总量和部署的 OSD 数量自动调整每个 OSD 消耗。*



重要

*默认情况下，Red Hat Ceph Storage 集群中的 **osd_memory_target_autotune** 参数设置为 **true**。*

语法

```
ceph config set osd osd_memory_target_autotune true
```

Cephadm 以一个 **mgr/cephadm/autotune_memory_target_ratio** 分数开头，默认为系统总 RAM 的 **0.7**，这会减小非自动tuned守护进程（如 non-OSDS）以及 **osd_memory_target_autotune** 为 false 的 OSD，然后划分剩余的 OSD。

osd_memory_target 参数计算如下：

语法

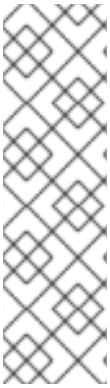
```
osd_memory_target = TOTAL_RAM_OF_THE_OSD * (1048576) * (autotune_memory_target_ratio) / NUMBER_OF_OSDS_IN_THE_OSD_NODE - (SPACE_ALLOCATED_FOR_OTHER_DAEMONS)
```

SPACE_ALLOCATED_FOR_OTHER_DAEMONS 可能包括以下守护进程空间分配：

- Alertmanager: 1 GB
- Grafana: 1 GB
- Ceph Manager：4 GB
- Ceph Monitor: 2 GB
- Node-exporter: 1 GB
- Prometheus: 1 GB

例如，如果节点有 24 个 OSD 且具有 251 GB RAM 空间，则 **osd_memory_target** 为 **7860684936**。

最终目标反映在带有选项的配置数据库中。您可以从 **ceph orch ps** 输出的 **MEM LIMIT** 列下查看各个守护进程使用的限值和当前内存。



注意

osd_memory_target_autotune true 的默认设置不适用于计算和 Ceph 存储服务在一起的超融合基础架构。在超融合基础架构中，**autotune_memory_target_ratio** 可以设置为 **0.2**，以减少 Ceph 的内存消耗。

示例

```
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/autotune_memory_target_ratio 0.2
```

您可以为存储集群中的 OSD 手动设置特定内存目标。

示例

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 7860684936
```

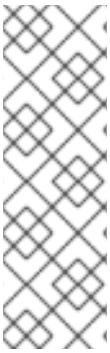
您可以为存储集群中的 OSD 主机手动设置特定内存目标。

语法

```
ceph config set osd/host:HOSTNAME osd_memory_target TARGET_BYTES
```

示例

```
[ceph: root@host01 /]# ceph config set osd/host:host01 osd_memory_target 1000000000
```



注意

启用 **osd_memory_target_autotune** 覆盖现有的手动 OSD 内存目标设置。要防止守护进程内存被调整（即使启用了 **osd_memory_target_autotune** 选项或启用了其他类似的选项），在主机上设置 **_no_autotune_memory** 标签。

语法

```
ceph orch host label add HOSTNAME _no_autotune_memory
```

您可以通过禁用 autotune 选项并设置特定内存目标，从内存自动调整 OSD 中排除。

示例

```
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target_autotune false
[ceph: root@host01 /]# ceph config set osd.123 osd_memory_target 16G
```

6.4. 列出 CEPH OSD 部署的设备

在使用 Ceph 编排器部署 OSD 之前，您可以检查可用设备列表。命令用于显示可由 Cephadm 发现的设备列表。如果满足以下条件，则存储设备被视为可用：

- 该设备不能有分区。
- 该设备不能有任何 LVM 状态。
- 不得挂载该设备。
- 该设备不得包含文件系统。
- 该设备不得包含 Ceph BlueStore OSD。
- 该设备必须大于 5 GB。

注意

Ceph 不会在不可用的设备上调配 OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 所有管理器和监控守护进程都已部署。

流程

1. 登录到 Cephadm shell：

   示例

```
[root@host01 ~]# cephadm shell
```

2. 列出可用的设备来部署 OSD：

语法

```
ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
```

示例

```
[ceph: root@host01 /]# ceph orch device ls --wide --refresh
```

使用 **--wide** 选项提供与该设备相关的所有详细信息，包括设备可能有资格用作 OSD 的原因。这个选项不支持 NVMe 设备。

3. 可选：要在 **ceph orch device ls** 输出中启用 Health、Ident 和 Failure 字段，请运行以下命令：

注意

libstoragemgmt 库支持这些字段，当前支持 SCSI、SAS 和 SATA 设备。

a. 在 Cephadm shell 外部以 root 用户身份，检查硬件与 **libstoragemgmt** 库的兼容性，以避免出现意外中断服务：

示例

```
[root@host01 ~]# cephadm shell lsmcli ldl
```

在输出中，您会看到 Health Status 为 Good，对应于 SCSI VPD 0x83 ID。

注意

如果没有获取这些信息，启用字段可能会导致设备错误行为。

b. 重新登录 Cephadm shell 并启用 **libstoragemgmt** 支持：

示例

```
[root@host01 ~]# cephadm shell
[ceph: root@host01 /]# ceph config set mgr mgr/cephadm/device_enhanced_scan true
```

启用之后，**ceph orch device ls** 会将 Health 字段的输出设置为 Good。

验证

- 列出设备：

示例

```
[ceph: root@host01 /]# ceph orch device ls
```

6.5. 为 CEPH OSD 部署的 ZAPPING 设备

在部署 OSD 前，您需要检查可用设备列表。如果设备中没有可用空间，可以通过 zapping 来清除设备中的数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 所有管理器和监控守护进程都已部署。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出可用的设备来部署 OSD：

   语法

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

3. 清除设备数据：

   语法

   ```
   ceph orch device zap HOSTNAME FILE_PATH --force
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch device zap host02 /dev/sdb --force
   ```

验证

- 验证该设备中的空间可用：

  示例

  ```
  [ceph: root@host01 /]# ceph orch device ls
  ```

  您将看到 Available 下的字段是 Yes。

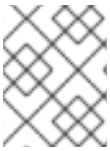
其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的 Listing devices for Ceph OSD deployment 部分。

6.6. 在所有可用设备上部署 CEPH OSD

您可以在所有可用设备上部署所有 OSD。Cephadm 允许 Ceph 编排器在任何可用和未使用的存储设备上发现和部署 OSD。

若要部署 OSD 所有可用的设备，可运行不带 **unmanaged** 参数的命令，然后使用 参数重新运行该命令，以防止创建将来的 OSD。



注意

使用 **--all-available-devices** 部署 OSD 通常用于较小的集群。对于较大的集群，请使用 OSD 规格文件。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 所有管理器和监控守护进程都已部署。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出可用的设备来部署 OSD：

   语法

   ```
   ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch device ls --wide --refresh
   ```

3. 在所有可用设备上部署 OSD：

   示例

   ```
   [ceph: root@host01 /]# ceph orch apply osd --all-available-devices
   ```

   ceph orch apply 的效果具有持久性，这意味着 Orchestrator 会自动找到该设备，将它添加到集群中，并创建新的 OSD。这在出现以下条件时发生：

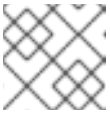
   - 在系统中添加了新的磁盘或驱动器。

- 现有磁盘或驱动器是 zapped。
- OSD 被删除，设备为 zapped。
  您可以使用 **--unmanaged** 参数，禁用在所有可用设备上自动创建 OSD。

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply osd --all-available-devices --unmanaged=true
  ```

  将参数 **--unmanaged** 设置为 **true** 可禁用创建 OSD，如果您应用新的 OSD 服务，也没有更改。

  注意

  命令 **ceph orch daemon add** 会创建新的 OSD，但不添加 OSD 服务。

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 查看节点和设备的详情：

  示例

  ```
  [ceph: root@host01 /]# ceph osd tree
  ```

其它资源

- 请参阅 Red Hat Ceph Storage Operations Guide 中的 Listing devices for Ceph OSD deployment 部分。

6.7. 在特定的设备和主机上部署 CEPH OSD

您可以使用 Ceph 编排器将所有 Ceph OSD 部署到特定的设备和主机上。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 所有管理器和监控守护进程都已部署。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   ```

```
[root@host01 ~]# cephadm shell
```

2. 列出可用的设备来部署 OSD：

语法

```
ceph orch device ls [--hostname=HOSTNAME_1 HOSTNAME_2] [--wide] [--refresh]
```

示例

```
[ceph: root@host01 /]# ceph orch device ls --wide --refresh
```

3. 在特定的设备和主机上部署 OSD：

语法

```
ceph orch daemon add osd HOSTNAME:DEVICE_PATH
```

示例

```
[ceph: root@host01 /]# ceph orch daemon add osd host02:/dev/sdb
```

要在没有 LVM 层的原始物理设备上部署 ODS，请使用 **--method raw** 选项。

语法

```
ceph orch daemon add osd --method raw HOSTNAME:DEVICE_PATH
```

示例

```
[ceph: root@host01 /]# ceph orch daemon add osd --method raw host02:/dev/sdb
```

注意

如果您有单独的 DB 或 WAL 设备，则块与 DB 或 WAL 设备的比例必须为 1:1。

验证

- 列出服务：

示例

```
[ceph: root@host01 /]# ceph orch ls osd
```

- 查看节点和设备的详情：

示例

```
[ceph: root@host01 /]# ceph osd tree
```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --service_name=SERVICE_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --service_name=osd
  ```

其它资源

- 请参阅 Red Hat Ceph Storage Operations Guide 中的 Listing devices for Ceph OSD deployment 部分。

6.8. 用于部署 OSD 的高级服务规格和过滤器

服务规格的 OSD 是利用磁盘属性描述集群布局的方法。它为用户提供了一种抽象的方式，告知 Ceph 哪个磁盘应该切换到带有所需配置的 OSD，而不必了解具体的设备名称和路径。对于每个设备和每个主机，定义 **yaml** 文件或 **json** 文件。

OSD 规格的常规设置

- **service_type**: 'osd': 对于创建 OSDS 是必须的
- **service_id** ：使用您首选的服务名称或标识。使用规范文件创建一组 OSD。此名称用于管理所有 OSD，并且代表一个 Orchestrator 服务。
- **placement** ：用于定义需要在其上部署 OSD 的主机。
  您可以在以下选项中使用：
  - **host_pattern**: '*' - 用于选择主机的主机名称模式。
  - **标签**: 'osd_host' - 需要部署 OSD 的主机中使用的标签。
  - **hosts**: 'host01', 'host02' - 需要部署 OSD 的显式主机名列表。
- **selection of devices** ：创建 OSD 的设备。这样，我们可以将 OSD 与不同的设备分开。您只能创建具有三个组件的 BlueStore OSD：
  - OSD 数据：包含所有 OSD 数据
  - WAL: BlueStore 内部日志或 write-ahead 日志
  - DB: BlueStore 内部元数据
- **data_devices** ：定义要部署 OSD 的设备。在本例中，OSD 在并置的架构中创建。您可以使用过滤器来选择设备和文件夹。
- **wal_devices** ：定义用于 WAL OSD 的设备。您可以使用过滤器来选择设备和文件夹。
- **db_devices** ：定义 DB OSD 的设备。您可以使用过滤器来选择设备和文件夹。
- **encrypted** ：一个可选参数加密 OSD 的信息，它可以设置为 **True** 或 **False**

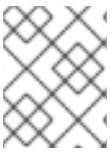
- ***unmanaged***: *可选参数，默认设置为 False。如果您不希望 Orchestrator 来管理 OSD 服务，您可以将其设置为 True。*
- ***block_wal_size*** *：用户定义的值，以字节为单位。*
- ***block_db_size*** *：用户定义的值，以字节为单位。*
- ***osds_per_device*** *：用于为每个设备部署多个 OSD 的用户定义的值。*
- ***方法*** *：一个可选参数，用于指定 OSD 是否使用 LVM 层创建。如果要在不包含 LVM 层的原始物理设备上创建 OSD，设置为* ***raw****。如果您有单独的 DB 或 WAL 设备，则块与 DB 或 WAL 设备的比例必须为 1:1。*

指定设备的过滤器

过滤器与 ***data_devices****、* ***wal_devices*** *和* ***db_devices*** *参数一同使用。*

过滤器的名称	描述	语法	示例
model	目标特定磁盘。您可以通过运行 **lsblk -o NAME,FSTYPE,LABEL,MOUNTPOINT,SIZE,MODEL** 命令或 **smartctl -i /*DEVIVE_PATH*** 来获取模型的详情	Model: *DISK_MODEL_NAME*	model: MC-55-44-XZ
Vendor	特定于目标磁盘	Vendor: *DISK_VENDOR_NAME*	Vendor: Vendor Cs
大小规格	包括精确大小的磁盘	size: *EXACT*	大小： '10G'
大小规格	包括位于范围内的磁盘大小	size: *LOW:HIGH*	大小： '10G:40G'
大小规格	包括小于或等于 size 的磁盘	size: *:HIGH*	大小： ':10G'
大小规格	包括等于或大于 size 的磁盘	大小： *LOW：*	大小： '40G:'
Rotational	磁盘轮转属性。1 与轮转的所有磁盘匹配，0 匹配所有非轮转磁盘。如果 rotational =0，则 OSD 配置有 SSD 或 NVME。如果 rotational=1，则使用 HDD 配置 OSD。	rotational: 0 或 1	rotational: 0
All	考虑所有可用磁盘	all: true	all: true

Limiter	指定有效过滤器后，但希望限制可以使用 'limit' 指令的匹配磁盘的数量。它应仅作为最后的手段使用。	Limit: *NUMBER*	限制：2



注意

要创建在同一主机上带有非并置组件的 OSD，您必须指定使用的不同设备类型，设备应该在同一主机上。

注意

*用于部署 OSD 的设备必须被 **libstoragemgmt** 支持。*

其它资源

- *请参阅 Red Hat Ceph Storage Operations Guide 中的使用高级规格部署 Ceph OSD。*
- *有关 **libstoragemgmt** 的更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的 Ceph OSD 部署列表设备一节。*

6.9. 使用高级服务规格部署 CEPH OSD

类型 OSD 的服务规格是利用磁盘属性描述集群布局的方法。它为用户提供了一种抽象的方式，告知 Ceph 哪个磁盘应该切换到带有所需配置的 OSD，而不必了解具体的设备名称和路径。

*您可以通过定义 **yaml** 文件或 **json** 文件，为每个设备和每个主机部署 OSD。*

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *主机添加到集群中。*
- *所有管理器和监控守护进程都已部署。*

流程

1. *在 monitor 节点上，创建 **osd_spec.yaml** 文件：*

   示例

   ```
   [root@host01 ~]# touch osd_spec.yaml
   ```

2. *编辑 **osd_spec.yaml** 文件，使其包含以下详情：*

   语法

   ```
   service_type: osd
   service_id: SERVICE_ID
   ```

```
placement:
  host_pattern: '*' # optional
data_devices: # optional
  model: DISK_MODEL_NAME # optional
  paths:
  - /DEVICE_PATH
osds_per_device: NUMBER_OF_DEVICES # optional
db_devices: # optional
  size: # optional
  all: true # optional
  paths:
   - /DEVICE_PATH
encrypted: true
```

a. *简单情况：在这些情况下，所有节点都具有相同的设置。*

示例

```
service_type: osd
service_id: osd_spec_default
placement:
  host_pattern: '*'
data_devices:
  all: true
  paths:
  - /dev/sdb
encrypted: true
```

示例

```
service_type: osd
service_id: osd_spec_default
placement:
  host_pattern: '*'
data_devices:
  size: '80G'
db_devices:
  size: '40G:'
  paths:
   - /dev/sdc
```

b. *简单场景：在这种情况下，所有节点都与原始模式中创建的 OSD 设备具有相同的设置，而无需 LVM 层。*

示例

```
service_type: osd
service_id: all-available-devices
encrypted: "true"
method: raw
placement:
  host_pattern: "*"
data_devices:
  all: "true"
```

c. *高级情景：这会将所有 HDD 用作* ***data_devices*****，** *并将两个 SSD 分配为专用 DB 或 WAL 设备来创建所需的布局。剩余的 SSD 是将 NVMEs 供应商分配给专用 DB 或 WAL 设备的* ***data_devices*****。**

示例

```
service_type: osd
service_id: osd_spec_hdd
placement:
  host_pattern: '*'
data_devices:
  rotational: 0
db_devices:
  model: Model-name
  limit: 2
---
service_type: osd
service_id: osd_spec_ssd
placement:
  host_pattern: '*'
data_devices:
  model: Model-name
db_devices:
  vendor: Vendor-name
```

d. *非统一节点的高级场景：这会根据 host_pattern 键将不同的 OSD specs 应用到不同的主机。*

示例

```
service_type: osd
service_id: osd_spec_node_one_to_five
placement:
  host_pattern: 'node[1-5]'
data_devices:
  rotational: 1
db_devices:
  rotational: 0
---
service_type: osd
service_id: osd_spec_six_to_ten
placement:
  host_pattern: 'node[6-10]'
data_devices:
  model: Model-name
db_devices:
  model: Model-name
```

e. *使用专用 WAL 和 DB 设备的高级场景：*

示例

```
service_type: osd
service_id: osd_using_paths
placement:
  hosts:
```

```
    - host01
    - host02
data_devices:
  paths:
    - /dev/sdb
db_devices:
  paths:
    - /dev/sdc
wal_devices:
  paths:
    - /dev/sdd
```

f. *每个设备有多个 OSD 的高级场景：*

示例

```
service_type: osd
service_id: multiple_osds
placement:
  hosts:
    - host01
    - host02
osds_per_device: 4
data_devices:
  paths:
    - /dev/sdb
```

g. *对于预先创建的卷，请编辑* ***osd_spec.yaml*** *文件，使其包含以下详情：*

语法

```
service_type: osd
service_id: SERVICE_ID
placement:
  hosts:
    - HOSTNAME
data_devices: # optional
  model: DISK_MODEL_NAME # optional
  paths:
  - /DEVICE_PATH
db_devices: # optional
  size: # optional
  all: true # optional
  paths:
   - /DEVICE_PATH
```

示例

```
service_type: osd
service_id: osd_spec
placement:
  hosts:
    - machine1
data_devices:
```

```
  paths:
    - /dev/vg_hdd/lv_hdd
db_devices:
  paths:
    - /dev/vg_nvme/lv_nvme
```

h. *对于 OSD，按 ID 编辑* ***osd_spec.yaml*** *文件，使其包含以下详情：*

注意

此配置适用于 Red Hat Ceph Storage 5.3z1 及更新的版本。对于早期版本，请使用预先创建的 lvm。

语法

```
service_type: osd
service_id: OSD_BY_ID_HOSTNAME
placement:
  hosts:
    - HOSTNAME
data_devices: # optional
  model: DISK_MODEL_NAME # optional
  paths:
  - /DEVICE_PATH
db_devices: # optional
  size: # optional
  all: true # optional
  paths:
   - /DEVICE_PATH
```

示例

```
service_type: osd
service_id: osd_by_id_host01
placement:
  hosts:
    - host01
data_devices:
  paths:
    - /dev/disk/by-id/scsi-0QEMU_QEMU_HARDDISK_drive-scsi0-0-0-5
db_devices:
  paths:
    - /dev/disk/by-id/nvme-nvme.1b36-31323334-51454d55204e564d652043747 26c-00000001
```

i. *对于 OSD 通过路径，编辑* ***osd_spec.yaml*** *文件，使其包含以下详情：*

注意

此配置适用于 Red Hat Ceph Storage 5.3z1 及更新的版本。对于早期版本，请使用预先创建的 lvm。

语法

```
service_type: osd
service_id: OSD_BY_PATH_HOSTNAME
placement:
  hosts:
    - HOSTNAME
data_devices: # optional
  model: DISK_MODEL_NAME # optional
  paths:
  - /DEVICE_PATH
db_devices: # optional
  size: # optional
  all: true # optional
  paths:
   - /DEVICE_PATH
```

示例

```
service_type: osd
service_id: osd_by_path_host01
placement:
  hosts:
    - host01
data_devices:
  paths:
    - /dev/disk/by-path/pci-0000:0d:00.0-scsi-0:0:0:4
db_devices:
  paths:
    - /dev/disk/by-path/pci-0000:00:02.0-nvme-1
```

3. 将 YAML 文件挂载到容器中的一个目录下：

示例

```
[root@host01 ~]# cephadm shell --mount osd_spec.yaml:/var/lib/ceph/osd/osd_spec.yaml
```

4. 进入该目录：

示例

```
[ceph: root@host01 /]# cd /var/lib/ceph/osd/
```

5. 在部署 OSD 之前，先执行空运行：

注意

此步骤提供部署预览，无需部署守护进程。

示例

```
[ceph: root@host01 osd]# ceph orch apply -i osd_spec.yaml --dry-run
```

6. 使用服务规格部署 OSD：

语法

```
ceph orch apply -i FILE_NAME.yml
```

示例

```
[ceph: root@host01 osd]# ceph orch apply -i osd_spec.yaml
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls osd
  ```

- 查看节点和设备的详情：

  示例

  ```
  [ceph: root@host01 /]# ceph osd tree
  ```

其它资源

- 请参阅 Red Hat Ceph Storage Operations 指南中的 部署 OSD 的高级服务规格和过滤器。

6.10. 使用 CEPH ORCHESTRATOR 删除 OSD 守护进程

您可以使用 Cephadm 从集群中移除该 OSD。

从集群中移除 OSD 涉及两个步骤：

1. 从集群中撤离所有放置组(PG)。
2. 从集群中移除 PG-free OSD。

--zap 选项删除了卷组、逻辑卷和 LVM 元数据。

注意

在移除 OSD 后，如果 OSD 再次可用，则 **cephadm** 可能会在这些驱动器上自动尝试部署更多 OSD （如果它们与现有 drivegroup 规格匹配）。如果您部署了 OSD，使用 spec 删除，且不想在删除后在驱动器上部署任何新 OSD，请在删除前修改 drivegroup 规格。在部署 OSD 时，如果您使用了 **--all-available-devices** 选项，请设置 **unmanaged: true** 以完全阻止它获取新驱动器。对于其他部署，修改规格。如需了解更多详细信息，请参阅使用高级服务规格部署 Ceph OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。

- Ceph Monitor、Ceph Manager 和 Ceph OSD 守护进程部署在存储集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 检查必须移除 OSD 的设备和节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd tree
   ```

3. 删除 OSD：

   语法

   ```
   ceph orch osd rm OSD_ID [--replace] [--force] --zap
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm 0 --zap
   ```

   注意

   如果您在没有选项的情况下从存储集群中移除 OSD，如 **--replace**，则会完全从存储集群中移除该设备。如果要使用同一设备来部署 OSD，则必须在将其添加到存储集群前首先断开该设备。

4. 可选：要从特定节点中删除多个 OSD，请运行以下命令：

   语法

   ```
   ceph orch osd rm OSD_ID OSD_ID --zap
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm 2 5 --zap
   ```

5. 检查移除 OSD 的状态：

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   OSD  HOST   STATE                    PGS  REPLACE  FORCE  ZAP   DRAIN STARTED AT
   9    host01 done, waiting for purge    0  False    False  True  2023-06-06 17:50:50.525690
   10   host03 done, waiting for purge    0  False    False  True  2023-06-06 17:49:38.731533
   11   host02 done, waiting for purge    0  False    False  True  2023-06-06 17:48:36.641105
   ```

当 OSD 上没有剩余的 PG 时，它会被停用并从集群中移除。

验证

- 验证设备的详细信息以及 Ceph OSD 已从中删除的节点：

  示例

  ```
  [ceph: root@host01 /]# ceph osd tree
  ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的 Deploying Ceph OSDs on all available devices 部分。
- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的在特定设备和主机上部署 CephOSD 部分。
- 有关在设备上清除数据的 zapping 设备的更多信息，请参阅 Red Hat Ceph Storage Operations 指南中的 Zapping devices for Ceph OSD deployment 部分。

6.11. 使用 CEPH ORCHESTRATOR 替换 OSD

当磁盘出现故障时，您可以替换物理存储设备并重复使用相同的 OSD ID，以避免重新配置 CRUSH map。

您可以使用 **--replace** 选项替换集群中的 OSD。

注意

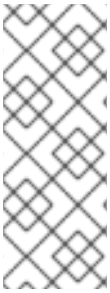
如果要替换单个 OSD，请参阅 在特定设备和主机上部署 Ceph OSD。如果要在所有可用设备上部署 OSD，请参阅 在所有可用设备上部署 Ceph OSD。

此选项使用 **ceph orch rm** 命令保留 OSD ID。OSD 不会从 CRUSH 层次结构中永久移除，而是分配有 **destroyed** 标志。此标志用于确定可在下一个 OSD 部署中重复使用的 OSD ID。**destroyed** 标记用于决定在下一个 OSD 部署中重复使用哪些 OSD ID。

与 **rm** 命令类似，替换集群中的 OSD 涉及两个步骤：

- 从集群中清空所有放置组(PG)。
- 从集群中移除 PG-free OSD。

如果将 OSD 规格用于部署，则新添加的磁盘将被分配其所替换的对应 OSD ID。



注意

移除 OSD 后，如果 OSD 再次部署一次可用，**cephadm** 可能会自动尝试在这些驱动器上部署更多 OSD （如果它们与现有 drivegroup 规格匹配）。如果您部署了 OSD，使用 spec 删除，且不想在删除后在驱动器上部署任何新 OSD，请在删除前修改 drivegroup 规格。在部署 OSD 时，如果您使用了 **--all-available-devices** 选项，请设置 **unmanaged: true** 以完全阻止它获取新驱动器。对于其他部署，修改规格。如需了解更多详细信息，请参阅使用高级服务规格部署 Ceph OSD。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 监控、管理器和 OSD 守护进程部署在存储集群中。
- 必须在同一主机上创建替换已移除 OSD 的新 OSD。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 确保转储并保存 OSD 配置的映射，以备将来参考：

   示例

   ```
   [ceph: root@node /]# ceph osd metadata -f plain | grep device_paths
   "device_paths": "sde=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:1,sdi=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:1",
   "device_paths": "sde=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:1,sdf=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:1",
   "device_paths": "sdd=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:2,sdg=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:2",
   "device_paths": "sdd=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:2,sdh=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:2",
   "device_paths": "sdd=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:2,sdk=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:2",
   "device_paths": "sdc=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:3,sdl=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:3",
   "device_paths": "sdc=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:3,sdj=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:3",
   "device_paths": "sdc=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:0:3,sdm=/dev/disk/by-path/pci-0000:03:00.0-scsi-0:1:0:3",
   [.. output omitted ..]
   ```

3. 检查必须替换 OSD 的设备和节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd tree
   ```

4. 替换 OSD：

   重要

   如果存储集群关联了 **health_warn** 或其他错误，请在替换 OSD 前尝试修复任何错误，以避免数据丢失。

语法

```
ceph orch osd rm OSD_ID --replace [--force]
```

当存储集群有持续操作时，可以使用 **--force** 选项。

示例

```
[ceph: root@host01 /]# ceph orch osd rm 0 --replace
```

5. 检查 OSD 替换的状态：

示例

```
[ceph: root@host01 /]# ceph orch osd rm status
```

6. 停止编配器以应用任何现有 OSD 规格：

示例

```
[ceph: root@node /]# ceph orch pause
[ceph: root@node /]# ceph orch status
Backend: cephadm
Available: Yes
Paused: Yes
```

7. zap 已删除的 OSD 设备：

示例

```
[ceph: root@node /]# ceph orch device zap node.example.com /dev/sdi --force
zap successful for /dev/sdi on node.example.com

[ceph: root@node /]# ceph orch device zap node.example.com /dev/sdf --force
zap successful for /dev/sdf on node.example.com
```

8. 从暂停模式恢复 Orcestrator

示例

```
[ceph: root@node /]# ceph orch resume
```

9. 检查 OSD 替换的状态：

示例

```
[ceph: root@node /]# ceph osd tree
ID  CLASS  WEIGHT   TYPE NAME      STATUS  REWEIGHT  PRI-AFF
-1         0.77112  root default
-3         0.77112      host node
 0    hdd  0.09639          osd.0      up   1.00000  1.00000
 1    hdd  0.09639          osd.1      up   1.00000  1.00000
```

```
2   hdd  0.09639        osd.2      up   1.00000  1.00000
3   hdd  0.09639        osd.3      up   1.00000  1.00000
4   hdd  0.09639        osd.4      up   1.00000  1.00000
5   hdd  0.09639        osd.5      up   1.00000  1.00000
6   hdd  0.09639        osd.6      up   1.00000  1.00000
7   hdd  0.09639        osd.7      up   1.00000  1.00000
[.. output omitted ..]
```

验证

- *验证设备的详细信息以及 Ceph OSD 所取代的节点：*

  示例

  ```
  [ceph: root@host01 /]# ceph osd tree
  ```

  您可以看到与您在同一主机上运行相同的 id 的 OSD。

- *验证新部署的 OSD 的* ***db_device*** *是否为所取代的* ***db_device*** *：*

  示例

  ```
  [ceph: root@host01 /]# ceph osd metadata 0 | grep bluefs_db_devices
  "bluefs_db_devices": "nvme0n1",

  [ceph: root@host01 /]# ceph osd metadata 1 | grep bluefs_db_devices
  "bluefs_db_devices": "nvme0n1",
  ```

其它资源

- *如需更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的 Deploying Ceph OSDs on all available devices 部分。*
- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的在特定设备和主机上部署 CephOSD 部分。*

6.12. 将 OSD 替换为预先创建的 LVM

使用 ***ceph-volume lvm zap*** *命令清除 OSD 后，如果目录不存在，您可以将 OSD 替换为 OSd 服务规格文件，并预先创建的 LVM。*

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *失败的 OSD*

流程

1. *登录到 Cephadm shell：*

   示例

```
[root@host01 ~]# cephadm shell
```

2. *删除 OSD：*

语法

```
ceph orch osd rm OSD_ID [--replace]
```

示例

```
[ceph: root@host01 /]# ceph orch osd rm 8 --replace
Scheduled OSD(s) for removal
```

3. *验证 OSD 是否已销毁：*

示例

```
[ceph: root@host01 /]# ceph osd tree

ID   CLASS  WEIGHT   TYPE NAME       STATUS     REWEIGHT  PRI-AFF
 -1         0.32297  root default
 -9         0.05177      host host10
  3    hdd  0.01520          osd.3          up   1.00000  1.00000
 13    hdd  0.02489          osd.13         up   1.00000  1.00000
 17    hdd  0.01169          osd.17         up   1.00000  1.00000
-13         0.05177      host host11
  2    hdd  0.01520          osd.2          up   1.00000  1.00000
 15    hdd  0.02489          osd.15         up   1.00000  1.00000
 19    hdd  0.01169          osd.19         up   1.00000  1.00000
 -7         0.05835      host host12
 20    hdd  0.01459          osd.20         up   1.00000  1.00000
 21    hdd  0.01459          osd.21         up   1.00000  1.00000
 22    hdd  0.01459          osd.22         up   1.00000  1.00000
 23    hdd  0.01459          osd.23         up   1.00000  1.00000
 -5         0.03827      host host04
  1    hdd  0.01169          osd.1          up   1.00000  1.00000
  6    hdd  0.01129          osd.6          up   1.00000  1.00000
  7    hdd  0.00749          osd.7          up   1.00000  1.00000
  9    hdd  0.00780          osd.9          up   1.00000  1.00000
 -3         0.03816      host host05
  0    hdd  0.01169          osd.0          up   1.00000  1.00000
  8    hdd  0.01129          osd.8    destroyed        0  1.00000
 12    hdd  0.00749          osd.12         up   1.00000  1.00000
 16    hdd  0.00769          osd.16         up   1.00000  1.00000
-15         0.04237      host host06
  5    hdd  0.01239          osd.5          up   1.00000  1.00000
 10    hdd  0.01540          osd.10         up   1.00000  1.00000
 11    hdd  0.01459          osd.11         up   1.00000  1.00000
-11         0.04227      host host07
  4    hdd  0.01239          osd.4          up   1.00000  1.00000
 14    hdd  0.01529          osd.14         up   1.00000  1.00000
 18    hdd  0.01459          osd.18         up   1.00000  1.00000
```

4. *使用 **ceph-volume** 命令切换并删除 OSD：*

语法

```
ceph-volume lvm zap --osd-id OSD_ID
```

示例

```
[ceph: root@host01 /]# ceph-volume lvm zap --osd-id 8

Zapping: /dev/vg1/data-lv2
Closing encrypted path /dev/mapper/l4D6ql-Prji-IzH4-dfhF-xzuf-5ETl-jNRcXC
Running command: /usr/sbin/cryptsetup remove /dev/mapper/l4D6ql-Prji-IzH4-dfhF-xzuf-5ETl-jNRcXC
Running command: /usr/bin/dd if=/dev/zero of=/dev/vg1/data-lv2 bs=1M count=10 conv=fsync
 stderr: 10+0 records in
10+0 records out
 stderr: 10485760 bytes (10 MB, 10 MiB) copied, 0.034742 s, 302 MB/s
Zapping successful for OSD: 8
```

5. 检查 OSD 拓扑：

示例

```
[ceph: root@host01 /]# ceph-volume lvm list
```

6. 使用与该特定 OSD 拓扑对应的规格文件重新创建 OSD：

示例

```
[ceph: root@host01 /]# cat osd.yml
service_type: osd
service_id: osd_service
placement:
  hosts:
  - host03
data_devices:
  paths:
  - /dev/vg1/data-lv2
db_devices:
  paths:
   - /dev/vg1/db-lv1
```

7. 应用更新的规格文件：

示例

```
[ceph: root@host01 /]# ceph orch apply -i osd.yml
Scheduled osd.osd_service update...
```

8. 验证 OSD 是否返回：

示例

```
[ceph: root@host01 /]# ceph -s
[ceph: root@host01 /]# ceph osd tree
```

6.13. 在非并置场景中替换 OSD

当 OSD 在非并置场景中失败时，您可以替换 WAL/DB 设备。DB 和 WAL 设备的步骤相同。您需要为 DB 设备编辑 **db_devices** 下的 **paths**，为 WAL 设备编辑 **wal_devices** 下的 **paths**。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 守护进程是非并置的。
- 失败的 OSD

流程

1. 识别集群中的设备：

   示例

```
[root@host01 ~]# lsblk

NAME                                                  MAJ:MIN RM  SIZE RO
TYPE MOUNTPOINT
sda                                                   8:0    0   20G  0 disk
├─sda1                                                8:1    0    1G  0 part
/boot
└─sda2                                                8:2    0   19G  0 part
  ├─rhel-root                                         253:0    0   17G  0 lvm  /
  └─rhel-swap                                         253:1    0    2G  0 lvm
[SWAP]
sdb                                                   8:16   0   10G  0 disk
└─ceph--5726d3e9--4fdb--4eda--b56a--3e0df88d663f-osd--block--3ceb89ec--87ef--46b4--
99c6--2a56bac09ff0 253:2    0   10G  0 lvm
sdc                                                   8:32   0   10G  0 disk
└─ceph--d7c9ab50--f5c0--4be0--a8fd--e0313115f65c-osd--block--37c370df--1263--487f--
a476--08e28bdbcd3c 253:4    0   10G  0 lvm
sdd                                                   8:48   0   10G  0 disk
├─ceph--1774f992--44f9--4e78--be7b--b403057cf5c3-osd--db--31b20150--4cbc--4c2c--
9c8f--6f624f3bfd89    253:7    0  2.5G  0 lvm
└─ceph--1774f992--44f9--4e78--be7b--b403057cf5c3-osd--db--1bee5101--dbab--4155--
a02c--e5a747d38a56    253:9    0  2.5G  0 lvm
sde                                                   8:64   0   10G  0 disk
sdf                                                   8:80   0   10G  0 disk
└─ceph--412ee99b--4303--4199--930a--0d976e1599a2-osd--block--3a99af02--7c73--4236--
9879--1fad1fe6203d 253:6    0   10G  0 lvm
sdg                                                   8:96   0   10G  0 disk
└─ceph--316ca066--aeb6--46e1--8c57--f12f279467b4-osd--block--58475365--51e7--42f2--
9681--e0c921947ae6 253:8    0   10G  0 lvm
sdh                                                   8:112  0   10G  0 disk
├─ceph--d7064874--66cb--4a77--a7c2--8aa0b0125c3c-osd--db--0dfe6eca--ba58--438a--
9510--d96e6814d853    253:3    0    5G  0 lvm
```

```
└─ceph--d7064874--66cb--4a77--a7c2--8aa0b0125c3c-osd--db--26b70c30--8817--45de--
8843--4c0932ad2429    253:5    0    5G  0 lvm
sr0
```

2. *登录到 Cephadm shell：*

示例

```
[root@host01 ~]# cephadm shell
```

3. *识别 OSD 及其 DB 设备：*

示例

```
[ceph: root@host01 /]# ceph-volume lvm list /dev/sdh


====== osd.2 =======

  [db]          /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-db-0dfe6eca-ba58-
438a-9510-d96e6814d853

      block device              /dev/ceph-5726d3e9-4fdb-4eda-b56a-3e0df88d663f/osd-block-
3ceb89ec-87ef-46b4-99c6-2a56bac09ff0
      block uuid                GkWLoo-f0jd-Apj2-Zmwj-ce0h-OY6J-UuW8aD
      cephx lockbox secret
      cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
      cluster name              ceph
      crush device class
      db device                 /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-db-
0dfe6eca-ba58-438a-9510-d96e6814d853
      db uuid                   6gSPoc-L39h-afN3-rDl6-kozT-AX9S-XR20xM
      encrypted                 0
      osd fsid                  3ceb89ec-87ef-46b4-99c6-2a56bac09ff0
      osd id                    2
      osdspec affinity          non-colocated
      type                      db
      vdo                       0
      devices                   /dev/sdh

====== osd.5 =======

  [db]          /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-db-26b70c30-8817-
45de-8843-4c0932ad2429

      block device              /dev/ceph-d7c9ab50-f5c0-4be0-a8fd-e0313115f65c/osd-block-
37c370df-1263-487f-a476-08e28bdbcd3c
      block uuid                Eay3I7-fcz5-AWvp-kRcI-mJaH-n03V-Zr0wmJ
      cephx lockbox secret
      cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
      cluster name              ceph
      crush device class
      db device                 /dev/ceph-d7064874-66cb-4a77-a7c2-8aa0b0125c3c/osd-db-
26b70c30-8817-45de-8843-4c0932ad2429
      db uuid                   mwSohP-u72r-DHcT-BPka-piwA-lSwx-w24N0M
```

```
    encrypted              0
    osd fsid               37c370df-1263-487f-a476-08e28bdbcd3c
    osd id                 5
    osdspec affinity       non-colocated
    type                   db
    vdo                    0
    devices                /dev/sdh
```

4. *在 **osds.yaml** 文件中，将 **unmanaged** 参数设置为 **true**，否则 **cephadm** 会重新部署 OSD：*

   示例

```
[ceph: root@host01 /]# cat osds.yml
service_type: osd
service_id: non-colocated
unmanaged: true
placement:
  host_pattern: 'ceph*'
data_devices:
  paths:
   - /dev/sdb
   - /dev/sdc
   - /dev/sdf
   - /dev/sdg
db_devices:
  paths:
   - /dev/sdd
   - /dev/sdh
```

5. *应用更新的规格文件：*

   示例

```
[ceph: root@host01 /]# ceph orch apply -i osds.yml

Scheduled osd.non-colocated update...
```

6. *检查状态：*

   示例

```
[ceph: root@host01 /]# ceph orch ls

NAME           PORTS        RUNNING  REFRESHED  AGE  PLACEMENT
alertmanager   ?:9093,9094      1/1  9m ago     4d   count:1
crash                           3/4  4d ago     4d   *
grafana        ?:3000           1/1  9m ago     4d   count:1
mgr                             1/2  4d ago     4d   count:2
mon                             3/5  4d ago     4d   count:5
node-exporter  ?:9100           3/4  4d ago     4d   *
osd.non-colocated               8  4d ago     5s   <unmanaged>
prometheus     ?:9095           1/1  9m ago     4d   count:1
```

7. *移除 OSD。确保使用 **--zap** 选项删除 hte backend 服务和 **--replace** 选项来保留 OSD ID：*

示例

```
[ceph: root@host01 /]# ceph orch osd rm 2 5 --zap --replace
Scheduled OSD(s) for removal
```

8. 检查状态：

示例

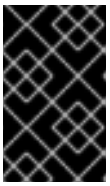
```
[ceph: root@host01 /]# ceph osd df tree | egrep -i "ID|host02|osd.2|osd.5"

ID  CLASS  WEIGHT   REWEIGHT  SIZE     RAW USE  DATA     OMAP  META     AVAIL
%USE   VAR   PGS  STATUS     TYPE NAME
-5         0.04877         -   55 GiB   15 GiB  4.1 MiB   0 B   60 MiB  40 GiB  27.27  1.17    -
host02
 2    hdd  0.01219   1.00000   15 GiB  5.0 GiB  996 KiB   0 B   15 MiB  10 GiB  33.33  1.43    0
destroyed          osd.2
 5    hdd  0.01219   1.00000   15 GiB  5.0 GiB  1.0 MiB   0 B   15 MiB  10 GiB  33.33  1.43    0
destroyed          osd.5
```

9. 编辑 **osds.yaml** 规格文件，将 **unmanaged** 参数改为 **false**，并在设备物理替换后替换 DB 设备的路径：

示例

```
[ceph: root@host01 /]# cat osds.yml
service_type: osd
service_id: non-colocated
unmanaged: false
placement:
  host_pattern: 'ceph01*'
data_devices:
  paths:
   - /dev/sdb
   - /dev/sdc
   - /dev/sdf
   - /dev/sdg
db_devices:
  paths:
   - /dev/sdd
   - /dev/sde
```

在上例中，**/dev/sdh** 替换为 **/dev/sde**。



重要

如果您使用同一主机规格文件替换单个 OSD 节点上的故障 DB 设备，请修改 **host_pattern** 选项以仅指定 OSD 节点，否则部署会失败，您无法在其他主机上找到新的 DB 设备。

10. 使用 **--dry-run** 选项重新应用规格文件，以确保 OSD 应该使用新的 DB 设备部署：

示例

-

```
[ceph: root@host01 /]# ceph orch apply -i osds.yml --dry-run
WARNING! Dry-Runs are snapshots of a certain point in time and are bound
to the current inventory setup. If any of these conditions change, the
preview will be invalid. Please make sure to have a minimal
timeframe between planning and applying the specs.
####################
SERVICESPEC PREVIEWS
####################
+---------+------+--------+-------------+
|SERVICE  |NAME  |ADD_TO  |REMOVE_FROM  |
+---------+------+--------+-------------+
+---------+------+--------+-------------+
#################
OSDSPEC PREVIEWS
#################
+---------+-------+-------+----------+----------+-----+
|SERVICE  |NAME   |HOST   |DATA      |DB        |WAL  |
+---------+-------+-------+----------+----------+-----+
|osd      |non-colocated  |host02  |/dev/sdb  |/dev/sde  |-    |
|osd      |non-colocated  |host02  |/dev/sdc  |/dev/sde  |-    |
+---------+-------+-------+----------+----------+-----+
```

11. *应用规格文件：*

示例

```
[ceph: root@host01 /]# ceph orch apply -i osds.yml
Scheduled osd.non-colocated update...
```

12. *检查 OSD 是否已重新部署：*

示例

```
[ceph: root@host01 /]# ceph osd df tree | egrep -i "ID|host02|osd.2|osd.5"

ID  CLASS  WEIGHT   REWEIGHT  SIZE     RAW USE  DATA     OMAP  META     AVAIL
%USE   VAR   PGS  STATUS  TYPE NAME
-5         0.04877         -   55 GiB   15 GiB  4.5 MiB   0 B   60 MiB  40 GiB  27.27  1.17    -
host host02
 2    hdd  0.01219   1.00000   15 GiB  5.0 GiB  1.1 MiB   0 B   15 MiB  10 GiB  33.33  1.43    0
up          osd.2
 5    hdd  0.01219   1.00000   15 GiB  5.0 GiB  1.1 MiB   0 B   15 MiB  10 GiB  33.33  1.43    0
up          osd.5
```

验证

1. *在重新部署 OSDS 的 OSD 主机中，验证它们是否在新的 DB 设备上：*

示例

```
[ceph: root@host01 /]# ceph-volume lvm list /dev/sde

====== osd.2 =======
```

```
  [db]          /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-db-1998a02e-5e67-
42a9-b057-e02c22bbf461

      block device              /dev/ceph-a4afcb78-c804-4daf-b78f-3c7ad1ed0379/osd-block-
564b3d2f-0f85-4289-899a-9f98a2641979
      block uuid                ITPVPa-CCQ5-BbFa-FZCn-FeYt-c5N4-ssdU41
      cephx lockbox secret
      cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
      cluster name              ceph
      crush device class
      db device                 /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-db-
1998a02e-5e67-42a9-b057-e02c22bbf461
      db uuid                   HF1bYb-fTK7-0dcB-CHzW-xvNn-dCym-KKdU5e
      encrypted                 0
      osd fsid                  564b3d2f-0f85-4289-899a-9f98a2641979
      osd id                    2
      osdspec affinity          non-colocated
      type                      db
      vdo                       0
      devices                   /dev/sde

====== osd.5 =======

  [db]          /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-db-6c154191-846d-
4e63-8c57-fc4b99e182bd

      block device              /dev/ceph-b37c8310-77f9-4163-964b-f17b4c29c537/osd-block-
b42a4f1f-8e19-4416-a874-6ff5d305d97f
      block uuid                0LuPoz-ao7S-UL2t-BDIs-C9pl-ct8J-xh5ep4
      cephx lockbox secret
      cluster fsid              fa0bd9dc-e4c4-11ed-8db4-001a4a00046e
      cluster name              ceph
      crush device class
      db device                 /dev/ceph-15ce813a-8a4c-46d9-ad99-7e0845baf15e/osd-db-
6c154191-846d-4e63-8c57-fc4b99e182bd
      db uuid                   SvmXms-iWkj-MTG7-VnJj-r5Mo-Moiw-MsbqVD
      encrypted                 0
      osd fsid                  b42a4f1f-8e19-4416-a874-6ff5d305d97f
      osd id                    5
      osdspec affinity          non-colocated
      type                      db
      vdo                       0
      devices                   /dev/sde
```

6.14. 使用 CEPH 编排器停止移除 OSD

您可以停止仅删除排队以进行移除的 OSD。这会重置 OSD 的初始状态，并将它移除队列关闭。

如果 OSD 处于移除过程中，则无法停止该进程。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *主机添加到集群中。*

- 在集群中部署 monitor、Manager 和 OSD 守护进程。
- 移除启动 OSD 进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 检查要移除 OSD 的设备和节点：

   示例

   ```
   [ceph: root@host01 /]# ceph osd tree
   ```

3. 停止删除已排队的 OSD：

   语法

   ```
   ceph orch osd rm stop OSD_ID
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm stop 0
   ```

4. 检查移除 OSD 的状态：

   示例

   ```
   [ceph: root@host01 /]# ceph orch osd rm status
   ```

验证

- 验证已排队 Ceph OSD 以进行移除的设备和节点的详细信息：

  示例

  ```
  [ceph: root@host01 /]# ceph osd tree
  ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的使用 Ceph Orchestrator 删除 OSD 守护进程。

6.15. 使用 CEPH ORCHESTRATOR 激活 OSD

在重新安装主机的操作系统时，您可以激活集群中的 OSD。

先决条件

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 监控、管理器和 OSD 守护进程部署在存储集群中。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 重新安装主机操作系统后，激活 OSD：

   语法

   ```
   ceph cephadm osd activate HOSTNAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph cephadm osd activate host03
   ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --service_name=SERVICE_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --service_name=osd
  ```

6.16. 观察数据迁移

将 OSD 添加到 CRUSH map 时，Ceph 开始通过将放置组迁移到新的或现有的 OSD 来重新平衡数据。您可以使用 **ceph-w** 命令观察数据迁移。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- 最近添加或删除 OSD。

流程

1. 观察数据迁移：

   示例

   ```
   [ceph: root@host01 /]# ceph -w
   ```

2. 在迁移完成后，观察放置组状态从 **active+clean** 变为 **active, some degraded objects**，最终变为 **active+clean**。

3. 要退出，请按 **Ctrl + C**。

6.17. 重新计算放置组

放置组(PG)定义将任何池数据分散到可用的 OSD 中。放置组基于要使用的给定冗余算法构建。对于三向复制，抗压定义冗余来使用 3 个不同的 OSD。对于纠删代码池，要使用的 OSD 数量由块数目定义。

在定义池数量时，放置组的数量定义了粒度的评分，数据会分散到所有可用的 OSD 中。容量负载相等的数量越高。但是，由于处理放置组在重组数据时也很重要，因此当重新构建数据时，这个数字非常重要。支持计算工具，可生成敏捷环境。

在存储池的生命周期内可能会超过最初的预期限制。当驱动器数量增加时，建议进行重新计算。每个 OSD 的 PG 数量应当大约为 100。向存储集群添加更多 OSD 时，每个 OSD 的 PG 数量会随着时间降低。从存储集群中最初使用 120 个驱动器，将池的 **pg_num** 设置为 4000 的结果是每个 OSD 有 100 个 PG（复制因子为三）。随着时间的推移，当增加到 OSD 数量的十倍时，每个 OSD 的 PG 数量将仅下降到十个。由于每个 OSD 数量少量的 PG 往往不会均匀分布容量，因此请考虑调整每个池的 PG。

可以在线调整放置组的数量。重新计算 PG 数值不仅会重新计算 PG 数量，而且会涉及数据重定位，该过程会是一个冗长的过程。但是，数据可用性可以随时维护。

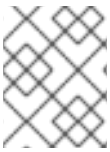
应避免每个 OSD 有大量 PG，因为对一个有故障的 OSD 上的所有 PG 进行重新构建将会一次启动。需要及时重新构建方法（可能不可用）执行大量 IOPS。这会导致深度 I/O 队列和高延迟渲染存储集群不可用，或者会导致长时间修复时间。

其它资源

- 请参阅 PG 计算器，以计算给定用例的值。
- 如需更多信息，请参阅 Red Hat Ceph Storage 策略指南中的 Erasure Code 池 一章。

第 7 章 使用 CEPH ORCHESTRATOR 管理监控堆栈

作为存储管理员，您可以在后端中将 Ceph Orchestrator 与 Cephadm 搭配使用，以部署监控和警报堆栈。监控堆栈由 Prometheus、Prometheus 导出器、Prometheus Alertmanager 和 Grafana 组成。用户需要在 YAML 配置文件中通过 Cephadm 定义这些服务，或者可以使用 命令行界面来部署这些服务。当部署了同一类型的多个服务时，会部署高可用性设置。节点 exporter 是此规则的一个例外。



注意

Red Hat Ceph Storage 不支持自定义镜像来部署监控服务，如 Prometheus、Grafana、Alertmanager 和 node-exporter。

以下监控服务可以通过 Cephadm 部署：

- Prometheus 是监控和警报工具包。它收集 Prometheus exporters 提供的数据，并在达到预定义的阈值时触发预配置的警报。Prometheus manager 模块提供了一个 Prometheus exporter，用于传递 **ceph-mgr** 中的集合点的 Ceph 性能计数器。
  Prometheus 配置，包括提取目标（如提供守护进程的指标）由 Cephadm 自动设置。Cephadm 还部署默认警报的列表，如健康错误、10 个 OSD down 或 pgs inactive。

- Alertmanager 处理 Prometheus 服务器发送的警报。它取消复制、组并将警报路由到正确的接收器。默认情况下，Ceph 仪表板自动配置为接收器。Alertmanager 处理 Prometheus 服务器发送的警报。可以使用 Alertmanager 静默警报，但也可以使用 Ceph 控制面板管理静默。

- Grafana 是一个视觉化和警报软件。此监控堆栈不使用 Grafana 的警报功能。对于警报，使用了 Alertmanager。
  默认情况下，到 Grafana 的流量会通过 TLS 加密。您可以提供自己的 TLS 证书，也可以使用自签名证书。如果在部署 Grafana 前没有配置自定义证书，则会自动为 Grafana 创建和配置自签名证书。Grafana 的自定义证书可通过以下命令配置：

  语法

  ```
  ceph config-key set mgr/cephadm/HOSTNAME/grafana_key -i
  PRESENT_WORKING_DIRECTORY/key.pem
  ceph config-key set mgr/cephadm/HOSTNAME/grafana_crt -i
  PRESENT_WORKING_DIRECTORY/certificate.pem
  ```

节点导出器是 Prometheus 的导出器，提供有关安装它的节点的数据。建议您在所有节点上安装节点导出器。这可以使用带有 node-exporter 服务类型的 **monitoring.yml** 文件来实现。

7.1. 使用 CEPH ORCHESTRATOR 部署监控堆栈

监控堆栈由 Prometheus、Prometheus 导出器、Prometheus Alertmanager、Grafana 和 Ceph Exporter 组成。Ceph 控制面板利用这些组件在集群使用和性能上存储和视觉化详细指标。

您可以使用 YAML 文件格式的服务规格部署监控堆栈。所有监控服务都可以具有其绑定到 **yml** 文件中的网络和端口。

先决条件

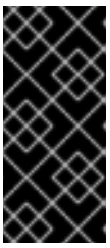
- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对节点的根级别访问权限。

流程

1. 在 Ceph Manager 守护进程中启用 prometheus 模块。这会公开内部 Ceph 指标，以便 Prometheus 可以读取它们：

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module enable prometheus
   ```

   重要

   确保在部署 Prometheus 前运行这个命令。如果在部署前运行该命令，您必须重新部署 Prometheus 以更新配置：

   ```
   ceph orch redeploy prometheus
   ```

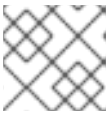
2. 进入以下目录：

   语法

   ```
   cd /var/lib/ceph/DAEMON_PATH/
   ```

   示例

   ```
   [ceph: root@host01 mds/]# cd /var/lib/ceph/monitoring/
   ```

   注意

   如果目录 **monitoring** 不存在，则会创建它。

3. 创建 **monitoring.yml** 文件：

   示例

   ```
   [ceph: root@host01 monitoring]# touch monitoring.yml
   ```

4. 使用类似以下示例的内容编辑规格文件：

   示例

   ```
   service_type: prometheus
   service_name: prometheus
   placement:
     hosts:
     - host01
   networks:
   - 192.169.142.0/24
   ---
   service_type: node-exporter
   ---
   service_type: alertmanager
   ```

```
service_name: alertmanager
placement:
  hosts:
  - host01
networks:
- 192.169.142.0/24
---
service_type: grafana
service_name: grafana
placement:
  hosts:
  - host01
networks:
- 192.169.142.0/24
---
service_type: ceph-exporter
```

注意

确保监控堆栈组件 ***alertmanager***、***prometheus*** *和* ***grafana*** *部署到同一主机上。应该在所有主机上部署* ***node-exporter*** *和* ***ceph-exporter*** *组件。*

5. *应用监控服务：*

示例

```
[ceph: root@host01 monitoring]# ceph orch apply -i monitoring.yml
```

验证

- *列出服务：*

示例

```
[ceph: root@host01 /]# ceph orch ls
```

- *列出主机、守护进程和进程：*

语法

```
ceph orch ps --service_name=SERVICE_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch ps --service_name=prometheus
```

重要

Prometheus、Grafana 和 Ceph 仪表板都会自动配置为相互通信，从而导致 Ceph 仪表板中完全正常工作的 Grafana 集成。

7.2. 使用 CEPH ORCHESTRATOR 删除监控堆栈

您可以使用 **ceph orch rm** 命令删除监控堆栈。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 使用 **ceph orch rm** 命令删除监控堆栈：

   语法

   ```
   ceph orch rm SERVICE_NAME --force
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch rm grafana
   [ceph: root@host01 /]# ceph orch rm prometheus
   [ceph: root@host01 /]# ceph orch rm node-exporter
   [ceph: root@host01 /]# ceph orch rm ceph-exporter
   [ceph: root@host01 /]# ceph orch rm alertmanager
   [ceph: root@host01 /]# ceph mgr module disable prometheus
   ```

3. 检查进程的状态：

   示例

   ```
   [ceph: root@host01 /]# ceph orch status
   ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps
  ```

  示例

```
[ceph: root@host01 /]# ceph orch ps
```

其它资源

- 请参阅 Red Hat Ceph Storage Operations Guide 中的使用 Ceph Orchestrator 部署监控堆栈部分。

第 8 章 基本 RED HAT CEPH STORAGE 客户端设置

作为存储管理员，您必须使用基本配置设置客户端机器，才能与存储集群交互。大多数客户端机器只需要安装 **ceph-common** 软件包及其依赖项。它将提供基本的 **ceph** 和 **rados** 命令，以及 **mount.ceph** 和 **rbd** 等其他命令。

8.1. 在客户端机器上配置文件设置

客户端机器通常需要比功能完整的存储群集成员更小的配置文件。您可以生成最小的配置文件，向客户端提供详细信息以到达 Ceph 监视器。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根访问节点。

流程

1. 在您要设置文件的节点上，在 **/etc** 文件夹中创建一个目录 **ceph** ：

   示例

   ```
   [root@host01 ~]# mkdir /etc/ceph/
   ```

2. 进入 **/etc/ceph** 目录：

   示例

   ```
   [root@host01 ~]# cd /etc/ceph/
   ```

3. 在 **ceph** 目录中生成配置文件：

   示例

   ```
   [root@host01 ceph]# ceph config generate-minimal-conf

   # minimal ceph.conf for 417b1d7a-a0e6-11eb-b940-001a4a000740
   [global]
    fsid = 417b1d7a-a0e6-11eb-b940-001a4a000740
    mon_host = [v2:10.74.249.41:3300/0,v1:10.74.249.41:6789/0]
   ```

   此文件的内容应安装在 **/etc/ceph/ceph.conf** 路径中。您可以使用此配置文件来访问 Ceph 监视器。

8.2. 在客户端机器上设置密钥环

大多数 Ceph 集群都在启用身份验证的情况下运行，客户端需要密钥才能与集群机器通信。您可以生成密钥环，以向客户端提供访问 Ceph 监视器的详细信息。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 根访问节点。

流程

1. 在您要设置密钥环的节点上，在 **/etc** 文件夹中创建一个目录 **ceph** ：

   示例

   ```
   [root@host01 ~]# mkdir /etc/ceph/
   ```

2. 进入 **ceph** 目录中的 **/etc/ceph** 目录：

   示例

   ```
   [root@host01 ~]# cd /etc/ceph/
   ```

3. 为客户端生成密钥环：

   语法

   ```
   ceph auth get-or-create client.CLIENT_NAME -o /etc/ceph/NAME_OF_THE_FILE
   ```

   示例

   ```
   [root@host01 ceph]# ceph auth get-or-create client.fs -o /etc/ceph/ceph.keyring
   ```

4. 验证 **ceph.keyring** 文件中的输出：

   示例

   ```
   [root@host01 ceph]# cat ceph.keyring

   [client.fs]
    key = AQAvoH5gkUCsExAATz3xCBLd4n6B6jRv+Z7CVQ==
   ```

   生成的输出应放入密钥环文件中，如 **/etc/ceph/ceph.keyring**。

第 9 章 使用 CEPH ORCHESTRATOR 管理 MDS 服务

作为存储管理员，您可以在后端中使用 Ceph 编排器和 Cephadm 部署 MDS 服务。默认情况下，Ceph 文件系统(CephFS)仅使用了一个活跃的 MDS 守护进程。但是，具有许多客户端的系统得益于多个活跃的 MDS 守护进程。

本节涵盖了以下管理任务：

- 使用命令行界面 部署 MDS 服务。
- 使用服务规格部署 MDS 服务。
- 使用 Ceph 编排器删除 MDS 服务。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。

9.1. 使用命令行界面部署 MDS 服务

通过使用 Ceph 编排器，您可以使用命令行界面中的 **placement** 规格部署元数据服务器(MDS)服务。Ceph 文件系统(CephFS)需要一个或多个 MDS。

注意

确保至少有一个池，一个用于 Ceph 文件系统(CephFS)数据，另一个用于 CephFS 元数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控和 OSD 守护进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

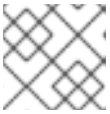
2. 使用放置规格部署 MDS 守护进程有两种方法：

方法 1

- 使用 **ceph fs volume** 来创建 MDS 守护进程。这将创建 CephFS 卷和与 CephFS 关联的池，也会在主机上启动 MDS 服务。

  语法

  ```
  ceph fs volume create FILESYSTEM_NAME --placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
  ```

  注意

  默认情况下，为此命令创建池。

  示例

  ```
  [ceph: root@host01 /]# ceph fs volume create test --placement="2 host01 host02"
  ```

方法 2

- 创建池 CephFS，然后使用放置规格部署 MDS 服务：

  a. 为 CephFS 创建池：

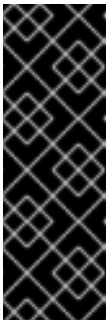
  语法

  ```
  ceph osd pool create DATA_POOL [PG_NUM]
  ceph osd pool create METADATA_POOL [PG_NUM]
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph osd pool create cephfs_data 64
  [ceph: root@host01 /]# ceph osd pool create cephfs_metadata 64
  ```

  通常，元数据池可以从保守的 PG 数量开始，因为它的对象通常比数据池少得多。如果需要，可以增加 PG 数量。池大小范围从 64 个 PG 到 512 个 PG。数据池的大小与您文件系统中预期的文件的编号和大小成比例。

  重要

  对于元数据池，请考虑使用：

  - 更高的复制级别，因为对此池的任何数据丢失都可能会导致整个文件系统无法访问。
  - 延迟较低的存储（如 Solid-State Drive(SSD)磁盘），因为这会直接影响客户端上观察到的文件系统操作延迟。

  b. 为数据池和元数据池创建文件系统：

  语法

  ```
  ceph fs new FILESYSTEM_NAME METADATA_POOL DATA_POOL
  ```

示例

```
[ceph: root@host01 /]# ceph fs new test cephfs_metadata cephfs_data
```

c. 使用 **ceph orch apply** 命令部署 MDS 服务：

语法

```
ceph orch apply mds FILESYSTEM_NAME --placement="NUMBER_OF_DAEMONS
HOST_NAME_1 HOST_NAME_2 HOST_NAME_3"
```

示例

```
[ceph: root@host01 /]# ceph orch apply mds test --placement="2 host01 host02"
```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 检查 CephFS 状态：

  示例

  ```
  [ceph: root@host01 /]# ceph fs ls
  [ceph: root@host01 /]# ceph fs status
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

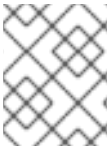
  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mds
  ```

其它资源

- 有关创建 Ceph 文件系统(CephFS)的更多信息，请参阅 Red Hat Ceph Storage File System Guide。
- 有关设置池值的详情，请参考 在池中设置放置组数量。

9.2. 使用服务规格部署 MDS 服务

通过使用 Ceph 编排器，您可以使用服务规格部署 MDS 服务。



注意

确保至少有两个池，一个用于 Ceph 文件系统(CephFS)数据，另一个用于 CephFS 元数据。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控和 OSD 守护进程。

流程

1. 创建 **mds.yaml** 文件：

   示例

   ```
   [root@host01 ~]# touch mds.yaml
   ```

2. 编辑 **mds.yaml** 文件，使其包含以下详情：

   语法

   ```
   service_type: mds
   service_id: FILESYSTEM_NAME
   placement:
     hosts:
     - HOST_NAME_1
     - HOST_NAME_2
     - HOST_NAME_3
   ```

   示例

   ```
   service_type: mds
   service_id: fs_name
   placement:
     hosts:
     - host01
     - host02
   ```

3. 将 YAML 文件挂载到容器中的一个目录下：

   示例

   ```
   [root@host01 ~]# cephadm shell --mount mds.yaml:/var/lib/ceph/mds/mds.yaml
   ```

4. 进入该目录：

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mds/
   ```

5. *登录到 Cephadm shell：*

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

6. *进入以下目录：*

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/mds/
   ```

7. *使用服务规格部署 MDS 服务：*

   语法

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   示例

   ```
   [ceph: root@host01 mds]# ceph orch apply -i mds.yaml
   ```

8. *部署 MDS 服务后，创建 CephFS：*

   语法

   ```
   ceph fs new CEPHFS_NAME METADATA_POOL DATA_POOL
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph fs new test metadata_pool data_pool
   ```

验证

- *列出服务：*

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- *列出主机、守护进程和进程：*

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=mds
  ```

其它资源

- 有关创建 Ceph 文件系统(CephFS)的更多信息，请参阅 Red Hat Ceph Storage File System Guide。

9.3. 使用 CEPH ORCHESTRATOR 删除 MDS 服务

您可以使用 **ceph orch rm** 命令删除该服务。或者，您也可以删除文件系统和相关池。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 在主机上至少部署一个 MDS 守护进程。

流程

- 可以通过两种方式从集群中移除 MDS 守护进程：

方法 1

- 移除 CephFS 卷、关联的池和服务：
  a. 登录到 Cephadm shell：

     示例

     ```
     [root@host01 ~]# cephadm shell
     ```

  b. 将配置参数 **mon_allow_pool_delete** 设置为 **true** ：

     示例

     ```
     [ceph: root@host01 /]# ceph config set mon mon_allow_pool_delete true
     ```

  c. 删除文件系统：

     语法

     ```
     ceph fs volume rm FILESYSTEM_NAME --yes-i-really-mean-it
     ```

     示例

     ```
     [ceph: root@host01 /]# ceph fs volume rm cephfs-new --yes-i-really-mean-it
     ```

     此命令将删除文件系统、其数据和元数据池。它还会尝试使用启用了 **ceph-mgr** Orchestrator 模块来删除 MDS。

方法 2

- *使用 **ceph orch rm** 命令从整个集群中删除 MDS 服务：*

  a. *列出服务：*

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

  b. *删除服务*

  语法

  ```
  ceph orch rm SERVICE_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch rm mds.test
  ```

验证

- *列出主机、守护进程和进程：*

  语法

  ```
  ceph orch ps
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps
  ```

其它资源

- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用命令行界面部分部署 MDS 服务。*
- *如需更多信息，请参阅 Red Hat Ceph Storage Operations Guide 中的使用服务规格部署 MDS 服务部分。*

第 10 章 使用 CEPH ORCHESTRATOR 管理 CEPH 对象网关

作为存储管理员，您可以使用命令行界面或使用服务规格部署 Ceph 对象网关。

您还可以配置多站点对象网关，并使用 Ceph 编排器移除 Ceph 对象网关。

Cephadm 将 Ceph 对象网关部署为守护进程的集合，可在多站点部署中管理单集群部署或特定 realm 和 zone。

注意

使用 Cephadm 时，对象网关守护进程使用 monitor 配置数据库而不是 **ceph.conf** 或命令行来进行配置。如果 **client.rgw** 部分中还没有该配置，则对象网关守护进程将使用默认设置启动并绑定到端口 **80**。

注意

只有 Ceph 对象网关中创建 bucket 之后，才会创建 **.default.rgw.buckets.index** 池，而数据上传到 bucket 后才会创建 **.default.rgw.buckets.data** 池。

本节涵盖了以下管理任务：

- 使用命令行界面部署 Ceph 对象网关。
- 使用服务规格部署 Ceph 对象网关。
- 使用 Ceph 编排器部署多站点 Ceph 对象网关。
- 使用 Ceph 编排器移除 Ceph 对象网关。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 所有管理器、监视器和 OSD 都部署在存储集群中。

10.1. 使用命令行界面部署 CEPH 对象网关

利用 Ceph 编排器，您可以在命令行界面中使用 **ceph orch** 命令部署 Ceph 对象网关。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 您可以通过三种不同的方式部署 Ceph 对象网关守护进程：

方法 1

- 创建 realm、zone group 和 zone，然后将放置规格与主机名搭配使用：

  a. 创建一个域：

  语法

  ```
  radosgw-admin realm create --rgw-realm=REALM_NAME --default
  ```

  示例

  ```
  [ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --default
  ```

  b. 创建区组：

  语法

  ```
  radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME  --master --default
  ```

  示例

  ```
  [ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=default  --master --default
  ```

  c. 创建区：

  语法

  ```
  radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME --rgw-zone=ZONE_NAME --master --default
  ```

  示例

  ```
  [ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=default --rgw-zone=test_zone --master --default
  ```

  d. 提交更改：

  语法

```
radosgw-admin period update --rgw-realm=REALM_NAME --commit
```

示例

```
[ceph: root@host01 /]# radosgw-admin period update --rgw-realm=test_realm --commit
```

e. 运行 **ceph orch apply** 命令：

语法

```
ceph orch apply rgw NAME [--realm=REALM_NAME] [--zone=ZONE_NAME] --
placement="NUMBER_OF_DAEMONS [HOST_NAME_1 HOST_NAME_2]"
```

示例

```
[ceph: root@host01 /]# ceph orch apply rgw test --realm=test_realm --zone=test_zone --
placement="2 host01 host02"
```

方法 2

- 使用任意服务名称为单个集群部署部署两个 Ceph 对象网关守护进程：

语法

```
ceph orch apply rgw SERVICE_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch apply rgw foo
```

方法 3

- 在标记的一组主机上使用任意服务名称：

语法

```
ceph orch host label add HOST_NAME_1 LABEL_NAME
ceph orch host label add HOSTNAME_2 LABEL_NAME
ceph orch apply rgw SERVICE_NAME --placement="label:LABEL_NAME count-per-
host:NUMBER_OF_DAEMONS" --port=8000
```

注意

NUMBER_OF_DAEMONS 控制每个主机上部署的 Ceph 对象网关数量。要在不增加成本的情况下获得最高的性能，请将此值设置为 2。

示例

```
[ceph: root@host01 /]# ceph orch host label add host01 rgw  # the 'rgw' label can be anything
[ceph: root@host01 /]# ceph orch host label add host02 rgw
[ceph: root@host01 /]# ceph orch apply rgw foo --placement="2 label:rgw" --port=8000
```

验证

- *列出服务：*

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- *列出主机、守护进程和进程：*

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
  ```

10.2. 使用服务规格部署 CEPH 对象网关

您可以使用服务规格和默认域、区域和区域组来部署 Ceph 对象网关。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *对 bootstrap 启动主机的 root 级别访问权限。*
- *主机添加到集群中。*
- *部署所有管理器、监控和 OSD 守护进程。*

流程

1. *作为 root 用户，创建规格文件：*

   示例

   ```
   [root@host01 ~]# touch radosgw.yml
   ```

2. *编辑 **radosgw.yml** 文件，使其包含 default realm、zone 和 zone group 的以下详情：*

   语法

   ```
   service_type: rgw
   service_id: REALM_NAME.ZONE_NAME
   placement:
     hosts:
   ```

```
  - HOST_NAME_1
  - HOST_NAME_2
  count_per_host: NUMBER_OF_DAEMONS
spec:
  rgw_realm: REALM_NAME
  rgw_zone: ZONE_NAME
  rgw_frontend_port: FRONT_END_PORT
networks:
  -  NETWORK_CIDR # Ceph Object Gateway service binds to a specific network
```

注意

NUMBER_OF_DAEMONS 控制每个主机上部署的 Ceph 对象网关数量。要在不增加成本的情况下获得最高的性能，请将此值设置为 2。

示例

```
service_type: rgw
service_id: default
placement:
  hosts:
  - host01
  - host02
  - host03
  count_per_host: 2
spec:
  rgw_realm: default
  rgw_zone: default
  rgw_frontend_port: 1234
networks:
  - 192.169.142.0/24
```

3. *可选：对于自定义 realm、zone 和 zone group，请创建资源，然后创建* ***radosgw.yml*** *文件：*

   a. *创建自定义 realm、zone 和 zone group：*

   示例

   ```
   [root@host01 ~]# radosgw-admin realm create --rgw-realm=test_realm
   [root@host01 ~]# radosgw-admin zonegroup create --rgw-zonegroup=test_zonegroup
   [root@host01 ~]# radosgw-admin zone create --rgw-zonegroup=test_zonegroup --rgw-zone=test_zone
   [root@host01 ~]# radosgw-admin period update --rgw-realm=test_realm --commit
   ```

   b. *使用以下详细信息，创建* ***radosgw.yml*** *文件：*

   示例

   ```
   service_type: rgw
   service_id: test_realm.test_zone
   placement:
     hosts:
     - host01
     - host02
   ```

```
  - host03
  count_per_host: 2
spec:
  rgw_realm: test_realm
  rgw_zone: test_zone
  rgw_frontend_port: 1234
networks:
  - 192.169.142.0/24
```

4. 将 **radosgw.yml** 文件挂载到容器中的某个目录下：

   示例

   ```
   [root@host01 ~]# cephadm shell --mount radosgw.yml:/var/lib/ceph/radosgw/radosgw.yml
   ```

   注意

   每次退出 shell 时，您都必须在部署守护进程前将该文件挂载到容器中。

5. 使用服务规格部署 Ceph 对象网关：

   语法

   ```
   ceph orch apply -i FILE_NAME.yml
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch apply -i radosgw.yml
   ```

验证

- 列出服务：

  示例

  ```
  [ceph: root@host01 /]# ceph orch ls
  ```

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=rgw
  ```

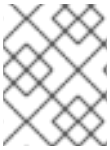
10.3. 使用 CEPH 编排器部署多站点 CEPH 对象网关

Ceph 编排器支持 Ceph 对象网关的多站点配置选项。

您可以将每个对象网关配置为在主动区域配置中工作，从而允许写入到非主要区域。多站点配置存储在名为 realm 的容器中。

*realm 存储 zone group、区域和一个时间周期。**rgw** 守护进程处理同步消除了对独立同步代理的需求，因此使用主动-主动配置运行。*

您还可以使用命令行界面(CLI)部署多站点区域。



注意

以下配置假定在地理上至少有两个 Red Hat Ceph Storage 集群。但是，配置也在同一站点工作。

先决条件

- *至少两个正在运行的 Red Hat Ceph Storage 集群。*
- *至少两个 Ceph 对象网关实例，每个实例对应一个 Red Hat Ceph Storage 集群。*
- *所有节点的根级别访问权限。*
- *节点或容器添加到存储集群中。*
- *部署所有 Ceph 管理器、监控器和 OSD 守护进程。*

流程

1. *在 **cephadm** shell 中，配置主区：*
   a. *创建一个域：*

   语法

   ```
   radosgw-admin realm create --rgw-realm=REALM_NAME --default
   ```

   示例

   ```
   [ceph: root@host01 /]# radosgw-admin realm create --rgw-realm=test_realm --default
   ```

   *如果存储集群只有一个域，则指定 **--default** 标志。*

   b. *创建主要区组：*

   语法

   ```
   radosgw-admin zonegroup create --rgw-zonegroup=ZONE_GROUP_NAME --
   endpoints=http://RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --
   master --default
   ```

   示例

   ```
   [ceph: root@host01 /]# radosgw-admin zonegroup create --rgw-zonegroup=us --
   endpoints=http://rgw1:80 --master --default
   ```

c. *创建一个主要区：*

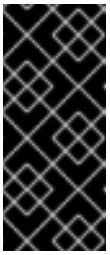
语法

```
radosgw-admin zone create --rgw-zonegroup=PRIMARY_ZONE_GROUP_NAME --rgw-zone=PRIMARY_ZONE_NAME --endpoints=http://RGW_PRIMARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY
```

示例

```
[ceph: root@host01 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-1 --endpoints=http://rgw1:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

d. *可选：删除默认 zone、zone group 和关联的池。*



重要

如果您使用默认 zone 和 zone group 存储数据，则不要删除默认区域及其池。另外，删除默认 zone group 会删除系统用户。

*要访问 **default** zone 和 zonegroup 中的旧数据，请在 **radosgw-admin** 命令中使用 **--rgw-zone default** 和 **--rgw-zonegroup default**。*

示例

```
[ceph: root@host01 /]# radosgw-admin zonegroup delete --rgw-zonegroup=default
[ceph: root@host01 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host01 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
```

e. *创建系统用户：*

语法

```
radosgw-admin user create --uid=USER_NAME --display-name="USER_NAME" --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY --system
```

示例

```
[ceph: root@host01 /]# radosgw-admin user create --uid=zone.user --display-name="Zone user" --system
```

*记录 **access_key** 和 **secret_key**。*

f. *在主区中添加 access key 和 system key：*

语法

```
radosgw-admin zone modify --rgw-zone=PRIMARY_ZONE_NAME --access-key=ACCESS_KEY --secret=SECRET_KEY
```

示例

```
[ceph: root@host01 /]# radosgw-admin zone modify --rgw-zone=us-east-1 --access-key=NE48APYCAODEPLKBCZVQ--secret=u24GHQWRE3yxxNBnFBzjM4jn14mFIckQ4EKL6LoW
```

g. *提交更改：*

语法

```
radosgw-admin period update --commit
```

示例

```
[ceph: root@host01 /]# radosgw-admin period update --commit
```

h. *在 **cephadm** shell 外部，获取存储集群的 **FSID** 及进程：*

示例

```
[root@host01 ~]#  systemctl list-units | grep ceph
```

i. *启动 Ceph 对象网关守护进程：*

语法

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

示例

```
[root@host01 ~]# systemctl start ceph-62a081a6-88aa-11eb-a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
[root@host01 ~]# systemctl enable ceph-62a081a6-88aa-11eb-a367-001a4a000672@rgw.test_realm.us-east-1.host01.ahdtsw.service
```

2. *在 Cephadm shell 中，配置 second zone。*

a. *从主机拉取主要域配置：*

语法

```
radosgw-admin realm pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-key=ACCESS_KEY --secret-key=SECRET_KEY
```

示例

```
[ceph: root@host04 /]# radosgw-admin realm pull --url=http://10.74.249.26:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

b. *从主机拉取主要 period 配置：*

语法

```
radosgw-admin period pull --url=URL_TO_PRIMARY_ZONE_GATEWAY --access-key=ACCESS_KEY --secret-key=SECRET_KEY
```

示例

```
[ceph: root@host04 /]# radosgw-admin period pull --url=http://10.74.249.26:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ
```

c. *配置 second zone:*

语法

```
radosgw-admin zone create --rgw-zonegroup=ZONE_GROUP_NAME \
        --rgw-zone=SECONDARY_ZONE_NAME --endpoints=http://RGW_SECONDARY_HOSTNAME:RGW_PRIMARY_PORT_NUMBER_1 \
        --access-key=SYSTEM_ACCESS_KEY --secret=SYSTEM_SECRET_KEY \
        --endpoints=http://FQDN:80 \
        [--read-only]
```

示例

```
[ceph: root@host04 /]# radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-2 --endpoints=http://rgw2:80 --access-key=LIPEYZJLTWXRKXS9LPJC --secret-key=IsAje0AVDNXNw48LjMAimpCpI7VaxJYSnfD0FFKQ --endpoints=http://rgw.example.com:80
```

d. *可选：删除默认区。*

重要

如果您使用默认 zone 和 zone group 存储数据，则不要删除默认区域及其池。

*要访问 **default** zone 和 zonegroup 中的旧数据，请在 **radosgw-admin** 命令中使用 **--rgw-zone default** 和 **--rgw-zonegroup default**。*

示例

```
[ceph: root@host04 /]# radosgw-admin zone rm --rgw-zone=default
[ceph: root@host04 /]# ceph osd pool rm default.rgw.log default.rgw.log --yes-i-really-really-mean-it
```

```
[ceph: root@host04 /]# ceph osd pool rm default.rgw.meta default.rgw.meta --yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.control default.rgw.control --yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.data.root default.rgw.data.root --yes-i-really-really-mean-it
[ceph: root@host04 /]# ceph osd pool rm default.rgw.gc default.rgw.gc --yes-i-really-really-mean-it
```

e. *更新 Ceph 配置数据库：*

语法

```
ceph config set SERVICE_NAME rgw_zone SECONDARY_ZONE_NAME
```

示例

```
[ceph: root@host04 /]# ceph config set rgw rgw_zone us-east-2
```

f. *提交更改：*

语法

```
radosgw-admin period update --commit
```

示例

```
[ceph: root@host04 /]# radosgw-admin period update --commit
```

g. *在 Cephadm shell 外，获取存储集群的 FSID 及进程：*

示例

```
[root@host04 ~]#  systemctl list-units | grep ceph
```

h. *启动 Ceph 对象网关守护进程：*

语法

```
systemctl start ceph-FSID@DAEMON_NAME
systemctl enable ceph-FSID@DAEMON_NAME
```

示例

```
[root@host04 ~]# systemctl start ceph-62a081a6-88aa-11eb-a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
[root@host04 ~]# systemctl enable ceph-62a081a6-88aa-11eb-a367-001a4a000672@rgw.test_realm.us-east-2.host04.ahdtsw.service
```

3. *可选：使用放置规格部署多站点 Ceph 对象网关：*

语法

```
ceph orch apply rgw NAME --realm=REALM_NAME --zone=PRIMARY_ZONE_NAME --
placement="NUMBER_OF_DAEMONS HOST_NAME_1 HOST_NAME_2"
```

示例

```
[ceph: root@host04 /]# ceph orch apply rgw east --realm=test_realm --zone=us-east-1 --
placement="2 host01 host02"
```

验证

- 检查同步状态以验证部署：

示例

```
[ceph: root@host04 /]# radosgw-admin sync status
```

10.4. 使用 CEPH 编排器移除 CEPH 对象网关

您可以使用 **ceph orch rm** 命令移除 Ceph 对象网关守护进程。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 主机上至少部署了一个 Ceph 对象网关守护进程。

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. 列出服务：

示例

```
[ceph: root@host01 /]# ceph orch ls
```

3. 删除服务：

语法

```
ceph orch rm SERVICE_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch rm rgw.test_realm.test_zone_bb
```

验证

- *列出主机、守护进程和进程：*

  语法

  ```
  ceph orch ps
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps
  ```

其它资源

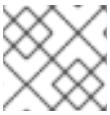
- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南中的使用命令行界面部分部署 Ceph 对象网关。*
- *如需更多信息，请参阅 Red Hat Ceph Storage 操作指南 中的使用服务规范部分部署 Ceph 对象网关。*

第 11 章 使用 CEPH ORCHESTRATOR(LIMITED AVAILABILITY) 管理 NFS-GANESHA 网关

作为存储管理员，您可以将 Orchestrator 与后端的 Cephadm 搭配使用，以部署 NFS-Ganesha 网关。Cephadm 利用预定义的 RADOS 池和可选命名空间部署 NFS Ganesha。

注意

该技术是有限可用性。如需更多信息，请参阅 已弃用的功能 章节。



注意

红帽支持仅对 NFS v4.0+ 协议进行 CephFS 导出。

本节涵盖了以下管理任务：

- 使用 Ceph Orchestrator 创建 NFS-Ganesha 集群。
- 使用命令行界面部署 NFS-Ganesha 网关。
- 使用服务规格部署 NFS-Ganesha 网关。
- 为 CephFS/NFS 服务实施 HA。
- 使用 Ceph Orchestrator 更新 NFS-Ganesha 集群。
- 使用 Ceph 编排器查看 NFS-Ganesha 集群信息。
- 使用 Ceph Orchestrator 获取 NFS-Ganesha 集群日志。
- 使用 Ceph 编排器设置自定义 NFS-Ganesha 配置。
- 使用 Ceph 编排器重置自定义 NFS-Ganesha 配置。
- 使用 Ceph Orchestrator 删除 NFS-Ganesha 集群。
- 使用 Ceph Orchestrator 移除 NFS Ganesha 网关。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。

11.1. 使用 CEPH ORCHESTRATOR 创建 NFS-GANESHA 集群

您可以使用 Ceph Orchestrator 的 **mgr/nfs** 模块来创建 NFS-Ganesha 集群。此模块使用后端中的 Cephadm 部署 NFS 集群。

*这会为所有 NFS-Ganesha 守护进程、基于 **clusterid** 的新用户和通用 NFS-Ganesha 配置 RADOS 对象创建一个通用恢复池。*

对于每个守护进程，池中都会创建一个新用户和一个通用配置。虽然所有集群都与集群名称相关的不同命名空间，但它们使用相同的恢复池。

先决条件

- *一个正在运行的 Red Hat Ceph Storage 集群。*
- *主机添加到集群中。*
- *部署所有管理器、监控器和 OSD 守护进程。*

流程

1. *登录到 Cephadm shell：*

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. *启用 **mgr/nfs** 模块：*

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module enable nfs
   ```

3. *创建集群：*

   语法

   ```
   ceph nfs cluster create CLUSTER_NAME ["HOST_NAME_1 HOST_NAME_2
   HOST_NAME_3"]
   ```

   CLUSTER_NAME 是一个任意字符串，HOST_NAME_1 是一个可选字符串，表示主机要部署 NFS-Ganesha 守护进程。

   示例

   ```
   [ceph: root@host01 /]# ceph nfs cluster create nfsganesha "host01 host02"
   NFS Cluster Created Successful
   ```

   *这会创建一个 NFS-Ganesha 集群 **nfsganesha**，并在 **host01** 和 **host02** 上有一个守护进程。*

验证

- *列出集群详情：*

  示例

  ```
  [ceph: root@host01 /]# ceph nfs cluster ls
  ```

- 显示 NFS-Ganesha 集群信息：

  语法

  ```
  ceph nfs cluster info CLUSTER_NAME
  ```

  示例

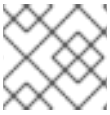
  ```
  [ceph: root@host01 /]# ceph nfs cluster info nfsganesha
  ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage 文件系统指南 中的通过 NFS 协议导出 Ceph 文件系统命名空间部分。
- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用服务规格显示 Ceph 守护进程。

11.2. 使用命令行界面部署 NFS-GANESHA 网关

您可以使用后端中的 Ceph Orchestrator 与 Cephadm 搭配使用，以根据放置规格部署 NFS-Ganesha 网关。在这种情况下，您必须创建 RADOS 池，并在部署网关前创建命名空间。



注意

红帽支持仅对 NFS v4.0+ 协议进行 CephFS 导出。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 创建 RADOS 池命名空间，再启用应用。对于 RBD 池，启用 RBD。

   语法

   ```
   ceph osd pool create POOL_NAME
   ceph osd pool application enable POOL_NAME freeform/rgw/rbd/cephfs/nfs
   rbd pool init -p POOL_NAME
   ```

   示例

```
[ceph: root@host01 /]# ceph osd pool create nfs-ganesha
[ceph: root@host01 /]# ceph osd pool application enable nfs-ganesha nfs
[ceph: root@host01 /]# rbd pool init -p nfs-ganesha
```

3. 在命令行界面中使用放置规格部署 NFS-Ganesha 网关：

语法

```
ceph orch apply nfs SERVICE_ID --placement="NUMBER_OF_DAEMONS HOST_NAME_1
HOST_NAME_2 HOST_NAME_3"
```

示例

```
[ceph: root@host01 /]# ceph orch apply nfs foo --placement="2 host01 host02"
```

这会部署一个 NFS-Ganesha 集群 **nfsganesha**，并在 **host01** 和 **host02** 上有一个守护进程。

验证

- 列出服务：

示例

```
[ceph: root@host01 /]# ceph orch ls
```

- 列出主机、守护进程和进程：

语法

```
ceph orch ps --daemon_type=DAEMON_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=nfs
```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用命令行界面显示 Ceph 守护进程。
- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage 块设备指南 中的创建块设备池部分。

11.3. 使用服务规格部署 NFS-GANESHA 网关

您可以使用后端中的 Ceph Orchestrator 与 Cephadm 搭配使用，以根据服务规格部署 NFS-Ganesha 网关。在这种情况下，您必须创建 RADOS 池，并在部署网关前创建命名空间。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- *主机添加到集群中。*

流程

1. *创建* ***nfs.yaml*** *文件：*

   示例

   ```
   [root@host01 ~]# touch nfs.yaml
   ```

2. *编辑 nfs.yaml 规格文件。*

   语法

   ```
   service_type: nfs
   service_id: SERVICE_ID
   placement:
     hosts:
       - HOST_NAME_1
       - HOST_NAME_2
   ```

   示例

   ```
   # cat nfs.yaml
   service_type: nfs
   service_id: foo
   placement:
     hosts:
       - host01
       - host02
   ```

   - *可选：通过在* ***ganesha.yaml*** *规格文件中添加* ***enable_nlm: true*** *来启用 NLM 支持 NFS 协议 v3 支持。*

     语法

     ```
     service_type: nfs
     service_id: SERVICE_ID
     placement:
       hosts:
         - HOST_NAME_1
         - HOST_NAME_2
     spec:
       enable_nlm: true
     ```

     示例

     ```
     # cat ganesha.yaml
     service_type: nfs
     service_id: foo
     placement:
       hosts:
         - host01
     ```

```
    - host02
spec:
    enable_nlm: true
```

3. 将 YAML 文件挂载到容器中的一个目录下：

示例

```
[root@host01 ~]# cephadm shell --mount nfs.yaml:/var/lib/ceph/nfs.yaml
```

4. 创建 RADOS 池、命名空间并启用 RBD：

语法

```
ceph osd pool create POOL_NAME
ceph osd pool application enable POOL_NAME rbd
rbd pool init -p POOL_NAME
```

示例

```
[ceph: root@host01 /]# ceph osd pool create nfs-ganesha
[ceph: root@host01 /]# ceph osd pool application enable nfs-ganesha rbd
[ceph: root@host01 /]# rbd pool init -p nfs-ganesha
```

5. 进入该目录：

示例

```
[ceph: root@host01 /]# cd /var/lib/ceph/
```

6. 使用服务规格部署 NFS-Ganesha 网关：

语法

```
ceph orch apply -i FILE_NAME.yaml
```

示例

```
[ceph: root@host01 ceph]# ceph orch apply -i nfs.yaml
```

验证

- 列出服务：

示例

```
[ceph: root@host01 /]# ceph orch ls
```

- 列出主机、守护进程和进程：

语法

-

```
ceph orch ps --daemon_type=DAEMON_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=nfs
```

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage 块设备指南 中的创建块设备池部分。

11.4. 为 CEPHFS/NFS 服务实施 HA （技术预览）

您可以使用 **--ingress** 标志并指定虚拟 IP 地址，使用高可用性(HA)前端、虚拟 IP 和负载均衡器部署 NFS。这会部署 **keepalived** 和 **haproxy** 的组合，并为 NFS 服务提供高可用性 NFS 前端。

当使用 **--ingress** 标志创建集群时，还会部署一个入口服务，以便为 NFS 服务器提供负载均衡和高可用性。虚拟 IP 用于提供所有 NFS 客户端可用于挂载的已知稳定 NFS 端点。Ceph 处理将虚拟 IP 上的 NFS 流量重定向到适当的后端 NFS 服务器的详细信息，并在 NFS 服务器失败时重新部署 NFS 服务器。

为现有服务部署 ingress 服务提供：

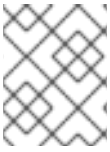
- 用于访问 NFS 服务器的稳定虚拟 IP。
- 跨多个 NFS 网关加载分布。
- 当主机出现故障时，主机之间的故障转移。

重要

CephFS/NFS 的 HA 只是一个技术预览功能。红帽产品服务级别协议（SLA）不支持技术预览功能，且其功能可能并不完善，因此红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。如需了解更多详细信息，请参阅红帽技术预览功能的支持范围。

重要

当在 NFS 集群前面部署 **入口** 服务时，后端 NFS-ganesha 服务器会看到 haproxy 的 IP 地址，而不是客户端的 IP 地址。因此，如果您根据 IP 地址限制客户端访问，则 NFS 导出的访问限制将无法正常工作。



注意

如果提供客户端的活动的 NFS 服务器停机，客户端的 I/O 会中断，直到替换活跃 NFS 服务器在线为止，NFS 集群再次处于活动状态。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控和 OSD 守护进程。

- 确保启用了 NFS 模块。

流程

1. 登录到 Cephadm shell：

   示例

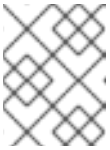
   ```
   [root@host01 ~]# cephadm shell
   ```

2. 使用 **--ingress** 标志创建 NFS 集群：

   语法

   ```
   ceph nfs cluster create CLUSTER_ID [PLACEMENT] [--port PORT_NUMBER] [--ingress --virtual-ip IP_ADDRESS/CIDR_PREFIX]
   ```

   - 将 CLUSTER_ID 替换为命名 NFS Ganesha 集群的唯一字符串。
   - 将 PLACEMENT 替换为要部署的 NFS 服务器数量，以及您要在其上部署 NFS Ganesha 守护进程容器的主机或主机。
   - 使用 **--port** PORT_NUMBER 标志，将 NFS 部署到默认端口 12049 以外的端口上。

     注意

     使用 ingress 模式时，高可用性代理使用端口 2049，NFS 服务会在 12049 上部署。

   - **--ingress** 标志与 **--virtual-ip** 标志相结合，使用高可用性前端（虚拟 IP 和负载均衡器）部署 NFS。
   - 将 **--virtual-ip** IP_ADDRESS 替换为 IP 地址，以提供已知稳定的 NFS 端点，所有客户端都可用于挂载 NFS 导出。**--virtual-ip** 必须包含 CIDR 前缀长度。虚拟 IP 通常将在第一个标识的网络接口上配置，在同一子网中具有现有 IP。

     注意

     为 NFS 服务分配的主机数量必须大于您请求要部署的活跃 NFS 服务器数量，由 **placement: count** 参数指定。在以下示例中，请求一个活跃的 NFS 服务器，并分配了两个主机。

     示例

     ```
     [ceph: root@host01 /]# ceph nfs cluster create mycephnfs "1 host02 host03" --ingress --virtual-ip 10.10.128.75/22
     ```

     注意

     NFS 守护进程和入口服务部署是异步的，命令可能会在服务完全启动前返回。

3. 检查服务是否已成功启动：

语法

```
ceph orch ls --service_name=nfs.CLUSTER_ID
ceph orch ls --service_name=ingress.nfs.CLUSTER_ID
```

示例

```
[ceph: root@host01 /]# ceph orch ls --service_name=nfs.mycephnfs

NAME          PORTS    RUNNING  REFRESHED  AGE  PLACEMENT
nfs.mycephnfs  ?:12049      2/2  0s ago     20s  host02;host03

[ceph: root@host01 /]# ceph orch ls --service_name=ingress.nfs.mycephnfs

NAME                 PORTS                 RUNNING  REFRESHED  AGE  PLACEMENT
ingress.nfs.mycephnfs  10.10.128.75:2049,9049      4/4   46s ago    73s  count:2
```

验证

- *查看 IP 端点、各个 NFS 守护进程的 IP 和* ***入口*** *服务的虚拟 IP：*

  语法

  ```
  ceph nfs cluster info CLUSTER_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph nfs cluster info mycephnfs

  {
      "mycephnfs": {
          "virtual_ip": "10.10.128.75",
          "backend": [
              {
                  "hostname": "host02",
                  "ip": "10.10.128.69",
                  "port": 12049
              },
              {
                  "hostname": "host03",
                  "ip": "10.10.128.70",
                  "port": 12049
              }
          ],
          "port": 2049,
          "monitor_port": 9049
      }
  }
  ```

- *列出主机和进程：*

  示例

```
[ceph: root@host01 /]# ceph orch ps | grep nfs

haproxy.nfs.cephnfs.host01.rftylv     host01  *:2049,9000  running (11m)    10m ago  11m
23.2M       -  2.2.19-7ea3822   5e6a41d77b38  f8cc61dc827e
haproxy.nfs.cephnfs.host02.zhtded     host02  *:2049,9000  running (11m)    53s ago  11m
21.3M       -  2.2.19-7ea3822   5e6a41d77b38  4cad324e0e23
keepalived.nfs.cephnfs.host01.zktmsk  host01               running (11m)    10m ago  11m
2349k       -  2.1.5           18fa163ab18f  66bf39784993
keepalived.nfs.cephnfs.host02.vyycvp  host02               running (11m)    53s ago  11m
2349k       -  2.1.5           18fa163ab18f  1ecc95a568b4
nfs.cephnfs.0.0.host02.fescmw         host02  *:12049      running (14m)     3m ago  14m
76.9M       -  3.5             cef6e7959b0a  bb0e4ee9484e
nfs.cephnfs.1.0.host03.avaddf         host03  *:12049      running (14m)     3m ago  14m
74.3M       -  3.5             cef6e7959b0a  ea02c0c50749
```

其他资源

- 有关在客户端主机上挂载 NFS 导出的详情，请参阅 Red Hat Ceph Storage 文件系统指南 中的通过 NFS 协议导出 Ceph 文件系统命名空间部分。

11.5. 为 HA 升级独立 CEPHFS/NFS 集群

作为存储管理员，您可以通过在现有 NFS 服务上部署 **ingress** 服务将独立存储集群升级到高可用性(HA)集群。

先决条件

- 正在运行的带有现有 NFS 服务的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控和 OSD 守护进程。
- 确保启用了 NFS 模块。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出现有的 NFS 集群：

   示例

   ```
   [ceph: root@host01 /]# ceph nfs cluster ls

   mynfs
   ```

注意

如果在一个节点上创建了独立 NFS 集群，则需要将其增加到两个或多个节点以实现 HA。要增加 NFS 服务，请编辑 **nfs.yaml** 文件，并使用同一端口号增加放置。

为 NFS 服务分配的主机数量必须大于您请求要部署的活跃 NFS 服务器数量，由 **placement: count** 参数指定。

语法

```
service_type: nfs
service_id: SERVICE_ID
placement:
  hosts:
    - HOST_NAME_1
    - HOST_NAME_2
  count: COUNT
spec:
  port: PORT_NUMBER
```

示例

```
service_type: nfs
service_id: mynfs
placement:
  hosts:
    - host02
    - host03
  count: 1
spec:
  port: 12345
```

在这个示例中，现有 NFS 服务在端口 **12345** 上运行，另一个节点则添加到具有相同端口的 NFS 集群中。

应用 **nfs.yaml** 服务规格更改来升级到两个节点 NFS 服务：

示例

```
[ceph: root@host01 ceph]# ceph orch apply -i nfs.yaml
```

3. 使用现有 NFS 集群 ID 编辑 **ingress.yaml** 规格文件：

语法

```
service_type: SERVICE_TYPE
service_id: SERVICE_ID
placement:
  count: PLACEMENT
spec:
  backend_service: SERVICE_ID_BACKEND (1)
  frontend_port: FRONTEND_PORT
  monitor_port: MONITOR_PORT (2)
  virtual_ip: VIRTUAL_IP_WITH_CIDR
```

-

(1) service_ID_BACKEND 应该包含一个 PORT 属性，它不是 **2049**，以避免与 ingress 服务冲突，该服务可以放在同一主机上。

(2) MONITOR_PORT 用于访问 haproxy 负载状态页面。

示例

```
service_type: ingress
service_id: nfs.mynfs
placement:
  count: 2
spec:
  backend_service: nfs.mynfs
  frontend_port: 2049
  monitor_port: 9000
  virtual_ip: 10.10.128.75/22
```

4. 部署 ingress 服务：

示例

```
[ceph: root@host01 /]# ceph orch apply -i ingress.yaml
```

注意

NFS 守护进程和入口服务部署是异步的，命令可能会在服务完全启动前返回。

5. 检查入口服务是否已成功启动：

语法

```
ceph orch ls --service_name=ingress.nfs.CLUSTER_ID
```

示例

```
[ceph: root@host01 /]# ceph orch ls --service_name=ingress.nfs.mynfs

NAME                 PORTS                 RUNNING  REFRESHED  AGE  PLACEMENT
ingress.nfs.mynfs  10.10.128.75:2049,9000      4/4   4m ago     22m  count:2
```

验证

- 查看 IP 端点、各个 NFS 守护进程的 IP 和 **入口** 服务的虚拟 IP：

语法

```
ceph nfs cluster info CLUSTER_NAME
```

示例

```
[ceph: root@host01 /]# ceph nfs cluster info mynfs

{
    "mynfs": {
        "virtual_ip": "10.10.128.75",
        "backend": [
            {
                "hostname": "host02",
                "ip": "10.10.128.69",
                "port": 12049
            },
            {
                "hostname": "host03",
                "ip": "10.10.128.70",
                "port": 12049
            }
        ],
        "port": 2049,
        "monitor_port": 9049
    }
}
```

- 列出主机和进程：

  示例

```
[ceph: root@host01 /]# ceph orch ps | grep nfs

haproxy.nfs.mynfs.host01.ruyyhq     host01  *:2049,9000  running (27m)     6m ago  34m
9.85M        -  2.2.19-7ea3822   5e6a41d77b38  328d27b3f706
haproxy.nfs.mynfs.host02.ctrhha     host02  *:2049,9000  running (34m)     6m ago  34m
4944k        -  2.2.19-7ea3822   5e6a41d77b38  4f4440dbfde9
keepalived.nfs.mynfs.host01.fqgjxd  host01               running (27m)     6m ago  34m    31.2M
-  2.1.5            18fa163ab18f  0e22b2b101df
keepalived.nfs.mynfs.host02.fqzkxb  host02               running (34m)     6m ago  34m    17.5M
-  2.1.5            18fa163ab18f  c1e3cc074cf8
nfs.mynfs.0.0.host02.emoaut         host02  *:12345      running (37m)     6m ago  37m    82.7M
-  3.5              91322de4f795  2d00faaa2ae5
nfs.mynfs.1.0.host03.nsxcfd         host03  *:12345      running (37m)     6m ago  37m    81.1M
-  3.5              91322de4f795  d4bda4074f17
```

其他资源

- 有关在客户端主机上挂载 NFS 导出的详情，请参阅 Red Hat Ceph Storage 文件系统指南 中的通过 NFS 协议导出 Ceph 文件系统命名空间部分。

11.6. 使用规格文件为 CEPHFS/NFS 部署 HA

您可以首先部署 NFS 服务，然后部署到相同 NFS 服务的 **ingress** 来使用规格文件部署 HA。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。

- *主机添加到集群中。*
- *部署所有管理器、监控和 OSD 守护进程。*
- *确保启用了 NFS 模块。*

流程

1. *登录到 Cephadm shell：*

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. *确定启用了 NFS 模块：*

   示例

   ```
   [ceph: root@host01 /]# ceph mgr module ls | more
   ```

3. *退出 Cephadm shell 并创建 **nfs.yaml** 文件：*

   示例

   ```
   [root@host01 ~]# touch nfs.yaml
   ```

4. *编辑 **nfs.yaml** 文件，使其包含以下详情：*

   语法

   ```
   service_type: nfs
   service_id: SERVICE_ID
   placement:
     hosts:
       - HOST_NAME_1
       - HOST_NAME_2
     count: COUNT
   spec:
     port: PORT_NUMBER
   ```

   注意

   *为 NFS 服务分配的主机数量必须大于您请求要部署的活跃 NFS 服务器数量，由 **placement: count** 参数指定。*

   示例

   ```
   service_type: nfs
   service_id: cephfsnfs
   placement:
     hosts:
       - host02
       - host03
   ```

```
  count: 1
spec:
  port: 12345
```

在本例中，服务器在非默认端口 ***12345*** *上运行，而不是在 host02 和 host03 的默认端口* ***2049*** *上运行。*

5. *将 YAML 文件挂载到容器中的一个目录下：*

   示例

   ```
   [root@host01 ~]# cephadm shell --mount nfs.yaml:/var/lib/ceph/nfs.yaml
   ```

6. *登录到 Cephadm shell 并导航到目录：*

   示例

   ```
   [ceph: root@host01 /]# cd /var/lib/ceph/
   ```

7. *使用服务规格部署 NFS 服务：*

   语法

   ```
   ceph orch apply -i FILE_NAME.yaml
   ```

   示例

   ```
   [ceph: root@host01 ceph]# ceph orch apply -i nfs.yaml
   ```

   注意

   NFS 服务的部署是异步的，命令可能会在服务完全启动前返回。

8. *检查 NFS 服务是否已成功启动：*

   语法

   ```
   ceph orch ls --service_name=nfs.CLUSTER_ID
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch ls --service_name=nfs.cephfsnfs

   NAME          PORTS    RUNNING  REFRESHED  AGE  PLACEMENT
   nfs.cephfsnfs  ?:12345      2/2  3m ago     13m  host02;host03
   ```

9. *退出 Cephadm shell 并创建* ***ingress.yaml*** *文件：*

   示例

   ```
   [root@host01 ~]# touch ingress.yaml
   ```

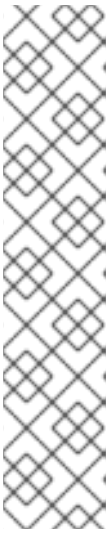
10. 编辑 **ingress.yaml** 文件，使其包含以下详情：

语法

```
service_type: SERVICE_TYPE
service_id: SERVICE_ID
placement:
  count: PLACEMENT
spec:
  backend_service: SERVICE_ID_BACKEND
  frontend_port: FRONTEND_PORT
  monitor_port: MONITOR_PORT
  virtual_ip: VIRTUAL_IP_WITH_CIDR
```

示例

```
service_type: ingress
service_id: nfs.cephfsnfs
placement:
  count: 2
spec:
  backend_service: nfs.cephfsnfs
  frontend_port: 2049
  monitor_port: 9000
  virtual_ip: 10.10.128.75/22
```



注意

在本例中，**placement: count: 2** 在随机节点上部署 **keepalived** 和 **haproxy** 服务。要指定要在其上部署 **keepalived** 和 **haproxy** 的节点，请使用 **placement: hosts** 选项：

示例

```
service_type: ingress
service_id: nfs.cephfsnfs
placement:
  hosts:
    - host02
    - host03
```

11. 将 YAML 文件挂载到容器中的一个目录下：

示例

```
[root@host01 ~]# cephadm shell --mount ingress.yaml:/var/lib/ceph/ingress.yaml
```

12. 登录到 Cephadm shell 并导航到目录：

示例

```
[ceph: root@host01 /]# cd /var/lib/ceph/
```

13. *使用服务规格部署* ***ingress*** *服务：*

语法

```
ceph orch apply -i FILE_NAME.yaml
```

示例

```
[ceph: root@host01 ceph]# ceph orch apply -i ingress.yaml
```

14. *检查入口服务是否已成功启动：*

语法

```
ceph orch ls --service_name=ingress.nfs.CLUSTER_ID
```

示例

```
[ceph: root@host01 /]# ceph orch ls --service_name=ingress.nfs.cephfsnfs

NAME                  PORTS                  RUNNING  REFRESHED  AGE  PLACEMENT
ingress.nfs.cephfsnfs  10.10.128.75:2049,9000      4/4   4m ago     22m  count:2
```

验证

- *查看 IP 端点、各个 NFS 守护进程的 IP 和* ***入口*** *服务的虚拟 IP：*

语法

```
ceph nfs cluster info CLUSTER_NAME
```

示例

```
[ceph: root@host01 /]# ceph nfs cluster info cephfsnfs

{
   "cephfsnfs": {
      "virtual_ip": "10.10.128.75",
      "backend": [
         {
            "hostname": "host02",
            "ip": "10.10.128.69",
            "port": 12345
         },
         {
            "hostname": "host03",
            "ip": "10.10.128.70",
            "port": 12345
         }
      ],
      "port": 2049,
```

```
        "monitor_port": 9049
    }
}
```

- 列出主机和进程：

  示例

```
[ceph: root@host01 /]# ceph orch ps | grep nfs

haproxy.nfs.cephfsnfs.host01.ruyyhq  host01  *:2049,9000  running (27m)  6m ago  34m
9.85M  -  2.2.19-7ea3822  5e6a41d77b38  328d27b3f706
haproxy.nfs.cephfsnfs.host02.ctrhha  host02  *:2049,9000  running (34m)  6m ago  34m
4944k  -  2.2.19-7ea3822  5e6a41d77b38  4f4440dbfde9
keepalived.nfs.cephfsnfs.host01.fqgjxd  host01  running (27m)  6m ago  34m
31.2M  -  2.1.5  18fa163ab18f  0e22b2b101df
keepalived.nfs.cephfsnfs.host02.fqzkxb  host02  running (34m)  6m ago  34m
17.5M  -  2.1.5  18fa163ab18f  c1e3cc074cf8
nfs.cephfsnfs.0.0.host02.emoaut  host02  *:12345  running (37m)  6m ago  37m
82.7M  -  3.5  91322de4f795  2d00faaa2ae5
nfs.cephfsnfs.1.0.host03.nsxcfd  host03  *:12345  running (37m)  6m ago  37m
81.1M  -  3.5  91322de4f795  d4bda4074f17
```

其他资源

- 有关在客户端主机上挂载 NFS 导出的详情，请参阅 Red Hat Ceph Storage 文件系统指南 中的通过 NFS 协议导出 Ceph 文件系统命名空间部分。

11.7. 使用 CEPH ORCHESTRATOR 更新 NFS-GANESHA 集群

您可以通过在后端中使用 Ceph Orchestrator 与 Cephadm 进行 Cephadm 更改主机上的守护进程放置来更新 NFS-Ganesha 集群。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。
- 使用 **mgr/nfs** 模块创建的 NFS-Ganesha 集群。

流程

1. 登录到 Cephadm shell：

   示例

```
[root@host01 ~]# cephadm shell
```

2. 更新集群：

   语法

```
ceph orch apply nfs CLUSTER_NAME ["HOST_NAME_1,HOST_NAME_2,HOST_NAME_3"]
```

CLUSTER_NAME 是一个任意字符串 HOST_NAME_1，它是一个可选的字符串，代表主机更新部署的 NFS-Ganesha 守护进程。

示例

```
[ceph: root@host01 /]# ceph orch apply nfs nfsganesha "host02"
```

*这将更新 **host02** 上的 **nfsganesha** 集群。*

验证

- *列出集群详情：*

  示例

  ```
  [ceph: root@host01 /]# ceph nfs cluster ls
  ```

- *显示 NFS-Ganesha 集群信息：*

  语法

  ```
  ceph nfs cluster info CLUSTER_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph nfs cluster info nfsganesha
  ```

- *列出主机、守护进程和进程：*

  语法

  ```
  ceph orch ps --daemon_type=DAEMON_NAME
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps --daemon_type=nfs
  ```

其它资源

- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 创建 NFS-Ganesha 集群 部分。*

11.8. 使用 CEPH ORCHESTRATOR 查看 NFS-GANESHA 集群信息

您可以使用 Ceph Orchestrator 查看 NFS-Ganesha 集群的信息。您可以获取有关所有 NFS Ganesha 集群或特定集群及其端口、IP 地址以及创建集群的主机名称的信息。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。
- 使用 **mgr/nfs** 模块创建的 NFS-Ganesha 集群。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 查看 NFS-Ganesha 集群信息：

   语法

   ```
   ceph nfs cluster info CLUSTER_NAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph nfs cluster info nfsganesha

   {
       "nfsganesha": [
           {
               "hostname": "host02",
               "ip": [
                   "10.10.128.70"
               ],
               "port": 2049
           }
       ]
   }
   ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 创建 NFS-Ganesha 集群 部分。

11.9. 使用 CEPH ORCHESTRATOR 获取 NFS-GANESHA 排除器日志

使用 Ceph 编排器，您可以获取 NFS-Ganesha 集群日志。您需要处于部署该服务的节点。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 部署 NFS 的节点上安装 Cephadm。

- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 使用 **mgr/nfs** 模块创建的 NFS-Ganesha 集群。

流程

1. 作为 root 用户，获取存储集群的 FSID ：

   示例

   ```
   [root@host03 ~]# cephadm ls
   ```

   复制 FSID 和服务的名称。

2. 获取日志：

   语法

   ```
   cephadm logs --fsid FSID --name SERVICE_NAME
   ```

   示例

   ```
   [root@host03 ~]# cephadm logs --fsid 499829b4-832f-11eb-8d6d-001a4a000635 --name nfs.foo.host03
   ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用放置规格显示 Ceph 守护进程。

11.10. 使用 CEPH ORCHESTRATOR 设置自定义 NFS-GANESHA 配置

NFS-Ganesha 集群在默认配置块中定义。通过使用 Ceph 编排器，您可以自定义配置，其优先级高于默认配置块。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。
- 使用 **mgr/nfs** 模块创建的 NFS-Ganesha 集群。

流程

1. 登录到 Cephadm shell：

   示例

```
[root@host01 ~]# cephadm shell
```

2. 以下是 NFS-Ganesha 集群默认配置的示例：

示例

```
# {{ cephadm_managed }}
NFS_CORE_PARAM {
        Enable_NLM = false;
        Enable_RQUOTA = false;
        Protocols = 4;
}

MDCACHE {
        Dir_Chunk = 0;
}

EXPORT_DEFAULTS {
        Attr_Expiration_Time = 0;
}

NFSv4 {
        Delegations = false;
        RecoveryBackend = 'rados_cluster';
        Minor_Versions = 1, 2;
}

RADOS_KV {
        UserId = "{{ user }}";
        nodeid = "{{ nodeid}}";
        pool = "{{ pool }}";
        namespace = "{{ namespace }}";
}

RADOS_URLS {
        UserId = "{{ user }}";
        watch_url = "{{ url }}";
}

RGW {
        cluster = "ceph";
        name = "client.{{ rgw_user }}";
}

%url    {{ url }}
```

3. 自定义 NFS-Ganesha 集群配置。以下是自定义配置的两个示例：

- 更改日志级别：

示例

```
LOG {
 COMPONENTS {
    ALL = FULL_DEBUG;
```

```
}
}
```

- *添加自定义导出块：*

  a. *创建用户。*

  注意

  在 FSAL 块中指定的用户应当具有适当的上限，以便 NFS-Ganesha 守护进程能访问 Ceph 集群。

  语法

  ```
  ceph auth get-or-create client.USER_ID mon 'allow r' osd 'allow rw pool=.nfs namespace=NFS_CLUSTER_NAME, allow rw tag cephfs data=FS_NAME' mds 'allow rw path=EXPORT_PATH'
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph auth get-or-create client.f64f341c-655d-11eb-8778-fa163e914bcc mon 'allow r' osd 'allow rw pool=.nfs namespace=nfs_cluster_name, allow rw tag cephfs data=fs_name' mds 'allow rw path=export_path'
  ```

  b. *进入以下目录：*

  语法

  ```
  cd /var/lib/ceph/DAEMON_PATH/
  ```

  示例

  ```
  [ceph: root@host01 /]# cd /var/lib/ceph/nfs/
  ```

  *如果 **nfs** 目录不存在，请在路径中创建一个目录。*

  c. *创建新配置文件：*

  语法

  ```
  touch PATH_TO_CONFIG_FILE
  ```

  示例

  ```
  [ceph: root@host01 nfs]#  touch nfs-ganesha.conf
  ```

  d. *通过添加自定义导出块来编辑配置文件。它创建一个导出，并且由 Ceph NFS 导出接口管理。*

  语法

```
EXPORT {
  Export_Id = NUMERICAL_ID;
  Transports = TCP;
  Path = PATH_WITHIN_CEPHFS;
  Pseudo = BINDING;
  Protocols = 4;
  Access_Type = PERMISSIONS;
  Attr_Expiration_Time = 0;
  Squash = None;
  FSAL {
    Name = CEPH;
    Filesystem = "FILE_SYSTEM_NAME";
    User_Id = "USER_NAME";
    Secret_Access_Key = "USER_SECRET_KEY";
  }
}
```

示例

```
EXPORT {
  Export_Id = 100;
  Transports = TCP;
  Path = /;
  Pseudo = /ceph/;
  Protocols = 4;
  Access_Type = RW;
  Attr_Expiration_Time = 0;
  Squash = None;
  FSAL {
    Name = CEPH;
    Filesystem = "filesystem name";
    User_Id = "user id";
    Secret_Access_Key = "secret key";
  }
}
```

4. 对集群应用新配置：

语法

```
ceph nfs cluster config set CLUSTER_NAME -i PATH_TO_CONFIG_FILE
```

示例

```
[ceph: root@host01 nfs]# ceph nfs cluster config set nfs-ganesha -i /var/lib/ceph/nfs/nfs-ganesha.conf
```

这还会为自定义配置重启服务。

验证

- 列出服务：

示例

```
[ceph: root@host01 /]# ceph orch ls
```

- 列出主机、守护进程和进程：

语法

```
ceph orch ps --daemon_type=DAEMON_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=nfs
```

- 验证自定义配置：

语法

```
/bin/rados -p POOL_NAME -N CLUSTER_NAME get userconf-nfs.CLUSTER_NAME -
```

示例

```
[ceph: root@host01 /]# /bin/rados -p nfs-ganesha -N nfsganesha get userconf-
nfs.nfsganesha -
```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 设置自定义 NFS-Ganesha 配置部分。

11.11. 使用 CEPH ORCHESTRATOR 重置自定义 NFS-GANESHA 配置

使用 Ceph 编排器，您可以将用户定义的配置重置为默认配置。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。
- 使用 **mgr/nfs** 模块部署的 NFS-Ganesha。
- 已设置自定义 NFS 集群配置

流程

1. 登录到 Cephadm shell：

示例

```
[root@host01 ~]# cephadm shell
```

2. *重置 NFS-Ganesha 配置：*

语法

```
ceph nfs cluster config reset CLUSTER_NAME
```

示例

```
[ceph: root@host01 /]# ceph nfs cluster config reset nfs-cephfs
```

验证

- *列出服务：*

示例

```
[ceph: root@host01 /]# ceph orch ls
```

- *列出主机、守护进程和进程：*

语法

```
ceph orch ps --daemon_type=DAEMON_NAME
```

示例

```
[ceph: root@host01 /]# ceph orch ps --daemon_type=nfs
```

- *验证自定义配置已被删除：*

语法

```
/bin/rados -p POOL_NAME -N CLUSTER_NAME get userconf-nfs.CLUSTER_NAME -
```

示例

```
[ceph: root@host01 /]# /bin/rados -p nfs-ganesha -N nfsganesha get userconf-
nfs.nfsganesha -
```

其它资源

- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 创建 NFS-Ganesha 集群 部分。*
- *如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 设置自定义 NFS-Ganesha 配置部分。*

11.12. 使用 CEPH ORCHESTRATOR 删除 NFS-GANESHA 集群

您可以在后端中将 Ceph 编排器与 Cephadm 一起删除 NFS-Ganesha 集群。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 主机添加到集群中。
- 部署所有管理器、监控器和 OSD 守护进程。
- 使用 **mgr/nfs** 模块创建的 NFS-Ganesha 集群。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 删除集群：

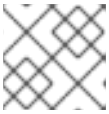
   语法

   ```
   ceph nfs cluster rm CLUSTER_NAME
   ```

   CLUSTER_NAME 是一个任意字符串。

   示例

   ```
   [ceph: root@host01 /]# ceph nfs cluster rm nfsganesha
   NFS Cluster Deleted Successfully
   ```

   注意

   delete 选项已弃用，您需要使用 **rm** 删除 NFS 集群。

验证

- 列出集群详情：

  示例

  ```
  [ceph: root@host01 /]# ceph nfs cluster ls
  ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage 文件系统指南 中的通过 NFS 协议导出 Ceph 文件系统命名空间部分。
- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用 Ceph Orchestrator 创建 NFS-Ganesha 集群 部分。

11.13. 使用 CEPH ORCHESTRATOR 删除 NFS-GANESHA 网关

您可以使用 **ceph orch rm** 命令删除 NFS-Ganesha 网关。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 所有节点的根级别访问权限。
- 主机添加到集群中。
- 主机上至少部署了一个 NFS-Ganesha 网关。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 列出服务：

   示例

   ```
   [ceph: root@host01 /]# ceph orch ls
   ```

3. 删除服务：

   语法

   ```
   ceph orch rm SERVICE_NAME
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch rm nfs.foo
   ```

验证

- 列出主机、守护进程和进程：

  语法

  ```
  ceph orch ps
  ```

  示例

  ```
  [ceph: root@host01 /]# ceph orch ps
  ```

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用服务规格显示 Ceph 守护进程。
- 如需更多信息，请参阅 Red Hat Ceph Storage Operations 指南 中的使用服务规格部署 NFS-Ganesha 网关 部分。

11.14. KERBEROS 集成

Kerberos 是一种计算机网络安全协议，提供集中身份验证服务器，向服务器验证用户，反之亦然。在 Kerberos 身份验证中，服务器和数据库用于客户端身份验证。

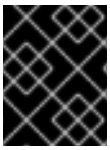
11.14.1. 设置 KDC （根据要求）

Kerberos 作为第三方可信服务器运行，称为密钥分发中心(KDC)，其中每个用户和服务都是主体。KDC 包含有关所有客户端（用户主体、服务主体）的信息，因此需要安全。在 Kerberos 设置中，因为 KDC 是单一故障点，建议有一个主 KDC 和多个从 KDC。

先决条件

验证 **/etc/hosts** 文件中是否进行了以下更改。如果需要，添加域名。

```
[root@chost ~]# cat /etc/hosts

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.208.97         ceph-node1-installer.ibm.com ceph-node1-installer
10.0.210.243 ceph-node2.ibm.com ceph-node2
10.0.208.63         ceph-node3.ibm.com ceph-node3
10.0.210.222 ceph-node4.ibm.com ceph-node4
10.0.210.235 ceph-node5.ibm.com ceph-node5
10.0.209.87         ceph-node6.ibm.com ceph-node6
10.0.208.89         ceph-node7.ibm.com ceph-node7
```



重要

确保设置中的所有涉及的节点都存在域名(Ceph 集群中的所有节点和所有 NFS 客户端节点)。

流程

按照以下步骤安装和配置 KDC。如果您已安装和配置了 KDC，请跳过此部分。

1. 检查您要设置 KDC 的机器上安装了所需的 RPM。

```
[root@host ~]# rpm -qa | grep krb5

krb5-libs-1.20.1-9.el9_2.x86_64
krb5-pkinit-1.20.1-9.el9_2.x86_64
krb5-server-1.20.1-9.el9_2.x86_64
krb5-server-ldap-1.20.1-9.el9_2.x86_64
krb5-devel-1.20.1-9.el9_2.x86_64
krb5-workstation-1.20.1-9.el9_2.x86_64
```

注意

- *最好根据 Kerberos REALM 名称具有域名。例如，Realm - PUNE.IBM.COM，管理员主体 - admin/admin。*
- *编辑安装的配置文件，以反映新的 KDC。请注意，KDC 可以作为 IP 地址或 DNS 名称提供。*

2. *更新 **krb5.conf** 文件：*

注意

*您需要使用 **krb5.conf** 文件中的 **kdc** 和 **admin_server** IP 更新所有域 (**default_realm** 和 **domain_realm**)。*

```
[root@host ~]# cat /etc/krb5.conf

# To opt out of the system crypto-policies configuration of krb5, remove the
# symlink at /etc/krb5.conf.d/crypto-policies which will not be recreated.

includedir /etc/krb5.conf.d/
[logging]
    default = [FILE:/var/log/krb5libs.log](file:///var/log/krb5libs.log)
    kdc = [FILE:/var/log/krb5kdc.log](file:///var/log/krb5kdc.log)
    admin_server = [FILE:/var/log/kadmind.log](file:///var/log/kadmind.log)
[libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = [FILE:/etc/pki/tls/certs/ca-bundle.crt](file:///etc/pki/tls/certs/ca-bundle.crt)
    spake_preauth_groups = edwards25519
    dns_canonicalize_hostname = fallback
    qualify_shortname = ""
    default_realm = PUNE.IBM.COM
    default_ccache_name = KEYRING:persistent:%{uid}
[realms]
    PUNE.IBM.COM = {
      kdc = 10.0.210.222:88
      admin_server = 10.0.210.222:749
    }
[domain_realm]
  .redhat.com = PUNE.IBM.COM
  redhat.com = PUNE.IBM.COM
```

3. *更新 **krb5.conf** 文件：*

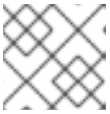
注意

*您需要更新 **kdc.conf** 文件中的域。*

```
[root@host ~]# cat /var/kerberos/krb5kdc/kdc.conf
```

```
 [kdcdefaults]
 kdc_ports = 88
 kdc_tcp_ports = 88
 spake_preauth_kdc_challenge = edwards25519
[realms]
   PUNE.IBM.COM = {
      master_key_type = aes256-cts-hmac-sha384-192
      acl_file = /var/kerberos/krb5kdc/kadm5.acl
      dict_file = /usr/share/dict/words
      default_principal_flags = +preauth
      admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
      supported_enctypes = aes256-cts-hmac-sha384-192:normal aes128-cts-hmac-sha256-
128:normal aes256-cts-hmac-sha1-96:normal aes128-cts-hmac-sha1-96:normal
camellia256-cts-cmac:normal camellia128-cts-cmac:normal arcfour-hmac-md5:normal
     # Supported encryption types for FIPS mode:
     #supported_enctypes = aes256-cts-hmac-sha384-192:normal aes128-cts-hmac-sha256-
128:normal
  }
```

4. 使用 **kdb5_util** 创建 KDC 数据库：



注意

确保主机名可以通过 **DNS** 或 **/etc/hosts** 解析。

```
[root@host ~]# kdb5_util create -s -r [PUNE.IBM.COM](http://pune.ibm.com/)

Initializing database '/var/kerberos/krb5kdc/principal' for realm 'PUNE.IBM.COM',
master key name 'K/M@PUNE.IBM.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

5. 将管理员添加到 ACL 文件中：

```
[root@host ~]# cat /var/kerberos/krb5kdc/kadm5.acl

*/admin@PUNE.IBM.COM *
```

输出表明，带有 admin 实例的 PUNE.IBM.COM 域中的任何主体都具有所有管理特权。

6. 将管理员添加到 Kerberos 数据库中：

```
[root@host ~]# kadmin.local

Authenticating as principal root/admin@PUNE.IBM.COM with password.
kadmin.local:  addprinc admin/admin@PUNE.IBM.COM
No policy specified for admin/admin@PUNE.IBM.COM; defaulting to no policy
Enter password for principal "admin/admin@PUNE.IBM.COM":
Re-enter password for principal "admin/admin@PUNE.IBM.COM":
Principal "admin/admin@PUNE.IBM.COM" created.
kadmin.local:
```

7. 启动 **kdc** 和 **kadmind**:

```
# krb5kdc
# kadmind
```

验证

- 检查 **kdc** 和 **kadmind** 是否在正确运行：

```
# ps -eaf | grep krb

root     27836     1  0 07:35 ?        00:00:00 krb5kdc
root     27846 13956  0 07:35 pts/8    00:00:00 grep --color=auto krb
# ps -eaf | grep kad
root     27841     1  0 07:35 ?        00:00:00 kadmind
root     27851 13956  0 07:36 pts/8    00:00:00 grep --color=auto kad
```

- 检查设置是否正确：

```
[root@host ~]# kinit admin/admin
Password for admin/admin@PUNE.IBM.COM:

[root@ceph-mani-o7fdxp-node4 ~]# klist
Ticket cache: KCM:0
Default principal: admin/admin@PUNE.IBM.COM

Valid starting     Expires            Service principal
10/25/23 06:37:08  10/26/23 06:37:01  krbtgt/PUNE.IBM.COM@PUNE.IBM.COM
    renew until 10/25/23 06:37:08
```

11.14.2. 设置 Kerberos 客户端

Kerberos 客户端计算机应该与 KDC 同步。确保使用 NTP 同步 KDC 和客户端。五分钟或更长时间差会导致 Kerberos 身份验证失败，并抛出时钟偏移错误。此步骤是所有要参与 NFS 客户端等 Kerberos 身份验证的系统(NFS Ganesha 容器将运行 NFS Ganesha 容器的主机)的先决条件。

流程

1. 检查所需的 RPM

```
[root@host ~]# rpm -qa | grep krb5

krb5-libs-1.20.1-9.el9_2.x86_64
krb5-pkinit-1.20.1-9.el9_2.x86_64
krb5-workstation-1.20.1-9.el9_2.x86_64
```

2. 更新 **krb5.conf** 文件，类似于 KDC 服务器上的文件：

```
[root@host ~]# cat /etc/krb5.conf

# To opt out of the system crypto-policies configuration of krb5, remove the
# symlink at /etc/krb5.conf.d/crypto-policies which will not be recreated.
```

```
includedir /etc/krb5.conf.d/
[logging]
    default = [FILE:/var/log/krb5libs.log](file:///var/log/krb5libs.log)
    kdc = [FILE:/var/log/krb5kdc.log](file:///var/log/krb5kdc.log)
    admin_server = [FILE:/var/log/kadmind.log](file:///var/log/kadmind.log)
[libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = [FILE:/etc/pki/tls/certs/ca-bundle.crt](file:///etc/pki/tls/certs/ca-bundle.crt)
    spake_preauth_groups = edwards25519
    dns_canonicalize_hostname = fallback
    qualify_shortname = ""
    default_realm = PUNE.IBM.COM
   default_ccache_name = KEYRING:persistent:%{uid}
[realms]
  PUNE.IBM.COM = {
     kdc = 10.0.210.222:88
     admin_server = 10.0.210.222:749
  }
[domain_realm]
  .IBM.com = PUNE.IBM.COM
  IBM.com = PUNE.IBM.COM
```

验证

- *验证客户端设置：*

```
[root@host ~]# kinit admin/admin

Password for admin/admin@PUNE.IBM.COM:
[root@ceph-mani-o7fdxp-node5 ~]# klist
Ticket cache: KCM:0
Default principal: admin/admin@PUNE.IBM.COM

Valid starting     Expires            Service principal
10/25/23 08:49:12  10/26/23 08:49:08  krbtgt/PUNE.IBM.COM@PUNE.IBM.COM
  renew until 10/25/23 08:49:12
```

11.14.3. NFS 特定的 Kerberos 设置

您需要为 NFS 服务器和客户端创建服务主体。对应的密钥存储在 keytab 文件中。这些主体需要设置 GSS_RPCSEC 所需的初始安全上下文。这些服务主体的格式与 nfs/@REALM 相似。您可以将 /etc/krb5.conf 文件从工作系统复制到 Ceph 节点。

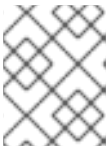
流程

1. *为该主机创建服务主体：*

```
[root@host ~]# kinit admin/admin

Password for admin/admin@PUNE.IBM.COM:
```

```
[root@host ~]# kadmin
Authenticating as principal admin/admin@PUNE.IBM.COM with password.
Password for admin/admin@PUNE.IBM.COM:
kadmin:  addprinc -randkey nfs/<hostname>.ibm.com
No policy specified for nfs/<hostname>.ibm.com@PUNE.IBM.COM; defaulting to no policy
Principal "nfs/<hostname>.ibm.com@PUNE.IBM.COM" created.
```

2. 将密钥添加到 keytab 文件中：



注意

在这一步中，您已位于 NFS 服务器上，并使用 kadmin 接口。在这里，keytab 操作反映了 NFS 服务器的 keytab。

```
kadmin:  ktadd nfs/<hostname>.ibm.com

Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type aes256-cts-hmac-sha384-192 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type aes128-cts-hmac-sha256-128 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type aes256-cts-hmac-sha1-96 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type aes128-cts-hmac-sha1-96 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type camellia256-cts-cmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type camellia128-cts-cmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com with kvno 2, encryption type arcfour-hmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
kadmin:
```

3. 在运行 NFS Ganesha 容器以及所有 NFS 客户端的所有 Ceph 节点上运行第 1 和 2 步。

11.14.4. NFS Ganesha 容器设置

按照以下步骤在 Ceph 环境中配置 NFS Ganesha 设置。

流程

1. 检索现有的 NFS Ganesha 容器配置：

```
[ceph: root@host /]# ceph orch ls --service-type nfs --export

service_type: nfs
service_id: c_ganesha
service_name: nfs.c_ganesha
placement:
  hosts:
  - host1
  - host2
  - host3
spec:
  port: 2049
```

2. *修改容器配置，将 **/etc/krb5.conf** 和 **/etc/krb5.keytab'** 文件传递给主机中的容器。NFS Ganesha 将在运行时引用这些文件，以验证传入的服务票据并确保 Ganesha 和 NFS 客户端(krb5p)之间的通信。*

```
[root@host ~]# cat nfs.yaml

service_type: nfs
service_id: c_ganesha
service_name: nfs.c_ganesha
placement:
  hosts:
  - host1
  - host2
  - host3
spec:
  port: 2049
extra_container_args:
  - "-v"
  - "/etc/krb5.keytab:/etc/krb5.keytab:ro"
  - "-v"
  - "/etc/krb5.conf:/etc/krb5.conf:ro"
```

3. *使修改后的 nfs.yaml 文件在 cephadm shell 中可用：*

```
[root@host ~]# cephadm shell --mount nfs.yaml:/var/lib/ceph/nfs.yaml

Inferring fsid ff1c1498-73ec-11ee-af38-fa163e9a17fd
Inferring config /var/lib/ceph/ff1c1498-73ec-11ee-af38-fa163e9a17fd/mon.ceph-msaini-qp49z7-node1-installer/config
Using ceph image with id 'fada497f9c5f' and tag 'ceph-7.0-rhel-9-containers-candidate-73711-20231018030025' created on 2023-10-18 03:03:39 +0000 UTC
registry-proxy.engineering.ibm.com/rh-osbs/rhceph@sha256:e66e5dd79d021f3204a183f5dbe4537d0c0e4b466df3b2cc4d50cc79c0f34d75
```

4. *验证该文件是否有所需的更改：*

```
[ceph: root@host /]# cat /var/lib/ceph/nfs.yaml

service_type: nfs
service_id: c_ganesha
service_name: nfs.c_ganesha
placement:
  hosts:
  - host1
  - host2
  - host3
spec:
  port: 2049
extra_container_args:
  - "-v"
  - "/etc/krb5.keytab:/etc/krb5.keytab:ro"
  - "-v"
  - "/etc/krb5.conf:/etc/krb5.conf:ro"
```

5. 将所需的更改应用到 NFS Ganesha 容器并重新部署容器：

```
[ceph: root@host /]# ceph orch apply -i /var/lib/ceph/nfs.yaml

Scheduled nfs.c_ganesha update...
[ceph: root@ceph-msaini-qp49z7-node1-installer /]# ceph orch redeploy nfs.c_ganesha
Scheduled to redeploy nfs.c_ganesha.1.0.ceph-msaini-qp49z7-node1-installer.sxzuts on host
'ceph-msaini-qp49z7-node1-installer'
Scheduled to redeploy nfs.c_ganesha.2.0.ceph-msaini-qp49z7-node2.psuvki on host 'ceph-
msaini-qp49z7-node2'
Scheduled to redeploy nfs.c_ganesha.0.0.ceph-msaini-qp49z7-node3.qizzvk on host 'ceph-
msaini-qp49z7-node3'
```

6. 验证重新部署的服务是否具有所需的更改：

```
[ceph: root@host /]# ceph orch ls --service-type nfs --export

service_type: nfs
service_id: c_ganesha
service_name: nfs.c_ganesha
placement:
  hosts:
  - ceph-msaini-qp49z7-node1-installer
  - ceph-msaini-qp49z7-node2
  - ceph-msaini-qp49z7-node3
extra_container_args:
- -v
- /etc/krb5.keytab:/etc/krb5.keytab:ro
- -v
- /etc/krb5.conf:/etc/krb5.conf:ro
spec:
  port: 2049
```

7. 修改导出定义，使其具有 **krb5*(krb5i、krb5i、krb5p)** 安全类型：

注意

您可以在完成上述设置后创建此类导出。

```
[ceph: root@host /]# ceph nfs export info c_ganesha /exp1

{
  "access_type": "RW",
  "clients": [],
  "cluster_id": "c_ganesha",
  "export_id": 1,
  "fsal": {
    "fs_name": "fs1",
    "name": "CEPH",
    "user_id": "nfs.c_ganesha.1"
  },
  "path": "/volumes/_nogroup/exp1/81f9a67e-ddf1-4b5a-9fe0-d87effc7ca16",
  "protocols": [
    4
```

```
  ],
  "pseudo": "/exp1",
  "sectype": [
    "krb5"
  ],
  "security_label": true,
  "squash": "none",
  "transports": [
    "TCP"
  ]
}
```

11.14.5. NFS 客户端侧操作

以下是 NFS 客户端可以进行的一些操作。

流程

1. 创建服务主体：

```
kadmin:  addprinc -randkey nfs/<hostname>.ibm.com@PUNE.IBM.COM
No policy specified for nfs/<hostname>.ibm.com@PUNE.IBM.COM; defaulting to no policy
Principal "nfs/<hostname>.ibm.com@PUNE.IBM.COM" created.
kadmin:  ktadd nfs/<hostname>.ibm.com@PUNE.IBM.COM
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
aes256-cts-hmac-sha384-192 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
aes128-cts-hmac-sha256-128 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
aes256-cts-hmac-sha1-96 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
aes128-cts-hmac-sha1-96 added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
camellia256-cts-cmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
camellia128-cts-cmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
Entry for principal nfs/<hostname>.ibm.com@PUNE.IBM.COM with kvno 2, encryption type
arcfour-hmac added to keytab [FILE:/etc/krb5.keytab](file:///etc/krb5.keytab).
```

2. 重启 **rpc.gssd** 服务，使修改/新 keytab 文件生效：

```
# systemctl restart rpc-gssd
```

3. 挂载 NFS 导出：

语法

```
[root@host ~]# mount -t nfs -o vers=4.1,port=2049 <IP>:/<export_name> >mount_point>
```

示例

```
mount -t nfs -o vers=4.1,port=2049 10.8.128.233:/ganesha /mnt/test/
```

4. *创建用户。挂载 NFS 导出后，常规用户将用于挂载的导出。这些常规用户（通常是系统中的本地用户或来自集中系统的用户（如 LDAP/AD）需要成为 Kerberos 设置的一部分。根据设置类型，还需要在 KDC 中创建本地用户。*

11.14.6. 验证设置

按照以下步骤验证设置。

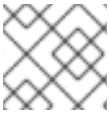
流程

- *以普通用户身份访问导出，**没有** Kerberos 票据：*

```
[user@host ~]$ klist

klist: Credentials cache 'KCM:1001' not found

[user@host ~]$ cd /mnt
-bash: cd: /mnt: Permission denied
```

- *以普通用户身份访问导出，**使用** Kerberos 票据：*

```
[user@host ~]$ kinit sachin

Password for user@PUNE.IBM.COM:

[user@host ~]$ klist
Ticket cache: KCM:1001
Default principal: user@PUNE.IBM.COM

Valid starting     Expires            Service principal
10/27/23 12:57:21  10/28/23 12:57:17  krbtgt/PUNE.IBM.COM@PUNE.IBM.COM
  renew until 10/27/23 12:57:21

[user@host ~]$ cd /mnt

[user@host mnt]$ klist

Ticket cache: KCM:1001
Default principal: user@PUNE.IBM.COM

Valid starting     Expires            Service principal
10/27/23 12:57:21  10/28/23 12:57:17  krbtgt/PUNE.IBM.COM@PUNE.IBM.COM
renew until 10/27/23 12:57:21
10/27/23 12:57:28  10/28/23 12:57:17  nfs/ceph-msaini-qp49z7-node1-installer.ibm.com@
renew until 10/27/23 12:57:21
Ticket server: nfs/ceph-msaini-qp49z7-node1-installer.ibm.com@PUNE.IBM.COM
```



注意

*注意：在客户端上观察到 **nfs**/ 服务的 Tickets。*

第 12 章 SNMP TRAP 配置

作为存储管理员，您可以在 Red Hat Ceph Storage 集群中部署和配置简单的网络管理协议(SNMP)网关，从 Prometheus Alertmanager 接收警报并将其作为 SNMP 陷入集群。

12.1. 简单的网络管理协议

简单的网络管理协议(SNMP)是最广泛使用的开放协议之一，用于监控不同硬件和软件平台的分布式系统和设备。Ceph 的 SNMP 集成侧重于将警报从 Prometheus Alertmanager 集群转发到网关守护进程。网关守护进程将警报转换为 SNMP 通知，并将其发送到指定的 SNMP 管理平台。网关守护进程来自 **snmp_notifier_project**，它通过验证和加密提供 SNMP V2c 和 V3 支持。

Red Hat Ceph Storage SNMP 网关服务默认部署一个网关实例。您可以通过提供放置信息来增加这个值。但是，如果您启用多个 SNMP 网关守护进程，则 SNMP 管理平台会收到同一事件的多个通知。

SNMP 陷阱是警报消息，Prometheus Alertmanager 会将这些警报发送到 SNMP notifier，然后在给定的警报标签中查找对象标识符(OID)。每个 SNMP trap 都有一个唯一的 ID，允许它将带有更新状态的其他 trap 发送到给定的 SNMP 轮询程序。SNMP hook 在 Ceph 健康检查中，使得每个健康警告都会生成特定的 SNMP 陷阱。

为了正常工作并将设备状态的信息传送到用户 monitor，SNMP 依赖于几个组件。makeup SNMP 有四个主要组件：

- **SNMP Manager**- SNMP manager，也称为管理站，是运行网络监控平台的计算机。具有轮询 SNMP 功能的设备并从中检索数据的平台。SNMP Manager 查询代理，接收来自代理的响应，并确认来自代理的异步事件。
- **SNMP Agent** - SNMP 代理是在要管理的系统上运行并包含系统的 MIB 数据库的程序。它们收集带宽和磁盘空间、聚合数据并将其发送到管理信息基础(MIB)。
- **管理信息基础(MIB)** - 这些组件包含在 SNMP 代理中。SNMP 管理器将此用作数据库，并要求代理访问特定信息。网络管理系统(NMS)需要此信息。NMS 轮询代理以从这些文件中获取信息，然后继续将其转换为图形并显示用户可以查看的图形。MiB 包含由网络设备决定的统计信息和控制值。
- **SNMP 设备**

以下 SNMP 版本在网关实施中兼容和支持：

- **V2c** - 使用一个没有身份验证的社区字符串，并容易受到外部攻击。
- **V3 authNoPriv** - 在没有加密的情况下使用用户名和密码身份验证。
- **V3 authPriv** - 通过加密将用户名和密码身份验证用于 SNMP 管理平台。

重要

在使用 SNMP 陷阱时，请确保您的版本号具有正确的安全配置，以最大程度降低 SNMP 固有的漏洞，防止您的网络不受未授权用户保护。

12.2. 配置 SNMPTRAPD

在部署 **snmp-gateway** 前务必要配置简单的网络管理协议(SNMP)目标，因为 **snmptrapd** 守护进程包含您在创建 **snmp-gateway** 服务时指定的 auth 设置。

SNMP 网关功能提供了一种将 Prometheus 堆栈中生成的警报公开给 SNMP 管理平台的方法。您可以根据 **snmptrapd** 工具配置 SNMP 陷入目的地。此工具允许您建立一个或多个 SNMP 陷阱监听程序。

以下参数对于配置很重要：

- **engine-id** 是设备的唯一标识符，hex 格式，SNNV3 网关需要。红帽建议在这个参数中使用 '8000C53F_CLUSTER_FSID_WITHOUT_DASHES_'。
- **snmp-community** 是 SNMP_COMMUNITY_FOR_SNMPV2 参数，对于 SNMPV2c 网关是 **public**。
- **auth-protocol** 是 AUTH_PROTOCOL， 它是 SNMPV3 网关的强制性，默认情况下是 **SHA**。
- 对于 SNMPV3 网关，**privacy-protocol**（为 PRIVACY_PROTOCOL）是必须的。
- PRIVACY_PASSWORD 是 SNMPV3 网关使用加密所必需的。
- SNMP_V3_AUTH_USER_NAME 是用户名，对于 SNMPV3 网关是必需的。
- SNMP_V3_AUTH_PASSWORD 是密码，是 SNMPV3 网关所必需的。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对节点的根级别访问权限。
- 在 Red Hat Enterprise Linux 系统上安装 **firewalld**。

流程

1. 在 SNMP 管理主机上，安装 SNMP 软件包：

   示例

   ```
   [root@host01 ~]# dnf install -y net-snmp-utils net-snmp
   ```

2. 为 SNMP 打开端口 162 以接收警报：

   示例

   ```
   [root@host01 ~]# firewall-cmd --zone=public --add-port=162/udp
   [root@host01 ~]# firewall-cmd --zone=public --add-port=162/udp --permanent
   ```

3. 实施管理信息基础(MIB)，以便对 SNMP 通知有意义，并增强目的地主机上的 SNMP 支持。从主仓库中复制 raw 文件：https://github.com/ceph/ceph/blob/master/monitoring/snmp/CEPH-MIB.txt

   示例

   ```
   [root@host01 ~]# curl -o CEPH_MIB.txt -L
   https://raw.githubusercontent.com/ceph/ceph/master/monitoring/snmp/CEPH-MIB.txt
   [root@host01 ~]# scp CEPH_MIB.txt root@host02:/usr/share/snmp/mibs
   ```

4. 创建 **snmptrapd** 目录。

示例

```
[root@host01 ~]# mkdir /root/snmptrapd/
```

5. 根据 SNMP 版本为每个协议创建 **snmptrapd** 目录中的配置文件：

语法

```
format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/Value Pair Array:\n%v \n -------------- \n
createuser -e 0x_ENGINE_ID_ SNMPV3_AUTH_USER_NAME AUTH_PROTOCOL SNMP_V3_AUTH_PASSWORD PRIVACY_PROTOCOL PRIVACY_PASSWORD
authuser log,execute SNMP_V3_AUTH_USER_NAME
authCommunity   log,execute,net SNMP_COMMUNITY_FOR_SNMPV2
```

- 对于 SNMPV2c，按如下所示创建 **snmptrapd_public.conf** 文件：

  示例

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/Value Pair Array:\n%v \n -------------- \n

  authCommunity   log,execute,net public
  ```

  此处的 **public** 设置必须与部署 **snmp-gateway** 服务时使用的 **snmp_community** 设置匹配。

- 对于仅限通过身份验证的 SNMPV3，请创建 **snmptrapd_auth.conf** 文件，如下所示：

  示例

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/Value Pair Array:\n%v \n -------------- \n
  createuser -e 0x8000C53Ff64f341c655d11eb8778fa163e914bcc myuser SHA mypassword
  authuser log,execute myuser
  ```

  0x8000C53Ff64f341c655d11eb8778fa163e914bcc 字符串是 **engine_id**，**myuser** 和 **mypassword** 是凭证。密码安全性由 **SHA** 算法定义。

  这与部署 **snmp-gateway** 守护进程的设置对应。

  示例

  ```
  snmp_v3_auth_username: myuser
  snmp_v3_auth_password: mypassword
  ```

- 对于带有身份验证和加密的SNMPV3，请创建 **snmptrapd_authpriv.conf** 文件，如下所示：

  示例

  ```
  format2 %V\n% Agent Address: %A \n Agent Hostname: %B \n Date: %H - %J - %K - %L - %M - %Y \n Enterprise OID: %N \n Trap Type: %W \n Trap Sub-Type: %q \n Community/Infosec Context: %P \n Uptime: %T \n Description: %W \n PDU Attribute/Value Pair Array:\n%v \n ---------- \n
  createuser -e 0x8000C53Ff64f341c655d11eb8778fa163e914bcc myuser SHA mypassword DES mysecret
  authuser log,execute myuser
  ```

  0x8000C53Ff64f341c655d11eb8778fa163e914bcc 字符串是 **engine_id**，**myuser** 和 **mypassword** 是凭证。密码安全性由 **SHA** 算法定义，而 **DES** 是隐私加密的类型。

  这与部署 **snmp-gateway** 守护进程的设置对应。

  示例

  ```
  snmp_v3_auth_username: myuser
  snmp_v3_auth_password: mypassword
  snmp_v3_priv_password: mysecret
  ```

6. 在SNMP 管理主机上运行守护进程：

   语法

   ```
   /usr/sbin/snmptrapd -M /usr/share/snmp/mibs -m CEPH-MIB.txt -f -C -c /root/snmptrapd/CONFIGURATION_FILE -Of -Lo :162
   ```

   示例

   ```
   [root@host01 ~]# /usr/sbin/snmptrapd -M /usr/share/snmp/mibs -m CEPH-MIB.txt -f -C -c /root/snmptrapd/snmptrapd_auth.conf -Of -Lo :162
   ```

7. 如果存储集群上触发了任何警报，您可以监控SNMP 管理主机上的输出。验证SNMP 陷阱以及 MIB 解码的陷阱。

   示例

   ```
   NET-SNMP version 5.8
    Agent Address: 0.0.0.0
    Agent Hostname: <UNKNOWN>
    Date: 15 - 5 - 12 - 8 - 10 - 4461391
    Enterprise OID: .
    Trap Type: Cold Start
    Trap Sub-Type: 0
    Community/Infosec Context: TRAP2, SNMP v3, user myuser, context
    Uptime: 0
    Description: Cold Start
    PDU Attribute/Value Pair Array:
   .iso.org.dod.internet.mgmt.mib-2.1.3.0 = Timeticks: (292276100) 3 days, 19:52:41.00
   .iso.org.dod.internet.snmpV2.snmpModules.1.1.4.1.0 = OID:
   .iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheus.promM
   ```

> *gr.promMgrPrometheusInactive*
> *.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheus.promMgr.promMgrPrometheusInactive.1 = STRING: "1.3.6.1.4.1.50495.1.2.1.6.2[alertname=CephMgrPrometheusModuleInactive]"*
> *.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheus.promMgr.promMgrPrometheusInactive.2 = STRING: "critical"*
> *.iso.org.dod.internet.private.enterprises.ceph.cephCluster.cephNotifications.prometheus.promMgr.promMgrPrometheusInactive.3 = STRING: "Status: critical*
> *- Alert: CephMgrPrometheusModuleInactive*
> *Summary: Ceph's mgr/prometheus module is not available*
> *Description: The mgr/prometheus module at 10.70.39.243:9283 is unreachable. This could mean that the module has been disabled or the mgr itself is down.*
> *Without the mgr/prometheus module metrics and alerts will no longer function. Open a shell to ceph and use 'ceph -s' to determine whether the mgr is active. If the mgr is not active, restart it, otherwise you can check the mgr/prometheus module is loaded with 'ceph mgr module ls' and if it's not listed as enabled, enable it with 'ceph mgr module enable prometheus'"*

在上例中，在禁用 Prometheus 模块后生成一个警报。

其它资源

- *请参阅 Red Hat Ceph Storage Operations 指南中的 部署 SNMP 网关 部分。*

12.3. 部署 SNMP 网关

您可以使用 SNMPV2c 或 SNMPV3 部署简单网络管理协议(SNMP)网关。部署 SNMP 网关的方法有两种：

1. *通过创建凭据文件。*
2. *通过创建一个带有所有详情的服务配置 yaml 文件。*

您可以使用以下参数根据版本部署 SNMP 网关：

- ***service_type** 是 **snmp-gateway**。*
- ***service_name** 是任何用户定义的字符串。*
- ***count** 是要在存储集群中部署的 SNMP 网关的数量。*
- ***snmp_destination** 参数必须为 hostname:port 格式。*
- ***engine-id** 是设备的唯一标识符，hex 格式，SNNV3 网关需要。红帽建议为此参数使用 '8000C53F_CLUSTER_FSID_WITHOUT_DASHES_'。*
- *对于 SNMPV2c 网关，**snmp_community** 参数为 **public**。*
- *对于 SNMPV3 网关，**auth-protocol** 是强制的，默认为 **SHA**。*
- *对于需要 进行身份验证和加密的 SNMPV3 网关，**privacy-protocol** 是必须的。*
- *端口默认为 **9464**。*
- *您必须提供 **-i FILENAME**，将 secret 和密码传递给编配器。*

在部署或更新 SNMP 网关服务后，Prometheus Alertmanager 配置会自动更新，以将对象识别符转发到 SNMP 网关守护进程以进行进一步处理。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对节点的根级别访问权限。
- 在目标主机上配置 **snmptrapd**，这是 SNMP 管理主机。

流程

1. 登录到 Cephadm shell：

   示例

   ```
   [root@host01 ~]# cephadm shell
   ```

2. 为需要部署 SNMP 网关的主机创建一个标签：

   语法

   ```
   ceph orch host label add HOSTNAME snmp-gateway
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph orch host label add host02 snmp-gateway
   ```

3. 根据 SNMP 版本创建凭证文件或服务配置文件：

   - 对于 SNMPV2c，按如下所示创建该文件：

     示例

     ```
     [ceph: root@host01 /]# cat snmp_creds.yml

     snmp_community: public
     ```

     或者

     示例

     ```
     [ceph: root@host01 /]# cat snmp-gateway.yml

     service_type: snmp-gateway
     service_name: snmp-gateway
     placement:
       count: 1
     spec:
       credentials:
         snmp_community: public
     ```

```
  port: 9464
  snmp_destination: 192.168.122.73:162
  snmp_version: V2c
```

- 对于仅限使用身份验证的SNMPV3，请按如下所示创建该文件：

  示例

```
[ceph: root@host01 /]# cat snmp_creds.yml

snmp_v3_auth_username: myuser
snmp_v3_auth_password: mypassword
```

  或者

  示例

```
[ceph: root@host01 /]# cat snmp-gateway.yml

service_type: snmp-gateway
service_name: snmp-gateway
placement:
  count: 1
spec:
  credentials:
    snmp_v3_auth_password: mypassword
    snmp_v3_auth_username: myuser
  engine_id: 8000C53Ff64f341c655d11eb8778fa163e914bcc
  port: 9464
  snmp_destination: 192.168.122.1:162
  snmp_version: V3
```

- 对于使用验证和加密的SNMPV3，请按如下所示创建该文件：

  示例

```
[ceph: root@host01 /]# cat snmp_creds.yml

snmp_v3_auth_username: myuser
snmp_v3_auth_password: mypassword
snmp_v3_priv_password: mysecret
```

  或者

  示例

```
[ceph: root@host01 /]# cat snmp-gateway.yml

service_type: snmp-gateway
service_name: snmp-gateway
placement:
  count: 1
spec:
  credentials:
```

```
  snmp_v3_auth_password: mypassword
  snmp_v3_auth_username: myuser
  snmp_v3_priv_password: mysecret
engine_id: 8000C53Ff64f341c655d11eb8778fa163e914bcc
port: 9464
snmp_destination: 192.168.122.1:162
snmp_version: V3
```

4. 运行 **ceph orch** 命令：

语法

```
ceph orch apply snmp-gateway --snmp_version=V2c_OR_V3 --
destination=SNMP_DESTINATION [--port=PORT_NUMBER]\
[--engine-id=8000C53F_CLUSTER_FSID_WITHOUT_DASHES_] [--auth-
protocol=MDS_OR_SHA] [--privacy_protocol=DES_OR_AES] -i FILENAME
```

或者

语法

```
ceph orch apply -i FILENAME.yml
```

- 对于SNMPV2c，使用 **snmp_creds** 文件，使用 **snmp-version** 作为 **V2c** 运行 **ceph orch** 命令：

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V2c --
  destination=192.168.122.73:162 --port=9464 -i snmp_creds.yml
  ```

- 对于仅限使用身份验证的SNMPV3，使用 **snmp_creds** 文件，使用 **snmp-version** 作为 **V3** 和 **engine-id** 运行 **ceph orch** 命令：

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V3 --engine-
  id=8000C53Ff64f341c655d11eb8778fa163e914bcc--destination=192.168.122.73:162 -i
  snmp_creds.yml
  ```

- 对于带有验证和加密的SNMPV3，带有 **snmp_creds** 文件，运行 **ceph orch** 命令，**snmp-version** 为 **V3**, **privacy-protocol**, 和 **engine-id**:

  示例

  ```
  [ceph: root@host01 /]# ceph orch apply snmp-gateway --snmp-version=V3 --engine-
  id=8000C53Ff64f341c655d11eb8778fa163e914bcc--destination=192.168.122.73:162 --
  privacy-protocol=AES -i snmp_creds.yml
  ```

  或者

- 对于所有SNMP 版本（使用 **snmp-gateway** 文件），请运行以下命令：

  示例

例如

```
[ceph: root@host01 /]# ceph orch apply -i snmp-gateway.yml
```

其它资源

- 请参阅 Red Hat Ceph Storage Operations Guide 中的 Configuring 'snmptrapd' 部分。

第 13 章 处理节点故障

作为存储管理员，您可以在存储集群中遇到整个节点故障，处理节点故障与处理磁盘故障类似。当节点出现故障时，而不是 Ceph 对只有一个磁盘恢复放置组(PG)，必须恢复该节点上的磁盘上的所有 PG。Ceph 将检测 OSD 是否都停止，并且自动启动恢复过程，称为自我修复。

有三个节点故障场景。

- 使用故障节点的 root 和 Ceph OSD 磁盘替换节点。
- 通过重新安装操作系统和使用来自故障节点的 Ceph OSD 磁盘来替换节点。
- 通过重新安装操作系统和使用所有新的 Ceph OSD 磁盘来替换节点。

有关每个节点替换场景的高级工作流，请参阅 link:https://access.redhat.com/documentation/zh-cn/red_hat_ceph_storage/7/html-single/operations_guide/#ops_workflow-for replace-a-node[用于替换节点] 的工作流。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 一个出现故障的节点。

13.1. 在添加或删除节点前的注意事项

Ceph 的其中一个未完成功能是能够在运行时添加或删除 Ceph OSD 节点。这意味着，您可以在不关闭存储集群的情况下调整存储集群容量或替换硬件的大小。

在存储集群 处于**降级状态** 时为 Ceph 客户端提供服务，也具有操作优势。例如，您可以在常规工作时间内添加或删除硬件，而不是在工作时间外或周末操作。但是，添加和删除 Ceph OSD 节点可能会对性能产生重大影响。

在添加或删除 Ceph OSD 节点前，请考虑以下对存储集群性能的影响：

- 无论您要扩展或减少存储容量，添加或删除 Ceph OSD 节点，都会降低回填存储集群重新平衡。在进行重新平衡期间，Ceph 使用额外的资源，这可能会影响存储集群性能。
- 在生产环境的 Ceph 存储集群中，Ceph OSD 节点具有特定的硬件配置，有助于实现特定类型的存储策略。
- 由于 Ceph OSD 节点是 CRUSH 层次结构中的一部分，因此添加或删除节点的性能通常会影响使用 CRUSH 规则集的池的性能。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage 策略指南 。

13.2. 替换节点的工作流

有三个节点故障场景。在替换节点时，为每个场景使用这些高级别工作流。

- 使用故障节点的 root 和 Ceph OSD 磁盘替换节点
- 通过重新安装操作系统和使用来自故障节点的 Ceph OSD 磁盘来替换节点

- 通过重新安装操作系统和使用所有新的 Ceph OSD 磁盘来替换节点

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 一个出现故障的节点。

13.2.1. 使用故障节点的 root 和 Ceph OSD 磁盘替换节点

使用故障节点的 root 和 Ceph OSD 磁盘替换节点。

流程

1. 禁用回填。

   语法

   ```
   ceph osd set noout
   ceph osd set noscrub
   ceph osd set nodeep-scrub
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

2. 替换节点，从旧节点获取磁盘，并将它们添加到新节点。

3. 启用回填。

   语法

   ```
   ceph osd unset noout
   ceph osd unset noscrub
   ceph osd unset nodeep-scrub
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset noscrub
   [ceph: root@host01 /]# ceph osd unset nodeep-scrub
   ```

13.2.2. 通过重新安装操作系统和使用来自故障节点的 Ceph OSD 磁盘来替换节点

重新安装操作系统，并使用故障节点的 Ceph OSD 磁盘替换节点。

流程

1. 禁用回填。

语法

```
ceph osd set noout
ceph osd set noscrub
ceph osd set nodeep-scrub
```

示例

```
[ceph: root@host01 /]# ceph osd set noout
[ceph: root@host01 /]# ceph osd set noscrub
[ceph: root@host01 /]# ceph osd set nodeep-scrub
```

2. 创建 Ceph 配置的备份。

语法

```
cp /etc/ceph/ceph.conf /PATH_TO_BACKUP_LOCATION/ceph.conf
```

示例

```
[ceph: root@host01 /]# cp /etc/ceph/ceph.conf /some/backup/location/ceph.conf
```

3. 替换节点，再添加来自故障节点的 Ceph OSD 磁盘。

4. 将磁盘配置为 JBOD。

注意

这应该由存储管理员完成。

5. 安装操作系统。有关操作系统要求的更多信息，请参阅 Red Hat Ceph Storage 的操作系统要求。有关安装操作系统的更多信息，请参阅 Red Hat Enterprise Linux 产品文档。

注意

这应该由系统管理员完成。

6. 恢复 Ceph 配置。

语法

```
cp /PATH_TO_BACKUP_LOCATION/ceph.conf /etc/ceph/ceph.conf
```

示例

```
[ceph: root@host01 /]# cp /some/backup/location/ceph.conf /etc/ceph/ceph.conf
```

7. 使用 Ceph 编排器命令将新节点添加到存储集群。Ceph 守护进程自动放置到对应的节点上。有关更多信息，请参阅 添加 Ceph OSD 节点。

8. 启用回填。

语法

```
ceph osd unset noout
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

示例

```
[ceph: root@host01 /]# ceph osd unset noout
[ceph: root@host01 /]# ceph osd unset noscrub
[ceph: root@host01 /]# ceph osd unset nodeep-scrub
```

13.2.3. 通过重新安装操作系统和使用所有新的 Ceph OSD 磁盘来替换节点

重新安装操作系统，并使用所有新的 Ceph OSD 磁盘替换节点。

流程

1. 禁用回填。

   语法

   ```
   ceph osd set noout
   ceph osd set noscrub
   ceph osd set nodeep-scrub
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

2. 从存储集群中移除故障节点上的所有 OSD。有关更多信息，请参阅 删除 Ceph OSD 节点。

3. 创建 Ceph 配置的备份。

   语法

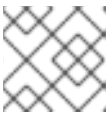
   ```
   cp /etc/ceph/ceph.conf /PATH_TO_BACKUP_LOCATION/ceph.conf
   ```

   示例

   ```
   [ceph: root@host01 /]# cp /etc/ceph/ceph.conf /some/backup/location/ceph.conf
   ```

4. 替换节点，再添加来自故障节点的 Ceph OSD 磁盘。

5. 将磁盘配置为 JBOD。



注意

这应该由存储管理员完成。

6. 安装操作系统。有关操作系统要求的更多信息，请参阅 Red Hat Ceph Storage 的操作系统要求。有关安装操作系统的更多信息，请参阅 Red Hat Enterprise Linux 产品文档。

注意

这应该由系统管理员完成。

7. 使用 Ceph 编排器命令将新节点添加到存储集群。Ceph 守护进程自动放置到对应的节点上。有关更多信息，请参阅 添加 Ceph OSD 节点。

8. 启用回填。

语法

```
ceph osd unset noout
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

示例

```
[ceph: root@host01 /]# ceph osd unset noout
[ceph: root@host01 /]# ceph osd unset noscrub
[ceph: root@host01 /]# ceph osd unset nodeep-scrub
```

13.3. 性能考虑

在添加或删除 Ceph OSD 节点时，以下因素通常会影响存储集群的性能：

- Ceph 客户端将负载放到 Ceph 的 I/O 接口上；也就是说，客户端会将负载放入池。池映射到 CRUSH 规则集。底层 CRUSH 层次结构允许 Ceph 在故障域之间放置数据。如果底层 Ceph OSD 节点涉及有高客户端负载的池，客户端负载可能会显著影响恢复时间和降低性能。因为写入操作需要数据复制才能进行持久性，特别是写入密集型客户端负载可能会增加存储集群的时间来恢复。

- 通常，您添加或删除的容量会影响存储集群恢复的时间。另外，您添加或删除的节点的存储密度可能会影响恢复时间。例如，具有 36 个 OSD 的节点通常需要更长的时间来恢复具有 12 个 OSD 的节点。

- 移除节点时，您需要确保有足够的备用容量，以便您的系统不会达到**全满比率**或**接近满**比率。如果存储集群达到**全满比率**，Ceph 将挂起写操作以防止数据丢失。

- Ceph OSD 节点映射到至少一个 Ceph CRUSH 层次结构，层次结构则映射到至少一个池。在添加或删除 Ceph OSD 节点时，使用 CRUSH 规则集的每个池都会遇到性能影响。

- 复制池往往使用更多网络带宽来复制数据的深度副本，而纠删代码池则倾向于使用更多 CPU 来计算 **k+m** 编码区块。数据存在的更多副本，存储集群需要更长的时间来恢复。例如，一个大于 **k+m** 块的大池或一个要比同一数据副本较少的复制池恢复的时间要更长。

- 驱动器、控制器和网络接口卡都有可能影响恢复时间的吞吐量特征。通常，具有更高吞吐量的节点（如 10 Gbps 和 SSD）可以比具有较低吞吐量的节点快速恢复，如 1 Gbps 和 SATA 驱动器。

13.4. 添加或删除节点的建议

红帽建议在一个节点中逐一添加或删除一个 OSD，并在继续执行下一个 OSD 前恢复存储集群。这有助于最大程度降低对存储集群性能的影响。请注意，如果某个节点失败，您可能需要一次性更改整个节点，而不是一次更改一个 OSD。

删除 OSD：

- 使用 使用 Ceph 编排器删除 OSD 守护进程。

添加 OSD：

- 在所有可用设备 上使用部署 Ceph OSD。
- 使用高级服务规范部署 Ceph OSD。
- 在 特定设备和主机 上使用部署 Ceph OSD。

在添加或删除 Ceph OSD 节点时，请考虑其他持续进程也会影响存储集群性能。要减少对客户端 I/O 的影响，红帽向您推荐以下几项：

计算容量

在移除 Ceph OSD 节点之前，请确保存储集群可以回填所有 OSD 的内容，而不会达到**全满比率**。达到 **全满比率** 将导致存储集群拒绝写操作。

临时禁用清理

清理是确保存储集群数据的持久性非常重要，但这是资源密集型。在添加或删除 Ceph OSD 节点之前，禁用清理和深度清理，并使当前清理操作在继续之前完成。

```
ceph osd set noscrub
ceph osd set nodeep-scrub
```

在添加或删除 Ceph OSD 节点且存储集群返回 **active+clean** 状态后，取消设置 **noscrub** 和 **nodeep-scrub** 设置。

```
ceph osd unset noscrub
ceph osd unset nodeep-scrub
```

限制回填和恢复

如果您有合理的数据持久性，则处于 **degraded** 状态应该不会出现问题。例如，您可以使用 **osd_pool_default_size = 3** 和 **osd_pool_default_min_size = 2** 来运行存储集群。您可以调整存储集群以最快的恢复时间，但这样做会对 Ceph 客户端 I/O 性能造成重大影响。要保持最高的 Ceph 客户端 I/O 性能，请限制回填和恢复操作，并允许它们花费更长的时间。

```
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

您还可以考虑设置 sleep 和 delay 参数，如 **osd_recovery_sleep**。

增加放置组数量

最后，如果您扩展存储集群的大小，可能需要增加放置组的数量。如果您确定需要扩展放置组数量，红帽建议在放置组数量中进行增量增长。通过显著数量增加放置组的数量会导致性能下降。

13.5. 添加 CEPH OSD 节点

要扩展 Red Hat Ceph Storage 集群的容量，您可以添加 OSD 节点。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 具有网络连接的置备节点。

流程

1. 通过短主机名验证存储集群中的其他节点是否可以访问新节点。
2. 临时禁用清理：

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

3. 限制回填和恢复功能：

   语法

   ```
   ceph tell DAEMON_TYPE.* injectargs --OPTION_NAME VALUE [--OPTION_NAME VALUE]
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph tell osd.* injectargs --osd-max-backfills 1 --osd-recovery-max-active 1 --osd-recovery-op-priority 1
   ```

4. 将集群的公共 SSH 密钥提取到文件夹：

   语法

   ```
   ceph cephadm get-pub-key > ~/PATH
   ```

   示例

   ```
   [ceph: root@host01 /]# ceph cephadm get-pub-key > ~/ceph.pub
   ```

5. 将 ceph 集群的公共 SSH 密钥复制到新主机上的 root 用户的 **authorized_keys** 文件中：

   语法

   ```
   ssh-copy-id -f -i ~/PATH root@HOST_NAME_2
   ```

   示例

   ```
   [ceph: root@host01 /]# ssh-copy-id -f -i ~/ceph.pub root@host02
   ```

6. 将新节点添加到 CRUSH map：

语法

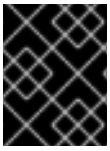
```
ceph orch host add NODE_NAME IP_ADDRESS
```

示例

```
[ceph: root@host01 /]# ceph orch host add host02 10.10.128.70
```

7. 为节点上的每个磁盘添加一个 OSD 到存储集群。

- 在所有可用设备上使用部署 Ceph OSD。
- 使用高级服务规范部署 Ceph OSD。
- 在 特定设备和主机 上使用部署 Ceph OSD。



重要

将 OSD 节点添加到 Red Hat Ceph Storage 集群时，红帽建议一次添加一个 OSD 守护进程，并允许集群在进入下一个 OSD 前恢复到 **active+clean** 状态。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage Configuration Guide 的在运行时设置特定配置设置部分。
- 如需了解有关将节点放置到 CRUSH 层次结构中的相应位置的详细信息，请参阅 Red Hat Ceph Storage Storage 策略指南中的添加 Bucket 和 Moving a Bucket 部分。

13.6. 删除 CEPH OSD 节点

要减少存储集群的容量，请删除 OSD 节点。

警告

在移除 Ceph OSD 节点之前，请确保存储集群可以回填所有 OSD 的内容，而无需达到**全满比率**。达到 **全满比率** 将导致存储集群拒绝写操作。

先决条件

- 一个正在运行的 Red Hat Ceph Storage 集群。
- 对存储集群中所有节点的根级别访问权限。

流程

1. 检查存储集群的容量：

语法

```
ceph df
rados df
ceph osd df
```

2. 临时禁用清理：

语法

```
ceph osd set noscrub
ceph osd set nodeep-scrub
```

3. 限制回填和恢复功能：

语法

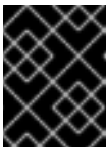
```
ceph tell DAEMON_TYPE.* injectargs --OPTION_NAME VALUE [--OPTION_NAME VALUE]
```

示例

```
[ceph: root@host01 /]# ceph tell osd.* injectargs --osd-max-backfills 1 --osd-recovery-max-active 1 --osd-recovery-op-priority 1
```

4. 从存储集群中移除节点上的每个 OSD：

- 使用 使用 Ceph 编排器删除 OSD 守护进程。



重要

从存储集群中移除 OSD 节点时，红帽建议在节点中一次删除一个 OSD，并允许集群恢复到 **active+clean** 状态，然后继续移除下一个 OSD。

a. 移除 OSD 后，检查以验证存储集群是否没有达到 **near-full 比率**：

语法

```
ceph -s
ceph df
```

b. 重复此步骤，直到将节点上的所有 OSD 从存储集群中移除。

5. 删除所有 OSD 后，删除主机：

- 使用 Ceph 编排器删除主机。

其它资源

- 如需了解更多详细信息，请参阅 Red Hat Ceph Storage 配置指南中的 在运行时设置特定配置部分。

13.7. 模拟节点故障

要模拟硬节点失败，请关闭节点并重新安装操作系统。

先决条件

- *一个正常运行的 Red Hat Ceph Storage 集群。*
- *对存储集群中所有节点的 root 级别访问。*

流程

1. *检查存储集群的容量以了解删除节点的影响：*

   示例

   ```
   [ceph: root@host01 /]# ceph df
   [ceph: root@host01 /]# rados df
   [ceph: root@host01 /]# ceph osd df
   ```

2. *（可选）禁用恢复和回填：*

   示例

   ```
   [ceph: root@host01 /]# ceph osd set noout
   [ceph: root@host01 /]# ceph osd set noscrub
   [ceph: root@host01 /]# ceph osd set nodeep-scrub
   ```

3. *关闭节点。*

4. *如果要更改主机名，请从 CRUSH 映射中删除节点：*

   示例

   ```
   [ceph: root@host01 /]# ceph osd crush rm host03
   ```

5. *检查存储集群的状态：*

   示例

   ```
   [ceph: root@host01 /]# ceph -s
   ```

6. *在节点上重新安装操作系统。*

7. *添加新节点：*

   - *使用 Ceph 编排器添加主机。*

8. *（可选）启用恢复和回填：*

   示例

   ```
   [ceph: root@host01 /]# ceph osd unset noout
   [ceph: root@host01 /]# ceph osd unset noscrub
   [ceph: root@host01 /]# ceph osd unset nodeep-scrub
   ```

9. *检查 Ceph 的健康状况：*

示例

```
[ceph: root@host01 /]# ceph -s
```

其它资源

- *如需了解更多详细信息，请参阅 Red Hat Ceph Storage 安装指南。*

第 14 章 处理数据中心故障

作为存储管理员，您可以采取预防措施来避免数据中心故障。这些防止措施包括：

- 配置数据中心基础架构.
- 在 CRUSH map 层次结构中设置故障域。
- 在域中设计故障节点。

先决条件

- 一个正常运行的 Red Hat Ceph Storage 集群。
- 对存储集群中所有节点的根级别访问权限。

14.1. 避免数据中心故障

配置数据中心基础架构

扩展集群中的每个数据中心都可以有不同的存储集群配置，以反映本地的功能和依赖项。设置数据中心之间的复制，以帮助保留数据。如果一个数据中心失败，则存储集群中的其他数据中心包含数据的副本。

在 CRUSH map 层次结构中设置故障域

故障或故障转移，域是存储集群中域的冗余副本。如果活动域失败，故障域将变为活动域。

默认情况下，CRUSH map 在扁平层次结构中列出存储群集中所有节点。但是，为获得最佳结果，在 CRUSH map 中创建一个逻辑层次结构。层次结构指定每个节点的域以及存储集群中这些域之间的关系，包括故障域。在层次结构中定义每个域的故障域可提高存储集群的可靠性。

当计划包含多个数据中心的存储集群时，将节点放置在 CRUSH map 层次结构中，以便在一个数据中心停机时，存储集群将保持启动并运行。

在域中设计故障节点

如果您计划在存储集群中使用三路复制数据，请考虑故障域中节点的位置。如果在数据中心内发生中断，某些数据可能只位于一个副本中。当发生这种情况时，有两个选项：

- 将数据保留为只读状态，并将数据保留为标准设置。
- 在停机期间只有一个副本。

使用标准设置，由于数据在节点间的数据放置的随机性，不是所有数据都会受到影响，一些数据只能有一个副本，而存储集群将恢复到只读模式。但是，如果一些数据只存在于一个副本中，则存储集群会恢复到只读模式。

14.2. 处理数据中心故障

Red Hat Ceph Storage 可能会对基础架构造成灾难性故障，例如在扩展集群中丢失一个数据中心。对于标准对象存储用例，可通过之间设置来独立配置所有三个数据中心。在这种情况下，每个数据中心的存储集群配置可能会有所不同，反映本地功能和依赖项。

应考虑放置层次结构的逻辑结构。可以使用适当的 CRUSH map，反映基础架构中故障域的层次结构。使用逻辑分级定义可提高存储集群的可靠性，而不是使用标准分级定义。故障域在 CRUSH 映射中定义。默认 CRUSH map 包含扁平层次结构中的所有节点。在三个数据中心环境中，如扩展集群，节点放置应以一

个数据中心停机的方式进行管理，但存储集群可以保持启动并运行。在为数据使用三向复制时，请考虑节点位于哪个故障域中。

在以下示例中，生成的 map 源自存储集群的初始设置，包含 6 个 OSD 节点。在本例中，所有节点都只有一个磁盘，因此有一个 OSD。所有节点在默认 root 下排列，这是层次结构树的标准 root。由于分配给两个 OSD 的权重，这些 OSD 接收比其他 OSD 更少的数据区块。这些节点比初始 OSD 磁盘大于初始 OSD 磁盘而稍后引入。这不会影响到一组节点失败的数据放置。

示例

```
[ceph: root@host01 /]# ceph osd tree
ID WEIGHT  TYPE NAME          UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1 0.33554 root default
-2 0.04779     host host03
 0 0.04779         osd.0          up  1.00000          1.00000
-3 0.04779     host host02
 1 0.04779         osd.1          up  1.00000          1.00000
-4 0.04779     host host01
 2 0.04779         osd.2          up  1.00000          1.00000
-5 0.04779     host host04
 3 0.04779         osd.3          up  1.00000          1.00000
-6 0.07219     host host06
 4 0.07219         osd.4          up  0.79999          1.00000
-7 0.07219     host host05
 5 0.07219         osd.5          up  0.79999          1.00000
```

使用逻辑分层定义将节点分组到同一数据中心可以达到数据放置成熟度。可能的定义类型 root, datacenter, rack, row 和 host 可以反映出三个数据中心扩展集群的故障域：

- 节点 host01 和 host02 位于数据中心 1(DC1)
- 节点 host03 和 host05 位于数据中心 2(DC2)
- 节点 host04 和 host06 位于数据中心 3(DC3)
- 所有数据中心都属于相同的结构(allDC)

由于主机中的所有 OSD 都属于主机定义，因此不需要更改。所有其他分配可在存储集群的运行时调整：

- 使用以下命令定义 bucket 结构：

```
ceph osd crush add-bucket allDC root
ceph osd crush add-bucket DC1 datacenter
ceph osd crush add-bucket DC2 datacenter
ceph osd crush add-bucket DC3 datacenter
```

- 通过修改 CRUSH map，将节点移到此结构中的相应位置：

```
ceph osd crush move DC1 root=allDC
ceph osd crush move DC2 root=allDC
ceph osd crush move DC3 root=allDC
ceph osd crush move host01 datacenter=DC1
ceph osd crush move host02 datacenter=DC1
ceph osd crush move host03 datacenter=DC2
```

```
ceph osd crush move host05 datacenter=DC2
ceph osd crush move host04 datacenter=DC3
ceph osd crush move host06 datacenter=DC3
```

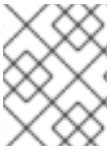
在这种结构中，也可以添加任何新主机以及新磁盘。将 OSD 放置到层次结构中的正确位置，即 CRUSH 算法将冗余部分放入结构中的不同故障域中。

以上示例会产生以下内容：

示例

```
[ceph: root@host01 /]# ceph osd tree
ID  WEIGHT  TYPE NAME             UP/DOWN REWEIGHT PRIMARY-AFFINITY
 -8 6.00000 root allDC
 -9 2.00000     datacenter DC1
 -4 1.00000         host host01
  2 1.00000             osd.2         up  1.00000          1.00000
 -3 1.00000         host host02
  1 1.00000             osd.1         up  1.00000          1.00000
-10 2.00000     datacenter DC2
 -2 1.00000         host host03
  0 1.00000             osd.0         up  1.00000          1.00000
 -7 1.00000         host host05
  5 1.00000             osd.5         up  0.79999          1.00000
-11 2.00000     datacenter DC3
 -6 1.00000         host host06
  4 1.00000             osd.4         up  0.79999          1.00000
 -5 1.00000         host host04
  3 1.00000             osd.3         up  1.00000          1.00000
 -1       0 root default
```

以上列表通过显示 osd 树来显示生成的 CRUSH map。便于查看现在，主机属于数据中心和所有数据中心如何属于相同的顶级结构，但清晰区分位置。



注意

根据映射将数据放在正确的位置，只在健康的集群中正常工作。当某些 OSD 不可用时，misplacement 可能会发生。这些错误替换会在可能这样做后自动更正。

其它资源

- 如需更多信息，请参阅 Red Hat Ceph Storage Storage 策略指南中的 CRUSH 管理一章。