管理指南

SUSE Enterprise Storage 5

管理指南

SUSE Enterprise Storage 5

作者：Tomáš Bažant、Jana Haláčková、Sven Seeberg

出版日期：2019-09-02

SUSE LLC
10 Canal Park Drive
Suite 200
Cambridge MA 02141
USA
https://www.suse.com/documentation

目录

关于本指南

SUSE Enterprise Storage 是 SUSE Linux Enterprise 的扩展。它融合了 Ceph (http://ceph.com/) 存储项目的功能与 SUSE 的企业工程和支持。SUSE Enterprise Storage 为 IT 组织提供了部署分布式存储体系结构的能力，该体系结构可支持使用市售硬件平台的许多用例。

本指南可帮助您了解 SUSE Enterprise Storage 的概念，并重点介绍如何管理 Ceph 基础架构。它还说明了如何将 Ceph 与其他相关解决方案（例如 OpenStack 或 KVM）搭配使用。

本手册中的许多章节中都包含指向附加文档资源的链接。其中包括系统上提供的附加文档以及因特网上提供的文档。

有关该产品可用文档的概述和最新文档更新，请参见 http://www.suse.com/documentation。

1 可用文档

针对本产品提供的手册如下：

管理指南

该指南说明了安装后通常需要执行的各项管理任务。该指南还介绍了将 Ceph 与 `libvirt`、Xen 或 KVM 等虚拟化解决方案集成的步骤，以及通过 iSCSI 和 RADOS 网关访问集群中存储的对象的方法。

《部署指南》

引导您完成 Ceph 集群及 Ceph 所有相关服务的安装步骤。该指南还阐述了基本 Ceph 集群结构，并提供了相关的术语。

在已安装系统的 `/usr/share/doc/manual` 下可以找到产品手册的 HTML 版本。在 http://www.suse.com/documentation 上可以找到最新的文档更新，并可从中下载多种格式的产品文档。

2 反馈

提供了多种反馈渠道：

错误和增强请求

有关产品可用的服务和支持选项，请参见 http://www.suse.com/support/ 。

要报告产品组件的错误，请从 http://www.suse.com/support/ 登录 Novell Customer Center，然后选择 My Support（我的支持）› Service Request（服务请求）。

用户意见

我们希望收到您对本手册和本产品中包含的其他文档的意见和建议。请使用联机文档每页底部的"用户注释"功能或转到 http://www.suse.com/documentation/feedback.html 并在此处输入注释。

邮件

如有对本产品文档的反馈，也可以发送邮件至 `doc-team@suse.de` 。请确保反馈中含有文档标题、产品版本和文档发布日期。要报告错误或给出增强建议，请提供问题的简要说明并指出相应章节编号和页码（或 URL）。

3 文档约定

以下是本手册中使用的版式约定：

- `/etc/passwd`：目录名称和文件名
- `placeholder`：将 `placeholder` 替换为实际值
- `PATH`：环境变量 PATH
- `ls`、`--help`：命令、选项和参数
- `user`：用户和组
- Alt、Alt-F1：按键或组合键；这些键以大写形式显示，如在键盘上一样
- 文件、文件 › 另存为：菜单项，按钮
- 跳舞的企鹅（企鹅一章，↑其他手册）：此内容参见自其他手册中的一章。

4 关于本手册的制作

本书用 GeekoDoc（DocBook 的子集，请参见 http://www.docbook.org ）编写。XML 源文件采用 `xmllint` 校验、经过 `xsltproc` 处理，并使用 Norman Walsh 的样式表的自定义版本转换为 XSL-FO。最终的 PDF 可通过 Apache 开发的 FOP 或 RenderX 开发的 XEP 编排格式。包 `daps` 中提供了用于制作此手册的创作和发布工具。DocBook Authoring and Publishing Suite (DAPS) 以开源软件的形式开发。有关详细信息，请参见 http://daps.sf.net/ 。

5 Ceph Contributors

The Ceph project and its documentation is a result of hundreds of contributors and organizations. See https://ceph.com/contributors/ for more details.

I 集群管理

1 Salt 集群管理

部署 Ceph 集群后，有些时候可能还需要对它执行若干修改。这些修改包括添加或删除新的节点、磁盘或服务。本章介绍该如何完成这些管理任务。

1.1 添加新的集群节点

为集群添加新节点的过程与《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”中所述的初始集群节点部署过程几乎完全相同。

1. 在新节点上安装 SUSE Linux Enterprise Server 12 SP3、配置其网络设置使它能够正确解析 Salt Master 主机名，并安装 `salt-minion` 包：

   ```
   root@minion > zypper in salt-minion
   ```

   如果 Salt Master 的主机名不是 `salt`，请编辑 `/etc/salt/minion` 并添加下面一行：

   ```
   master: DNS_name_of_your_salt_master
   ```

   如果您对上面提到的配置文件进行了任何更改，请重启动 `salt.minion` 服务：

   ```
   root@minion > systemctl restart salt-minion.service
   ```

2. 在 Salt Master 上接受所有 Salt 密钥：

   ```
   root@master # salt-key --accept-all
   ```

3. 校验 `/srv/pillar/ceph/deepsea_minions.sls` 是否也以新的 Salt Minion 为目标。有关更多详细信息，请参见《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”, 运行部署阶段的《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.2.2.1 节 “匹配 Minion 名称”。

4. 运行准备阶段。该阶段会同步模块和 Grains 数据，以便新的 Minion 可以提供 DeepSea 需要的所有信息：

   ```
   root@master # salt-run state.orch ceph.stage.0
   ```

5. 运行发现阶段。该阶段将在 `/srv/pillar/ceph/proposals` 目录中写入新的文件项，您可在其中编辑相关的 .yml 文件：

```
root@master # salt-run state.orch ceph.stage.1
```

6. （可选）如果新添加的主机与现有命名模式不匹配，请更改 `/srv/pillar/ceph/proposals/policy.cfg`。有关详细信息，请参见《部署指南》，第 4 章 “使用 DeepSea/Salt 部署”，第 4.5.1 节 “`policy.cfg` 文件”。

7. 运行配置阶段。该阶段会读取 `/srv/pillar/ceph` 下的所有内容，并相应地更新 Pillar：

```
root@master # salt-run state.orch ceph.stage.2
```

Pillar 用于存储可以使用以下命令访问的数据：

```
root@master # salt target pillar.items
```

8. 配置和部署阶段包含新添加的节点：

```
root@master # salt-run state.orch ceph.stage.3
root@master # salt-run state.orch ceph.stage.4
```

1.2 为节点添加新的角色

您可通过 DeepSea 部署所有受支持的角色类型。有关受支持角色类型的更多信息以及匹配示例，请参见《部署指南》，第 4 章 “使用 DeepSea/Salt 部署”，第 4.5.1.2 节 “角色指定”。

提示：必需的与可选的角色和阶段

一般而言，在将新角色添加到集群节点时，建议您运行全部部署阶段 0 到 5。为了节省一些时间，可根据要部署的角色类型，跳过阶段 3 或 4。OSD 和 MON 角色包含核心服务，是 Ceph 的必要组成部分，而其他角色（例如对象网关）则是可选项。DeepSea 部署阶段是分层的：阶段 3 部署核心服务，而阶段 4 则部署可选服务。

因此，部署核心角色（例如在现有 OSD 节点上部署 MON）时需要运行阶段 3，可以跳过阶段 4。

同样，部署可选服务（例如对象网关）时可以跳过阶段 3，但需要运行阶段 4。

要将新服务添加到现有节点，请执行下列步骤：

1. 修改 `/srv/pillar/ceph/proposals/policy.cfg`，以使现有主机与新角色匹配。有关详细信息，请参见《部署指南》，第 4 章 “使用 DeepSea/Salt 部署”，第 4.5.1 节 “policy.cfg 文件”。例如，如果您需要在 MON 节点上运行对象网关，命令行类似下方所示：

   ```
   role-rgw/xx/x/example.mon-1.sls
   ```

2. 运行阶段 2 以更新 Pillar：

   ```
   root@master # salt-run state.orch ceph.stage.2
   ```

3. 运行阶段 3 以部署核心服务，或者运行阶段 4 以部署可选服务。同时运行这两个阶段也没有问题。

提示

将 OSD 添加到现有集群时请注意，集群将在此后的一段时间内进行重新平衡。为了尽可能缩短重新平衡的时间，建议您同时添加所有要添加的 OSD。

1.3 删除和重新安装集群节点

要从集群中删除角色，请编辑 `/srv/pillar/ceph/proposals/policy.cfg` 并删除相应的行。然后按《部署指南》，第 4 章 “使用 DeepSea/Salt 部署”，第 4.3 节 “集群部署”中所述运行阶段 2 和 5。

注意：从集群中删除 OSD

如果您需要从集群中删除特定 OSD 节点，请确保集群的可用磁盘空间多于要删除的磁盘空间。切记，删除 OSD 会导致整个集群进行重新平衡。

从 Minion 中删除角色时，其目的是撤销与该角色相关的所有更改。对于大部分角色，要实现该任务都很简单，但可能会存在包依赖项问题。如果包被卸装，其依赖项并不会被卸装。

删除的 OSD 会显示为空白驱动器。相关任务除了会擦除分区表外，还会覆盖文件系统的开头并删除备份分区。

注意：保留通过其他方法创建的分区

先前通过其他方法（例如 `ceph-deploy`）配置的磁盘驱动器可能仍然包含分区。DeepSea 不会自动销毁这些分区。管理员必须回收这些驱动器。

例 1.1： 从集群中删除 SALT MINION

举例来说，如果您的存储 Minion 名为“data1.ceph”、“data2.ceph”...“data6.ceph”，则 `policy.cfg` 中的相关行类似下方所示：

```
[...]
# Hardware Profile
profile-default/cluster/data*.sls
profile-default/stack/default/ceph/minions/data*.yml
[...]
```

要删除 Salt Minion“data2.ceph”，请将这些行更改为：

```
[...]
# Hardware Profile
profile-default/cluster/data[1,3-6]*.sls
profile-default/stack/default/ceph/minions/data[1,3-6]*.yml
[...]
```

然后运行阶段 2 和 5：

```
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.5
```

例 1.2： 迁移节点

假设出现下面的情况：在全新安装集群期间，您（管理员）在等待网关硬件就绪时，将其中一个存储节点分配为独立的对象网关。现在，当网关的永久硬件就绪时，您就可以将所需角色最终指定给备用存储节点并删除网关角色。

在针对新硬件运行阶段 0 和 1（请参见《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”, 运行部署阶段）之后，您将新网关命名为 `rgw1`。如果节点 `data8` 需要删除对象网关角色并添加存储角色，且当前的 `policy.cfg` 类似下方所示：

```
# Hardware Profile
profile-default/cluster/data[1-7]*.sls
profile-default/stack/default/ceph/minions/data[1-7]*.sls

# Roles
role-rgw/cluster/data8*.sls
```

则将它更改为：

```
# Hardware Profile
profile-default/cluster/data[1-8]*.sls
profile-default/stack/default/ceph/minions/data[1-8]*.sls

# Roles
role-rgw/cluster/rgw1*.sls
```

运行阶段 2 到 5。阶段 3 会将 `data8` 添加为存储节点。稍候片刻，`data8` 将同时具有两个角色。阶段 4 会将对象网关角色添加到 `rgw1`，而阶段 5 会从 `data8` 中删除对象网关角色。

1.4 重新部署监视器节点

一个或多个监视器节点发生故障且不响应时，您需要将发生故障的监视器从集群中删除，然后再添加回集群（如有需要）。

重要：至少须有三个监视器节点

监视器节点的数量不能少于 3。如果某个监视器节点发生故障，导致您的集群中只有一个或两个监视器节点，您需要临时将监视器角色指定给其他集群节点，然后再重新部署发生故障的监视器节点。重新部署发生故障的监视器节点后，便可以卸装临时监视器角色。

有关向 Ceph 集群添加新节点/角色的详细信息，请参见第 1.1 节 “添加新的集群节点”和第 1.2 节 “为节点添加新的角色”。

有关删除集群节点的详细信息，请参见第 1.3 节 “删除和重新安装集群节点”。

Ceph 节点故障分为以下两种基本程度：

- Salt Minion 主机发生硬件或 OS 级别损坏，无法响应 `salt 'minion_name' test.ping` 调用。在此情况下，您需要按照《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”中的相关说明，对服务器进行彻底的重新部署。
- 监视器相关服务失败并拒绝恢复，但主机会响应 `salt 'minion_name' test.ping` 调用。在此情况下，请执行以下步骤：

1. 编辑 Salt Master 上的 `/srv/pillar/ceph/proposals/policy.cfg`，删除或更新发生故障的监视器节点对应的行，使它们现在指向正常工作的监视器节点。
2. 运行 DeepSea 阶段 2 到 5 以应用这些更改：

```
root@master # deepsea stage run ceph.stage.2
root@master # deepsea stage run ceph.stage.3
root@master # deepsea stage run ceph.stage.4
root@master # deepsea stage run ceph.stage.5
```

1.5 为节点添加 OSD

要向现有 OSD 节点添加磁盘，请校验是否已删除并擦除磁盘上的所有分区。有关详细信息，请参见《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”中的步骤 13。磁盘变为空磁盘后，将磁盘添加到节点的 YAML 文件。该文件的路径是 `/srv/pillar/ceph/proposals/profile-default/stack/default/ceph/minions/node_name.yml`。保存文件后，运行 DeepSea 阶段 2 和 3：

```
root@master # deepsea stage run ceph.stage.2
root@master # deepsea stage run ceph.stage.3
```

提示：自动更新的配置

您无需手动编辑 YAML 文件，DeepSea 可创建新的配置。要让 DeepSea 创建新的配置，需要移走现有的配置：

```
root@master # old /srv/pillar/ceph/proposals/profile-default/
root@master # deepsea stage run ceph.stage.1
root@master # deepsea stage run ceph.stage.2
root@master # deepsea stage run ceph.stage.3
```

1.6 删除 OSD

您可以通过运行以下命令从集群中删除 Ceph OSD：

```
root@master # salt-run disengage.safety
root@master # salt-run remove.osd OSD_ID
```

`OSD_ID` 需为 OSD 的编号，不含 `osd` 一词。例如，对于 `osd.3`，仅使用数字 `3`。

提示：删除多个 OSD

使用 `salt-run remove.osd` 命令无法同时删除多个 OSD。要自动删除多个 OSD，您可以使用以下循环（5、21、33、19 是要删除的 OSD 的 ID 号）：

```
for i in 5 21 33 19
do
 echo $i
 salt-run disengage.safety
 salt-run remove.osd $i
done
```

1.6.1 强制删除已中止的 OSD

有时会出现无法正常删除 OSD 的情况（请参见第 1.6 节 “删除 OSD”）。例如，如果 OSD 或其快速缓存中止、I/O 操作挂起或 OSD 磁盘无法卸载。在上述情况下，您需要强制删除 OSD：

```
root@master # target osd.remove OSD_ID force=True
```

此命令不仅会删除数据分区，还会删除日记或 WAL/DB 分区。

要识别可能处于孤立状态的日记/WAL/DB 设备，请执行以下步骤：

1. 选择可能存在孤立分区的设备，并将其分区列表保存到文件中：

```
root@minion > ls /dev/sdd?* > /tmp/partitions
```

2. 针对所有 block.wal、block.db 和日记设备运行 `readlink`，并将输出与之前保存的分区列表进行比较：

```
root@minion > readlink -f /var/lib/ceph/osd/ceph-*/
{block.wal,block.db,journal} \
 | sort | comm -23 /tmp/partitions -
```

输出内容为 Ceph 未使用的分区列表。

3. 使用您首选的命令（例如 `fdisk`、`parted` 或 `sgdisk`）删除不属于 Ceph 的孤立分区。

1.7 恢复重新安装的 OSD 节点

如果您的某个 OSD 节点上的操作系统损坏且无法恢复，请执行以下步骤恢复该节点，并在集群数据保持不变的情况下重新部署该节点的 OSD 角色：

1. 在节点上重新安装操作系统。
2. 在 OSD 节点上安装 `salt-minion` 包，删除 Salt Master 上的旧 Salt Minion 密钥，并向 Salt Master 注册新 Salt Minion 的密钥。有关 Salt Minion 部署的详细信息，请参见《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”。
3. 不要运行整个阶段 0，而是运行以下部分：

```
root@master # salt 'osd_node' state.apply ceph.sync
root@master # salt 'osd_node' state.apply ceph.packages.common
root@master # salt 'osd_node' state.apply ceph.mines
root@master # salt 'osd_node' state.apply ceph.updates
```

4. 运行 DeepSea 阶段 1 到 5：

```
root@master # salt-run state.orch ceph.stage.1
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.3
root@master # salt-run state.orch ceph.stage.4
root@master # salt-run state.orch ceph.stage.5
```

5. 运行 DeepSea 阶段 0：

```
root@master # salt-run state.orch ceph.stage.0
```

6. 重引导相关的 OSD 节点。系统将重新发现并重新使用所有 OSD 磁盘。

1.8 通过 Salt 实现自动安装

通过使用 Salt 反应器可让安装自动进行。对于虚拟环境或一致的硬件环境，此配置将允许创建具有指定行为的 Ceph 集群。

警告

Salt 无法根据反应器事件执行依赖项检查。存在可能使 Salt Master 过载而无法响应的风险。

自动安装需要：

- 正确创建的 `/srv/pillar/ceph/proposals/policy.cfg`。
- 准备好并已放入 `/srv/pillar/ceph/stack` 目录中的自定义配置。

默认反应器配置只会运行阶段 0 和 1。如此不必等待后续阶段完成即可测试反应器。

第一个 salt-minion 启动时，阶段 0 即会开始。使用锁定可阻止多个实例。所有 Minion 都完成阶段 0 后，阶段 1 将会开始。

如果该操作正确执行，请将 `/etc/salt/master.d/reactor.conf` 中的最后一行：

```
- /srv/salt/ceph/reactor/discovery.sls
```

更改为

```
- /srv/salt/ceph/reactor/all_stages.sls
```

1.9 更新集群节点

最好定期对您的集群节点应用滚动更新。要应用更新，请运行阶段 0：

```
root@master # salt-run state.orch ceph.stage.0
```

如果 DeepSea 检测到正在运行的 Ceph 集群，它会按顺序应用更新并重启动节点。DeepSea 会遵照 Ceph 的官方建议，先更新监视器，然后更新 OSD，最后更新其他服务，例如 MDS、对象网关、iSCSI 网关或 NFS Ganesha。如果 DeepSea 检测到集群中存在问题，会停止更新过程。触发问题的因素可能是：

- Ceph 报告“HEALTH_ERR”的持续时长超过 300 秒。
- 查询 Salt Minion，以了解所指定的服务在更新后是否仍然启动且正在运行。如果服务处于关闭状态超过 900 秒，更新即会失败。

如此安排可确保即使更新损坏或失败，Ceph 集群仍可以运作。

DeepSea 阶段 0 通过 `zypper update` 更新系统，并重引导系统（如果更新了内核）。如果您想避免发生将所有可能的节点强制重引导的情况，请在启动 DeepSea 阶段 0 之前，确保最新的内核已安装且正在运行。

提示：`zypper patch`

如果您更想使用 `zypper patch` 命令来更新系统，请编辑 `/srv/pillar/ceph/stack/global.yml`，添加下面一行：

```
update_method_init: zypper-patch
```

您可以通过将下面几行添加到 `/srv/pillar/ceph/stack/global.yml`，更改 DeepSea 阶段 0 的默认重引导行为：

```
stage_prep_master: default-update-no-reboot
stage_prep_minion: default-update-no-reboot
```

`stage_prep_master` 用于设置 Salt Master 的阶段 0 行为，`stage_prep_minion` 用于设置所有 Minion 的行为。所有可用的参数如下：

default
: 安装更新并在更新之后重引导。

default-update-no-reboot
: 安装更新而不重引导。

default-no-update-reboot
: 重引导而不安装更新。

default-no-update-no-reboot
: 不安装更新，也不重引导。

1.10 停止或重引导集群

在某些情况下，可能需要停止或重引导整个集群。建议您仔细检查运行中服务的依赖项。下列步骤概要说明如何停止和启动集群：

1. 告知 Ceph 集群不要将 OSD 标记为 out：

```
root # ceph osd set noout
```

2. 按下面的顺序停止守护进程和节点：
    1. 存储客户端
    2. 网关，例如 NFS Ganesha 或对象网关
    3. 元数据服务器
    4. Ceph OSD
    5. Ceph Manager
    6. Ceph Monitor
3. 根据需要执行维护任务。

4. 以与关闭过程相反的顺序启动节点和服务器：

    1. Ceph Monitor
    2. Ceph Manager
    3. Ceph OSD
    4. 元数据服务器
    5. 网关，例如 NFS Ganesha 或对象网关
    6. 存储客户端

5. 删除 noout 标志：

```
root # ceph osd unset noout
```

1.11 自定义 ceph.conf 文件

如果您需要将自定义设置放入 `ceph.conf` 文件中，可通过修改 `/srv/salt/ceph/configuration/files/ceph.conf.d` 目录中的配置文件来实现：

- global.conf
- mon.conf
- mgr.conf
- mds.conf
- osd.conf
- client.conf
- rgw.conf

注意：唯一的 rgw.conf

与 `ceph.conf` 的其他段落相比，对象网关具有很强的灵活性，并且是唯一的。所有其他 Ceph 组件都包含静态标题，例如 `[mon]` 或 `[osd]`。对象网关的标题是唯一的，例如 `[client.rgw.rgw1]`。也就是说，`rgw.conf` 文件需要有标题项。请参见 `/srv/salt/ceph/configuration/files/rgw.conf` 查看示例。

重要：运行阶段 3

对上述配置文件进行自定义更改之后，运行阶段 3 以将这些更改应用到集群节点：

```
root@master # salt-run state.orch ceph.stage.3
```

这些文件通过 `/srv/salt/ceph/configuration/files/ceph.conf.j2` 模板文件加入，与 Ceph 配置文件接受的不同段落相对应。将配置片段放入正确的文件，可让 DeepSea 将其放入正确的段落。您不需要添加任何段落标题。

提示

要将任何配置选项仅应用于守护进程的特定实例，请添加标题，例如 `[osd.1]`。以下配置选项将只应用于 ID 为 1 的 OSD 守护进程。

1.11.1 覆盖默认值

段落中位于后面的语句会覆盖前面的语句。因此，可以按照 `/srv/salt/ceph/configuration/files/ceph.conf.j2` 模板中指定的内容来覆盖默认配置。例如，要关闭 cephx 身份验证，可将下面三行添加到 `/srv/salt/ceph/configuration/files/ceph.conf.d/global.conf` 文件：

```
auth cluster required = none
auth service required = none
auth client required = none
```

1.11.2 包括配置文件

如果您需要应用许多自定义配置，请在自定义配置文件中使用以下 include 语句来让文件管理更轻松。下面是 `osd.conf` 文件的示例：

```
[osd.1]
{% include "ceph/configuration/files/ceph.conf.d/osd1.conf" ignore missing %}
[osd.2]
{% include "ceph/configuration/files/ceph.conf.d/osd2.conf" ignore missing %}
[osd.3]
{% include "ceph/configuration/files/ceph.conf.d/osd3.conf" ignore missing %}
[osd.4]
{% include "ceph/configuration/files/ceph.conf.d/osd4.conf" ignore missing %}
```

在前面的示例中，`osd1.conf`、`osd2.conf`、`osd3.conf` 和 `osd4.conf` 文件包含特定于相关 OSD 的配置选项。

提示：运行时配置

对 Ceph 配置文件所做的更改将在相关 Ceph 守护进程重启动之后生效。有关更改 Ceph 运行时配置的详细信息，请参见第 1.12 节 “运行时 Ceph 配置”。

1.12 运行时 Ceph 配置

第 1.11 节 “自定义 ceph.conf 文件”介绍如何更改 Ceph 配置文件 `ceph.conf`。但是，实际的集群行为并不是由 `ceph.conf` 文件的当前状态决定，而是由正在运行的 Ceph 守护进程的配置（存储在内存中）决定。

要查询单个 Ceph 守护进程以了解特定的配置设置，您可以在运行守护进程的节点上使用 admin socket。例如，以下命令可从名为 `osd.0` 的守护进程获取 `osd_max_write_size` 配置参数的值：

```
root # ceph --admin-daemon /var/run/ceph/ceph-osd.0.asok \
config get osd_max_write_size
{
  "osd_max_write_size": "90"
```

```
}
```

您还可以在运行时更改守护进程的设置。请注意，此更改是暂时的，守护进程下次重启动时，更改将会丢失。例如，以下命令可针对集群中的所有 OSD 将 `osd_max_write_size` 参数更改为“50”：

```
root # ceph tell osd.* injectargs --osd_max_write_size 50
```

警告：`injectargs` 并非百分百可靠

有时，使用 `injectargs` 命令可能无法成功更改集群设置。如果您需要确保启用更改的参数，请在配置文件中进行更改并重启动集群中的所有守护进程。

II 操作集群

2 简介

在本手册的这一部分，您将了解如何启动或停止 Ceph 服务、监视集群的状态、使用和修改 CRUSH 地图或管理存储池。

指南中还包含了高级主题，例如通常如何管理用户和身份验证、如何管理存储池和 RADOS 设备快照、如何设置纠删码池，如何通过快速缓存分层提高集群性能。

3 操作 Ceph 服务

您可以使用 `systemd` 或通过 DeepSea 来操作 Ceph 服务。

3.1 使用 systemd 操作 Ceph 服务

使用 `systemctl` 命令操作所有与 Ceph 相关的服务。操作在您当前登录的节点上进行。您需要具备 `root` 特权才能操作 Ceph 服务。

3.1.1 使用目标启动、停止和重启动服务

为了简化启动、停止和重启动节点上特定类型的所有服务（例如所有 Ceph 服务、所有 MON 或所有 OSD）的操作，Ceph 提供了以下 `systemd` 单元文件：

```
root #  ls /usr/lib/systemd/system/ceph*.target
ceph.target
ceph-osd.target
ceph-mon.target
ceph-mgr.target
ceph-mds.target
ceph-radosgw.target
ceph-rbd-mirror.target
```

要启动/停止/重启动节点上的所有 Ceph 服务，请运行以下命令：

```
root # systemctl stop ceph.target
root # systemctl start ceph.target
root # systemctl restart ceph.target
```

要启动/停止/重启动节点上的所有 OSD，请运行以下命令：

```
root # systemctl stop ceph-osd.target
root # systemctl start ceph-osd.target
root # systemctl restart ceph-osd.target
```

针对其他目标的命令与此类似。

3.1.2 启动、停止和重启动个别服务

可以使用以下参数化 `systemd` 单元文件操作个别服务：

```
ceph-osd@.service
ceph-mon@.service
ceph-mds@.service
ceph-radosgw@.service
ceph-rbd-mirror@.service
```

要使用这些命令，首先需要确定要操作的服务的名称。请参见第 3.1.3 节 “识别个别服务”了解有关如何识别服务的更多信息。

要启动/停止/重启动 `osd.1` 服务，请运行以下命令：

```
root # systemctl stop ceph-osd@1.service
root # systemctl start ceph-osd@1.service
root # systemctl restart ceph-osd@1.service
```

针对其他服务类型的命令与此类似。

3.1.3 识别个别服务

通过运行 `systemctl` 并使用 `grep` 命令过滤结果，可以确定特定类型服务的名称/编号。例如：

```
root # systemctl | grep -i 'ceph-osd.*service'
root # systemctl | grep -i 'ceph-mon.*service'
[...]
```

3.1.4 服务状态

可以查询 `systemd` 来了解服务的状态。例如：

```
root # systemctl status ceph-osd@1.service
root # systemctl status ceph-mon@HOSTNAME.service
```

请将 HOSTNAME 替换为运行守护进程的主机名。

如果您不知道服务的确切名称/编号，请参见第 3.1.3 节 “识别个别服务”。

3.2 使用 DeepSea 重启动 Ceph 服务

将更新应用到集群节点之后，需要重启动所指定的服务，以使用最近安装的版本。

注意：检查重启动

重启动集群的过程可能需要一段时间。可通过运行以下命令来使用 Salt 事件总线检查事件：

```
root@master # salt-run state.event pretty=True
```

3.2.1 重启动所有服务

要重启动集群上的所有服务，请运行以下命令：

```
root@master # salt-run state.orch ceph.restart
```

各角色重启动的顺序不同，具体视 DeepSea 版本 (`rpm -q deepsea`) 而定：

- 如果 DeepSea 版本低于 0.8.4，则元数据服务器、iSCSI 网关、对象网关和 NFS Ganesha 服务将并行重启动。
- 如果 DeepSea 为 0.8.4 或更高版本，您已配置的所有角色将按以下顺序重启动：Ceph Monitor、Ceph Manager、Ceph OSD、元数据服务器、对象网关、iSCSI 网关、NFS Ganesha。为了保持较短的停机时间并尽早发现潜在问题，请按顺序重启动各节点。例如，一次只重启动一个监视节点。

如果集群处于降级、不良的状态，该命令会等待集群恢复。

3.2.2 重启动特定服务

要重启动集群上的特定服务，请运行以下命令：

```
root@master # salt-run state.orch ceph.restart.service_name
```

例如，要重启动所有对象网关，请运行以下命令：

```
root@master # salt-run state.orch ceph.restart.rgw
```

您可以使用以下目标：

```
root@master # salt-run state.orch ceph.restart.mon
```

```
root@master # salt-run state.orch ceph.restart.mgr
```

```
root@master # salt-run state.orch ceph.restart.osd
```

```
root@master # salt-run state.orch ceph.restart.mds
```

```
root@master # salt-run state.orch ceph.restart.rgw
```

```
root@master # salt-run state.orch ceph.restart.igw
```

```
root@master # salt-run state.orch ceph.restart.ganesha
```

4 确定集群状态

当集群正在运行时，您可以使用 `ceph` 工具来监视集群。确定集群状态通常涉及检查 OSD、监视器、归置组和元数据服务器的状态。

提示：交互方式

要以交互模式运行 `ceph` 工具，请不带任何自变量在命令行中键入 `ceph`。如果要在一行中输入多条 `ceph` 命令，则使用交互模式较为方便。例如：

```
cephadm > ceph
ceph> health
ceph> status
ceph> quorum_status
ceph> mon_status
```

4.1 检查集群运行状况

在启动集群后到开始读取和/或写入数据期间，检查集群的运行状况：

```
root # ceph health
HEALTH_WARN 10 pgs degraded; 100 pgs stuck unclean; 1 mons down, quorum 0,2 \
node-1,node-2,node-3
```

Ceph 集群会返回下列运行状况代码之一：

OSD_DOWN
: 一个或多个 OSD 标记为已停机。OSD 守护进程可能已停止，或对等 OSD 可能无法通过网络连接 OSD。常见原因包括守护进程已停止或已崩溃、主机已停机或网络中断。
校验主机是否运行良好，守护进程是否已启动，并且网络是否正常工作。如果守护进程已崩溃，守护进程日志文件（`/var/log/ceph/ceph-osd.*`）可能会包含调试信息。

OSD_`crush type`_DOWN，例如 OSD_HOST_DOWN
: 特定 CRUSH 子树中的所有 OSD 均标记为已停机，例如主机上的所有 OSD。

OSD_ORPHAN

在 CRUSH 地图层次结构中引用了 OSD，但它不存在。可使用以下命令从 CRUSH 层次结构中删除 OSD：

```
root # ceph osd crush rm osd.ID
```

OSD_OUT_OF_ORDER_FULL

backfillfull、nearfull、full 和/或 failsafe_full 的用量阈值没有采用升序。特别是，我们需要 backfillfull < nearfull，nearfull < full 且 full < failsafe_full。可使用以下命令调整阈值：

```
root # ceph osd set-backfillfull-ratio ratio
root # ceph osd set-nearfull-ratio ratio
root # ceph osd set-full-ratio ratio
```

OSD_FULL

一个或多个 OSD 超出了 full 阈值，阻止集群处理写入操作。可使用以下命令检查各存储池的用量：

```
root # ceph df
```

可使用以下命令查看当前定义的 full 比例：

```
root # ceph osd dump | grep full_ratio
```

恢复写入可用性的临时解决方法是稍稍提高 full 阈值：

```
root # ceph osd set-full-ratio ratio
```

请通过部署更多 OSD 将新的存储添加到集群，或者删除现有数据来腾出空间。

OSD_BACKFILLFULL

一个或多个 OSD 超出了 backfillfull 阈值，因而不允许将数据重新平衡到此设备。这是一条预警，意味着重新平衡可能无法完成，并且集群将满。可使用以下命令检查各存储池的用量：

```
root # ceph df
```

OSD_NEARFULL

一个或多个 OSD 超出了 nearfull 阈值。这是一条预警，意味着集群将满。可使用以下命令检查各存储池的用量：

```
root # ceph df
```

OSDMAP_FLAGS

已设置一个或多个所需的集群标志。可使用以下命令设置或清除这些标志（full 除外）：

```
root # ceph osd set flag
root # ceph osd unset flag
```

这些标志包括：

full

集群标记为已满，无法处理写入操作。

pauserd、pausewr

已暂停读取或写入。

noup

不允许 OSD 启动。

nodown

将会忽略 OSD 故障报告，如此监视器便不会将 OSD 标记为 down。

noin

先前标记为 out 的 OSD 在启动时将不会重新标记为 in。

noout

停机的 OSD 在配置间隔过后将不会自动标记为 out。

nobackfill、norecover、norebalance

恢复或数据重新平衡进程已暂停。

noscrub、nodeep_scrub

整理 (scrub) 进程已禁用（请参见第 6.5 节 “整理 (Scrub)”）。

notieragent

快速缓存分层活动已暂停。

OSD_FLAGS

一个或多个 OSD 设置了所需的每 OSD 标志。这些标志包括：

noup

不允许 OSD 启动。

nodown

将会忽略此 OSD 的故障报告。

noin

如果此 OSD 先前在发生故障后自动标记为 out，当它启动时将不会标记为 in。

noout

如果此 OSD 已停机，则在配置的间隔过后，它将不会自动标记为 out。

可使用以下命令来设置和清除每 OSD 标志：

```
root # ceph osd add-flag osd-ID
root # ceph osd rm-flag osd-ID
```

OLD_CRUSH_TUNABLES

CRUSH 地图目前使用的设置很旧，应予以更新。`mon_crush_min_required_version` 配置选项可确定使用时不会触发此运行状况警告的最旧可调变量（即能够连接到集群的最旧客户端版本）。

OLD_CRUSH_STRAW_CALC_VERSION

CRUSH 地图目前使用较旧的非最佳方法来计算 straw 桶的中间权重值。应该更新 CRUSH 地图以使用较新的方法（`straw_calc_version`=1）。

CACHE_POOL_NO_HIT_SET

一个或多个快速缓存池未配置命中集来跟踪用量，这使分层代理无法识别要从快速缓存中清理和逐出的冷对象。可使用以下命令对快速缓存池配置命中集：

```
root # ceph osd pool set poolname hit_set_type type
root # ceph osd pool set poolname hit_set_period period-in-seconds
root # ceph osd pool set poolname hit_set_count number-of-hitsets
root # ceph osd pool set poolname hit_set_fpp target-false-positive-rate
```

OSD_NO_SORTBITWISE

未在运行早于 Luminous 12 版本的 OSD，但是尚未设置 `sortbitwise` 标志。您需要先设置 `sortbitwise` 标志，Luminous 12 或更新版本的 OSD 才能启动：

```
root # ceph osd set sortbitwise
```

POOL_FULL

一个或多个存储池已达到其配额，不再允许写入。可使用以下命令设置存储池配额和用量：

```
root # ceph df detail
```

您可以使用以下命令提高存储池配额

```
root # ceph osd pool set-quota poolname max_objects num-objects
root # ceph osd pool set-quota poolname max_bytes num-bytes
```

或者删除一些现有数据以减少用量。

PG_AVAILABILITY

数据可用性下降，这意味着集群无法处理针对集群中某些数据的潜在读取或写入请求。具体而言，一个或多个 PG 处于不允许处理 IO 请求的状态。有问题的 PG 状态包括连接建立中、不新鲜、不完整和不活跃（如果这些状况不迅速解决）。运行以下命令可获得有关哪些 PG 受影响的详细信息：

```
root # ceph health detail
```

大多数情况下，出现此情形的根本原因在于一个或多个 OSD 当前已停机。可使用以下命令查询特定的有问题 PG 的状态：

```
root # ceph tell pgid query
```

PG_DEGRADED

某些数据的数据冗余降低，这意味着集群没有所需数量的副本用于所有数据（对于副本池）或纠删码分段（对于纠删码池）。具体而言，一个或多个 PG 设置了 degraded 或 undersized 标志（集群中没有该归置组的足够实例），或者有一段时间未设置 clean 标志。运行以下命令可获得有关哪些 PG 受影响的详细信息：

```
root # ceph health detail
```

大多数情况下，出现此情形的根本原因在于一个或多个 OSD 当前已停机。可使用以下命令查询特定的有问题 PG 的状态：

```
root # ceph tell pgid query
```

PG_DEGRADED_FULL

由于集群中的可用空间不足，某些数据的数据冗余可能已降低或面临风险。具体而言，一个或多个 PG 设置了 backfill_toofull 或 recovery_tooful 标志，这意味着集群无法迁移或恢复数据，原因是一个或多个 OSD 高于 backfillfull 阈值。

PG_DAMAGED

数据整理 (scrub)（请参见第 6.5 节 “整理 (Scrub)”）进程发现集群中存在某些数据一致性问题。具体而言，一个或多个 PG 设置了 inconsistent 或 snaptrim_error 标志（表示某个较早的整理操作发现问题），或者设置了 repair 标志（表示当前正在修复此类不一致问题）。

OSD_SCRUB_ERRORS

最近的 OSD 整理 (scrub) 操作发现了不一致问题。

CACHE_POOL_NEAR_FULL

快速缓存层池将满。在此环境中，“满”由快速缓存池的 target_max_bytes 和 target_max_objects 属性确定。池达到目标阈值时，如果正在从快速缓存清理并逐出数据，写入池的请求可能会被阻止，出现常会导致延迟很高且性能变差的状态。可使用以下命令调整快速缓存池目标大小：

```
root # ceph osd pool set cache-pool-name target_max_bytes bytes
root # ceph osd pool set cache-pool-name target_max_objects objects
```

正常的快速缓存清理和逐出活动还可能因基础层可用性或性能下降或者集群的整体负载较高而受到限制。

TOO_FEW_PGS

使用中的 PG 数量低于每个 OSD 的 PG 数的可配置阈值 mon_pg_warn_min_per_osd。这可能导致集群中各 OSD 间的数据分布和平衡未达到最佳，以致降低整体性能。

TOO_MANY_PGS

使用中的 PG 数量高于每个 OSD 的 PG 数的可配置阈值 mon_pg_warn_max_per_osd。这可能导致 OSD 守护进程的内存用量较高，集群状态更改（例如 OSD 重启动、添加或删除）之后建立连接速度降低，并且 Ceph manager 和 Ceph monitor 上的负载较高。虽然不能降低现有存储池的 pg_num 值，但是可以降低 pgp_num 值。这样可有效地在同组 OSD 上并置一些 PG，从而减轻上述的一些负面影响。可使用以下命令调整 pgp_num 值：

```
root # ceph osd pool set pool pgp_num value
```

SMALLER_PGP_NUM

一个或多个存储池的 pgp_num 值小于 pg_num。这通常表示 PG 计数有所提高，但未同时提升归置行为。使用以下命令设置 pgp_num，使其与触发数据迁移的 pg_num 相匹配，通常便可解决此问题：

```
ceph osd pool set pool pgp_num pg_num_value
```

MANY_OBJECTS_PER_PG

一个或多个存储池的每 PG 平均对象数大大高于集群的整体平均值。该特定阈值通过 mon_pg_warn_max_object_skew 配置值控制。这通常表示包含集群中大部分数据的存储池具有的 PG 太少，以及/或者不包含这么多数据的其他存储池具有的 PG 太多。可通过调整监视器上的 mon_pg_warn_max_object_skew 配置选项提高阈值，来消除该运行状况警告。

POOL_APP_NOT_ENABLED

存在包含一个或多个对象但尚未标记为供特定应用使用的存储池。将存储池标记为供某个应用使用即可消除此警告。例如，如果存储池由 RBD 使用：

```
root # rbd pool init pool_name
```

如果存储池正由自定义应用“foo”使用，您还可以使用低级别命令标记它：

```
root # ceph osd pool application enable foo
```

POOL_FULL

一个或多个存储池已达到（或几乎要达到）其配额。触发此错误状况的阈值通过 mon_pool_quota_crit_threshold 配置选项控制。可使用以下命令上调、下调（或删除）存储池配额：

```
root # ceph osd pool set-quota pool max_bytes bytes
root # ceph osd pool set-quota pool max_objects objects
```

将配额值设置为 0 将禁用配额。

POOL_NEAR_FULL

一个或多个存储池接近其配额。触发此警告状况的阈值通过 mon_pool_quota_warn_threshold 配置选项控制。可使用以下命令上调、下调（或删除）存储池配额：

```
root # ceph osd osd pool set-quota pool max_bytes bytes
root # ceph osd osd pool set-quota pool max_objects objects
```

将配额值设置为 0 将禁用配额。

OBJECT_MISPLACED

集群中的一个或多个对象未存储在集群希望用于存储这些对象的节点上。这表示集群最近的某项更改导致的数据迁移尚未完成。误放的数据本质上不属于危险状况。数据一致性方面永远不会有风险，仅当所需位置放置了对象所需份数的新副本之后，系统才会删除对象的旧副本。

OBJECT_UNFOUND

找不到集群中的一个或多个对象。具体而言，OSD 知道对象的新副本或已更新副本应该存在，但当前在线的 OSD 上找不到该版本的对象副本。系统将阻止对“未找到”对象的读取和写入请求。理想情况下，系统可将具有未找到对象的最近副本的已停机 OSD 恢复在线状态。可通过负责处理未找到对象的 PG 的互联状态识别候选 OSD：

```
root # ceph tell pgid query
```

REQUEST_SLOW

正花费很长的时间处理一个或多个 OSD 请求。这可能表示负载极重、存储设备速度缓慢或有软件错误。可以从 OSD 主机执行以下命令来查询有问题的 OSD 上的请求队列：

```
root # ceph daemon osd.id ops
```

可以查看近期最慢的请求摘要：

```
root # ceph daemon osd.id dump_historic_ops
```

可使用以下命令查找 OSD 的位置：

```
root # ceph osd find osd.id
```

REQUEST_STUCK

已将一个或多个 OSD 请求阻止了很长时间。这表示集群很长一段时间运行状况不佳（例如没有足够的运行中 OSD），或 OSD 存在一些内部问题。

PG_NOT_SCRUBBED

最近未整理 (scrub) (请参见第 6.5 节 “整理 (Scrub)”) 一个或多个 PG。通常每 `mon_scrub_interval` 秒整理 (scrub) 一次 PG，当 `mon_warn_not_scrubbed` 这类间隔已过但未进行整理 (scrub) 时，就会触发此警告。如果 PG 未标记为清理，系统将不会整理 (scrub) 它们。如果 PG 放置错误或已降级，就会出现这种情况 (请参见上文中的 PG_AVAILABILITY 和 PG_DEGRADED) 。您可以使用以下命令手动对标记为清理的 PG 启动整理 (scrub):

```
root # ceph pg scrub pgid
```

PG_NOT_DEEP_SCRUBBED

最近未深层整理 (deep scrub) (请参见第 6.5 节 “整理 (Scrub)”) 一个或多个 PG。通常每 `osd_deep_scrub_interval` 秒整理 (scrub) 一次 PG，当 `mon_warn_not_deep_scrubbed` 这类间隔已过但未进行整理 (scrub) 时，就会触发此警告。如果 PG 未标记为清理，系统将不会 (深层) 整理 (scrub) 它们。如果 PG 放置错误或已降级，就会出现这种情况 (请参见上文中的 PG_AVAILABILITY 和 PG_DEGRADED) 。您可以使用以下命令手动对标记为清理的 PG 启动整理 (scrub):

```
root # ceph pg deep-scrub pgid
```

提示

如果之前为您的配置或密钥环指定了非默认位置，则此时可以指定它们的位置:

```
root # ceph -c /path/to/conf -k /path/to/keyring health
```

4.2 监视集群

可以使用 `ceph -s` 了解集群的即时状态。例如，由一个监视器和两个 OSD 组成的微型 Ceph 集群可在某工作负载正在运行时列显以下内容:

```
cluster:
  id:     6586341d-4565-3755-a4fd-b50f51bee248
  health: HEALTH_OK
```

```
services:
  mon: 3 daemons, quorum blueshark1,blueshark2,blueshark3
  mgr: blueshark3(active), standbys: blueshark2, blueshark1
  osd: 15 osds: 15 up, 15 in

data:
  pools:   8 pools, 340 pgs
  objects: 537 objects, 1985 MB
  usage:   23881 MB used, 5571 GB / 5595 GB avail
  pgs:     340 active+clean

io:
  client:   100 MB/s rd, 26256 op/s rd, 0 op/s wr
```

输出内容提供了以下信息：

- 集群 ID
- 集群运行状况
- 监视器地图版本号和监视器仲裁的状态
- OSD 地图版本号和 OSD 的状态
- 归置组地图版本
- 归置组和存储池数量
- 所存储数据理论上的数量和所存储对象的数量；以及
- 所存储数据的总量。

提示：Ceph 计算数据用量的方式

`used` 值反映实际使用的原始存储量。`xxx GB / xxx GB` 值表示集群的可用容量（两者中较小的数字），以及集群的整体存储容量。理论数量反映在复制、克隆所存储数据或创建其快照前这些数据的大小。因此，实际存储的数据量通常会超出理论上的存储量，因为 Ceph 会创建数据的副本，可能还会将存储容量用于克隆和创建快照。

显示即时状态信息的其他命令如下：

- `ceph pg stat`
- `ceph osd pool stats`
- `ceph df`
- `ceph df detail`

要获得实时更新的信息，请将以上任何命令（包括 `ceph -s`）放置在等待循环中，例如：

```
rootwhile true ; do ceph -s ; sleep 10 ; done
```

如果您看累了，请按 Ctrl-C 。

4.3 检查集群的用量统计数字

要检查集群的数据用量和在各存储池中的数据分布，可以使用 `df` 选项。它类似于 Linux `df`。执行以下命令：

```
root # ceph df
GLOBAL:
    SIZE       AVAIL      RAW USED     %RAW USED
    55886G     55826G       61731M          0.11
POOLS:
    NAME          ID     USED      %USED     MAX AVAIL     OBJECTS
    testpool      1          0         0        17676G           0
    ecpool        2      4077M      0.01        35352G        2102
    test1         3          0         0        17676G           0
    rbd           4         16         0        17676G           3
    rbd1          5         16         0        17676G           3
    ecpool1       6      5708M      0.02        35352G        2871
```

输出内容的 `GLOBAL` 段落提供集群用于数据的存储量概览。

- `SIZE`：集群的整体存储容量。
- `AVAIL`：集群中可以使用的可用空间容量。

- `RAW USED`：已用的原始存储量。
- `% RAW USED`：已用的原始存储量百分比。将此数字与 `full ratio` 和 `near full ratio` 搭配使用，可确保您不会用完集群的容量。有关其他详细信息，请参见存储容量 (http://docs.ceph.com/docs/master/rados/configuration/mon-config-ref#storage-capacit)。

注意：集群填充程度

原始存储填充程度达到 70% - 80%，表示需要向集群添加新的存储。较高的用量可能导致单个 OSD 填满，集群处于不良运行状况。

使用命令 `ceph osd df tree` 可列出所有 OSD 的填充程度。

输出内容的 `POOLS` 段落提供了存储池列表和每个存储池的理论用量。此段落的输出不反映副本、克隆数据或快照。例如，如果您存储含 1MB 数据的对象，理论用量将是 1MB，但是根据副本、克隆数据或快照数量，实际用量可能是 2MB 或更多。

- `NAME`：存储池的名称。
- `ID`：存储池 ID。
- `USED`：以千字节 (KB) 为单位的理论已存储数据量，如果该数字附加了 M，则以兆字节为单位，如果附加了 G，则以千兆字节为单位。
- `%USED`：每个存储池的理论已用存储百分比。
- `MAX AVAIL`：给定存储池中的最大可用空间。
- `OBJECTS`：每个存储池的理论已存储对象数。

注意

POOLS 段落中的数字是理论上的。它们不包括副本、快照或克隆数量。因此，USED 和 %USED 数量之和不会加总到输出内容 %GLOBAL 段落中的 RAW USED 和 %RAW USED 数量中。

4.4 检查集群的状态

要检查集群的状态，请执行以下命令：

```
root # ceph status
```

或者

```
root # ceph -s
```

在交互模式下，键入 `status`，然后按 `Enter`。

```
ceph> status
```

Ceph 将列显集群状态。例如，由一个监视器和两个 OSD 组成的微型 Ceph 集群可能会列显以下内容：

```
cluster b370a29d-9287-4ca3-ab57-3d824f65e339
 health HEALTH_OK
 monmap e1: 1 mons at {ceph1=10.0.0.8:6789/0}, election epoch 2, quorum 0 ceph1
 osdmap e63: 2 osds: 2 up, 2 in
  pgmap v41332: 952 pgs, 20 pools, 17130 MB data, 2199 objects
        115 GB used, 167 GB / 297 GB avail
               1 active+clean+scrubbing+deep
             951 active+clean
```

4.5 检查 OSD 状态

可通过执行以下命令来检查 OSD，以确保它们已启动且正在运行：

```
root # ceph osd stat
```

或者

```
root # ceph osd dump
```

还可以根据 OSD 在 CRUSH 地图中的位置查看 OSD。

```
root # ceph osd tree
```

Ceph 将列显 CRUSH 树及主机、它的 OSD、OSD 是否已启动及其权重。

```
# id    weight  type name       up/down reweight
-1      3       pool default
-3      3               rack mainrack
-2      3                       host osd-host
0       1                               osd.0   up      1
1       1                               osd.1   up      1
2       1                               osd.2   up      1
```

4.6 检查填满的 OSD

Ceph 可阻止您向填满的 OSD 写入数据，以防丢失数据。在正常运行的集群中，当集群接近其填满比例时，您会收到警告。`mon osd full ratio` 默认设为容量的 0.95 (95%)，达到该比例后，集群会阻止客户端写入数据。`mon osd nearfull ratio` 默认设为容量的 0.85 (85%)，达到该比例时，集群会生成运行状况警告。

可通过 `ceph health` 命令报告填满的 OSD 节点：

```
ceph health
  HEALTH_WARN 1 nearfull osds
  osd.2 is near full at 85%
```

或者

```
ceph health
  HEALTH_ERR 1 nearfull osds, 1 full osds
  osd.2 is near full at 85%
  osd.3 is full at 97%
```

处理填满的集群的最佳方法是添加新的 OSD 节点，以让集群将数据重新分布到新的可用存储。

如果 OSD 因填满而无法启动，您可以通过删除已满 OSD 中的一些归置组目录来删除一些数据。

提示：防止 OSD 填满

OSD 变满（即用完 100% 的磁盘空间）之后，往往会迅速崩溃而不发出警告。管理 OSD 节点时需记住下面几点提示。

- 每个 OSD 的磁盘空间（通常挂载在 `/var/lib/ceph/osd/osd-{1,2..}` 下）需放置在专用的底层磁盘或分区上。
- 检查 Ceph 配置文件，确保 Ceph 不会将其日志文件存储在专供 OSD 使用的磁盘/分区上。
- 确保没有其他进程写入专供 OSD 使用的磁盘/分区。

4.7 检查监视器状态

如果集群有多个监视器（这是很有可能的），则应在启动集群之后到读取和/或写入数据之前的期间检查监视器仲裁状态。有多个监视器在运行时，仲裁必须存在。您还应该定期检查监视器状态，确保它们正在运行。

要显示监视器地图，请执行以下命令：

```
root # ceph mon stat
```

或者

```
root # ceph mon dump
```

要检查监视器集群的仲裁状态，请执行以下命令：

```
root # ceph quorum_status
```

Ceph 将返回仲裁状态。例如，由三个监视器组成的 Ceph 集群可能返回以下内容：

```
{ "election_epoch": 10,
  "quorum": [
        0,
        1,
        2],
```

```
  "monmap": { "epoch": 1,
      "fsid": "444b489c-4f16-4b75-83f0-cb8097468898",
      "modified": "2011-12-12 13:28:27.505520",
      "created": "2011-12-12 13:28:27.505520",
      "mons": [
            { "rank": 0,
              "name": "a",
              "addr": "127.0.0.1:6789\/0"},
            { "rank": 1,
              "name": "b",
              "addr": "127.0.0.1:6790\/0"},
            { "rank": 2,
              "name": "c",
              "addr": "127.0.0.1:6791\/0"}
           ]
    }
}
```

4.8 检查归置组状态

归置组会将对象映射到 OSD。监视归置组时，您希望它们处于 `active` 和 `clean` 状态。有关详细的讨论内容，请参见监视 OSD 和归置组 (http://docs.ceph.com/docs/master/rados/operations/monitoring-osd-pg) ↗。

4.9 使用管理套接字

Ceph 管理套接字可让您通过套接字接口查询守护进程。默认情况下，Ceph 套接字驻留在 `/var/run/ceph` 下。要通过管理套接字访问守护进程，请登录运行守护进程的主机，并使用以下命令：

```
root # ceph --admin-daemon /var/run/ceph/socket-name
```

要查看可用的管理套接字命令，请执行以下命令：

```
root # ceph --admin-daemon /var/run/ceph/socket-name help
```

管理套接字命令可让您在运行时显示和设置您的配置。有关详细信息，请参见在运行时查看配置 (http://docs.ceph.com/docs/master/rados/configuration/ceph-conf#ceph-runtime-config)↗。

另外，您也可以直接在运行时设置配置（管理套接字会绕过监视器，这与 `ceph tell daemon-type.id` injectargs 不同，后者依赖于监视器，但不需要您直接登录有问题的主机）。

5 使用 cephx 进行身份验证

为了识别客户端并防范中间人攻击，Ceph 提供了 `cephx` 身份验证系统。在此环境中，客户端表示人类用户（例如 admin 用户）或 Ceph 相关的服务/守护进程（例如 OSD、监视器或对象网关）。

注意

`cephx` 协议不会处理 TLS/SSL 之类的传输中数据加密。

5.1 身份验证体系结构

`cephx` 使用共享机密密钥进行身份验证，这意味着，客户端和监视器集群均有客户端机密密钥的副本。身份验证协议可让双方互相证明各自持有密钥的副本，且无需真正透露密钥。这样就实现了相互身份验证，即，集群可确保用户拥有机密密钥，而用户亦可确保集群持有机密密钥的副本。

Ceph 的一项重要可伸缩性功能就是可免于通过集中式界面与 Ceph 对象存储交互。这意味着，Ceph 客户端可直接与 OSD 交互。为了保护数据，Ceph 提供了 `cephx` 身份验证系统来对 Ceph 客户端进行身份验证。

每个监视器都可对客户端进行身份验证并分发密钥，因此，在使用 `cephx` 时，不会出现单一故障点或瓶颈。监视器会返回身份验证数据结构，其中包含获取 Ceph 服务时要用到的会话密钥。此会话密钥本身已使用客户端的永久机密密钥进行了加密，因此，只有客户端才能向 Ceph monitor 请求服务。然后，客户端使用会话密钥向监视器请求所需的服务，监视器会为客户端提供一个票据，用于向实际处理数据的 OSD 验证客户端身份。Ceph monitor 和 OSD 共享一个机密，因此，客户端使用监视器提供的票据向集群中的任何 OSD 或元数据服务器表明身份。`cephx` 票据有失效时间，因此，攻击者无法使用已失效的票据或以不当方式获取的会话密钥。只要保证客户端机密密钥在失效之前不被泄露，这种身份验证方式就能防止能够访问通讯媒体的攻击者使用另一客户端的身份创建虚假讯息，或改动另一客户端的正当讯息。

要使用 `cephx`，管理员必须先设置客户端/用户。在下图中，`client.admin` 用户从命令行调用 `ceph auth get-or-create-key` 来生成用户名和机密密钥。Ceph 的 `auth` 子系统会生成该用户名和密钥，在监视器中存储一个副本，并将该用户的机密传回给 `client.admin` 用户。这意味着，客户端和监视器共享一个机密密钥。

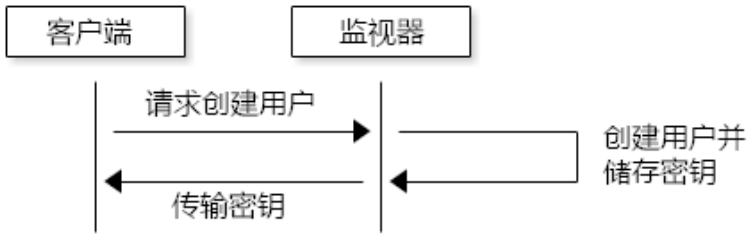


图 5.1： cephx 的基本身份验证

要在监视器中进行身份验证，客户端需将用户名传递给监视器。监视器会生成一个会话密钥，并使用与该用户名关联的机密密钥来加密该会话密钥，然后将加密的票据传回给客户端。之后，客户端会使用共享的机密密钥解密数据，以获取会话密钥。会话密钥可识别当前会话的用户。然后，客户端请求与该用户相关、由会话密钥签名的票据。监视器会生成一个票据，使用用户的机密密钥进行加密，然后将其传回给客户端。客户端解密该票据，并使用它对发往整个集群中的 OSD 和元数据服务器的请求进行签名。

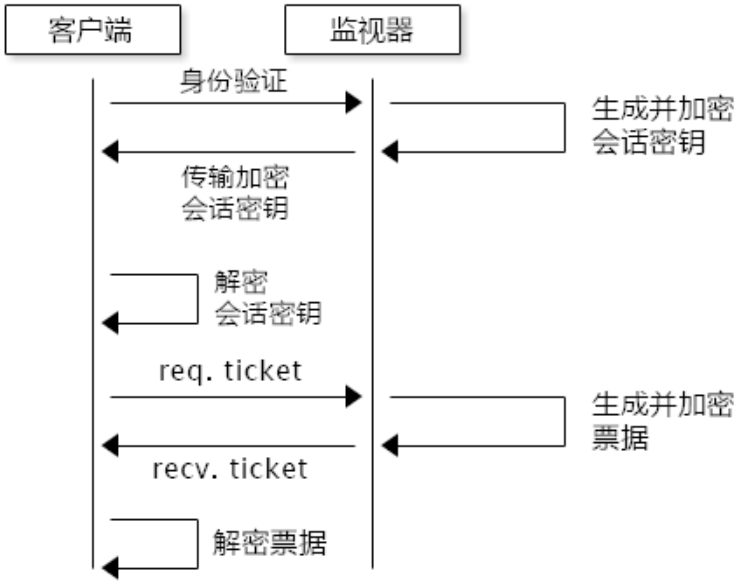


图 5.2： cephx 身份验证

cephx 协议会对客户端计算机与 Ceph 服务器之间进行的通讯进行身份验证。完成初始身份验证后，将使用监视器、OSD 和元数据服务器可通过共享密钥进行校验的票据，来对客户端与服务器之间发送的每条讯息进行签名。

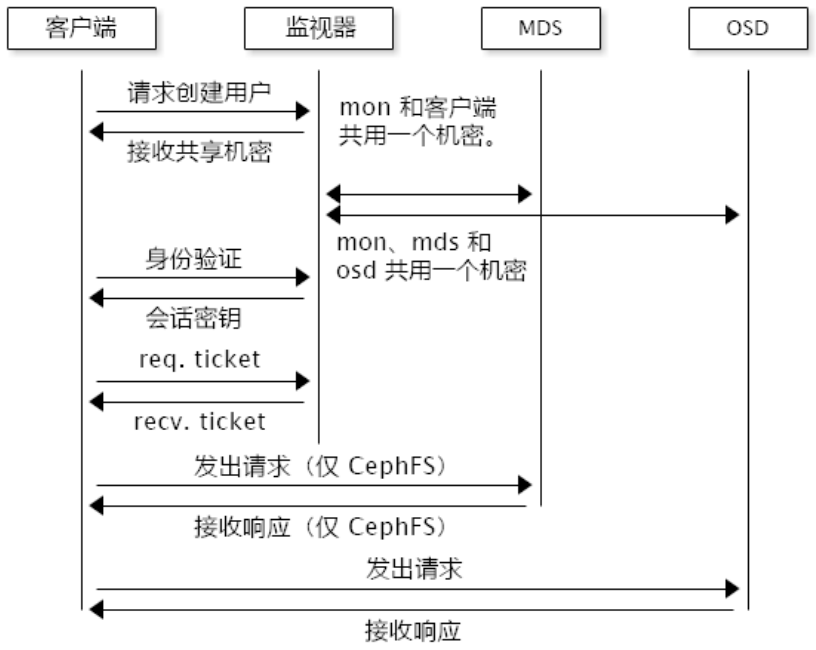


图 5.3： cephx 身份验证 - MDS 和 OSD

重要

这种身份验证提供的保护仅限于 Ceph 客户端与 Ceph 集群主机之间，身份验证不会扩展到 Ceph 客户端以外。如果用户从远程主机访问 Ceph 客户端，则不会对用户主机与客户端主机之间的连接应用 Ceph 身份验证。

5.2 密钥管理

本节介绍 Ceph 客户端用户，以及如何在 Ceph 存储集群中对其进行身份验证和授权。用户是指使用 Ceph 客户端来与 Ceph 存储集群守护进程交互的个人或系统参与者（例如应用）。

Ceph 在启用身份验证和授权（默认启用）的情况下运行时，您必须指定一个用户名，以及包含所指定用户的机密密钥的密钥环（通常通过命令行指定）。如果您未指定用户名，Ceph 将使用 `client.admin` 作为默认用户名。如果您未指定密钥环，Ceph 将通过 Ceph 配置文件中的密钥环设置来查找密钥环。例如，如果您在未指定用户名或密钥环的情况下执行 `ceph health` 命令，Ceph 将按如下所示解释该命令：

```
ceph -n client.admin --keyring=/etc/ceph/ceph.client.admin.keyring health
```

或者，您可以使用 CEPH_ARGS 环境变量来避免重复输入用户名和机密。

5.2.1 背景信息

无论 Ceph 客户端是何类型（例如，块设备、对象存储、文件系统、本机 API），Ceph 都会在存储池中将所有数据存储为对象。Ceph 用户需要拥有存储池访问权限才能读取和写入数据。此外，Ceph 用户必须拥有执行权限才能使用 Ceph 的管理命令。以下概念可帮助您理解 Ceph 用户管理。

5.2.1.1 用户

用户是指个人或系统参与者（例如应用）。通过创建用户，可以控制谁（或哪个参与者）能够访问您的 Ceph 存储集群、其存储池及存储池中的数据。

Ceph 使用多种类型的用户。进行用户管理时，将始终使用 client 类型。Ceph 通过句点 (.) 分隔格式来标识用户，该格式由用户类型和用户 ID 组成。例如，TYPE.ID、client.admin 或 client.user1。区分用户类型的原因在于，Ceph monitor、OSD 和元数据服务器也使用 cephx 协议，但它们并非客户端。区分用户类型有助于将客户端用户与其他用户区分开来，从而简化访问控制、用户监视和可追溯性。

注意

Ceph 存储集群用户与 Ceph 对象存储用户或 Ceph 文件系统用户均不同。Ceph Object Gateway 使用 Ceph 存储集群用户在网关守护进程与存储集群之间通讯，但网关有自己的用户管理功能来管理最终用户。Ceph 文件系统使用 POSIX 语义。与它关联的用户空间与 Ceph 存储集群用户不同。

5.2.1.2 授权和使能

Ceph 使用“使能”(caps) 术语来描述对已经过身份验证的用户授权，允许其运用监视器、OSD 和元数据服务器的功能。使能还可以限制对存储池中的数据或存储池中某个名称空间的访问。Ceph 管理用户可在创建或更新用户时设置用户的使能。

使能语法的格式如下：

```
daemon-type 'allow capability' [...]
```

下面是每个服务类型的使能列表：

监视器使能

包括 `r`、`w`、`x` 和 `allow profile cap`。

```
mon 'allow rwx'
mon 'allow profile osd'
```

OSD 使能

包括 `r`、`w`、`x`、`class-read`、`class-write` 和 `profile osd`。此外，OSD 使能还允许进行存储池和名称空间设置。

```
osd 'allow capability' [pool=poolname] [namespace=namespace-name]
```

MDS 使能

只需要 `allow`，或留空。

```
mds 'allow'
```

以下各项描述了每项使能：

allow

需先于守护进程的访问设置指定。仅对 MDS 表示 `rw`。

r

向用户授予读取权限。访问监视器以检索 CRUSH 地图时需具有此使能。

w

向用户授予针对对象的写入权限。

x

授予用户调用类方法（包括读取和写入）的能力，以及在监视器中执行 `auth` 操作的能力。

class-read

授予用户调用类读取方法的能力。`x` 的子集。

class-write

授予用户调用类写入方法的能力。`x` 的子集。

*

授予用户对特定守护进程/存储池的读取、写入和执行权限，以及执行管理命令的能力。

profile osd

授予用户以某个 OSD 身份连接到其他 OSD 或监视器的权限。授予 OSD 权限，使 OSD 能够处理复制检测信号流量和状态报告。

profile mds

授予用户以某个 MDS 身份连接到其他 MDS 或监视器的权限。

profile bootstrap-osd

授予用户引导 OSD 的权限。授权给部署工具，使其在引导 OSD 时有权添加密钥。

profile bootstrap-mds

授予用户引导元数据服务器的权限。授权给部署工具，使其在引导元数据服务器时有权添加密钥。

5.2.1.3 存储池

存储池是指用户在其中存储数据的逻辑分区。在 Ceph 部署中，常见的做法是为相似类型的数据创建一个存储池作为逻辑分区。例如，将 Ceph 部署为 OpenStack 的后端时，典型的部署方式是为卷、映像、备份和虚拟机以及用户（如 `client.glance` 或 `client.cinder`）创建相应的存储池。

5.2.2 管理用户

用户管理功能可让 Ceph 集群管理员能够直接在 Ceph 集群中创建、更新和删除用户。

在 Ceph 集群中创建或删除用户时，可能需要将密钥分发到客户端，以便将密钥添加到密钥环。有关详细信息，请参见第 5.2.3 节 “密钥环管理”。

5.2.2.1 列出用户

要列出集群中的用户，请执行以下命令：

```
ceph auth list
```

Ceph 将列出您集群中的所有用户。例如，在包含两个节点的集群中，`ceph auth list` 输出类似下方所示：

```
installed auth entries:

osd.0
        key: AQCvCbtToC6MDhAATtuT70Sl+DymPCfDSsyV4w==
        caps: [mon] allow profile osd
        caps: [osd] allow *
osd.1
        key: AQC4CbtTCFJBChAAVq5spj0ff4eHZICxIOVZeA==
        caps: [mon] allow profile osd
        caps: [osd] allow *
client.admin
        key: AQBHCbtT6APDHhAA5W00cBchwkQjh3dkKsyPjw==
        caps: [mds] allow
        caps: [mon] allow *
        caps: [osd] allow *
client.bootstrap-mds
        key: AQBICbtTOK9uGBAAdbe5zcIGHZL3T/u2g6EBww==
        caps: [mon] allow profile bootstrap-mds
client.bootstrap-osd
        key: AQBHCbtT4GxqORAADE5u7RkpCN/oo4e5W0uBtw==
        caps: [mon] allow profile bootstrap-osd
```

注意：TYPE.ID 表示法

请注意，针对用户采用 `TYPE.ID` 表示法，例如，`osd.0` 指定 `osd` 类型的用户，其 ID 为 `0`。`client.admin` 是 `client` 类型的用户，其 ID 为 `admin`。另请注意，每个项包含一个 `key: 值` 项，以及一个或多个 `caps:` 项。

可以结合使用 `-o 文件名` 选项和 `ceph auth list` 将输出保存到某个文件。

5.2.2.2 获取有关用户的信息

要检索特定的用户、密钥和使能，请执行以下命令：

```
ceph auth get TYPE.ID
```

例如：

```
ceph auth get client.admin
exported keyring for client.admin
[client.admin]
 key = AQA19uZUqIwkHxAAFuUwvq0eJD4S173oFRxe0g==
 caps mds = "allow"
 caps mon = "allow *"
 caps osd = "allow *"
```

开发人员也可以执行以下命令：

```
ceph auth export TYPE.ID
```

`auth export` 命令与 `auth get` 相同，不过它还会列显内部身份验证 ID。

5.2.2.3 添加用户

添加一个用户会创建用户名 (`TYPE.ID`)、机密密钥，以及包含在命令中用于创建该用户的所有使能。

用户可使用其密钥向 Ceph 存储集群进行身份验证。用户的使能授予该用户在 Ceph monitor (mon)、Ceph OSD (osd) 或 Ceph 元数据服务器 (mds) 上进行读取、写入或执行的能力。

可以使用以下几个命令来添加用户：

`ceph auth add`

此命令是添加用户的规范方法。它会创建用户、生成密钥，并添加所有指定的使能。

`ceph auth get-or-create`

此命令往往是创建用户的最简便方式，因为它会返回包含用户名（在方括号中）和密钥的密钥文件格式。如果该用户已存在，此命令只以密钥文件格式返回用户名和密钥。您可以使用 `-o 文件名` 选项将输出保存到某个文件。

`ceph auth get-or-create-key`

此命令是创建用户并仅返回用户密钥的简便方式。对于只需要密钥的客户端（例如 `libvirt`），此命令非常有用。如果该用户已存在，此命令只返回密钥。您可以使用 `-o 文件名` 选项将输出保存到某个文件。

创建客户端用户时，可以创建不具有使能的用户。不具有使能的用户可以进行身份验证，但不能执行其他操作。此类客户端无法从监视器检索集群地图。但是，如果您希望稍后再添加使能，可以使用 `ceph auth caps` 命令创建一个不具有使能的用户。

典型的用户至少对 Ceph monitor 具有读取功能，并对 Ceph OSD 具有读取和写入功能。此外，用户的 OSD 权限通常限制为只能访问特定的存储池。

```
root # ceph auth add client.john mon 'allow r' osd \
 'allow rw pool=liverpool'
root # ceph auth get-or-create client.paul mon 'allow r' osd \
 'allow rw pool=liverpool'
root # ceph auth get-or-create client.george mon 'allow r' osd \
 'allow rw pool=liverpool' -o george.keyring
root # ceph auth get-or-create-key client.ringo mon 'allow r' osd \
 'allow rw pool=liverpool' -o ringo.key
```

重要

如果您为某个用户提供了对 OSD 的使能，但未限制只能访问特定存储池，则该用户将有权访问集群中的所有存储池。

5.2.2.4 修改用户使能

使用 `ceph auth caps` 命令可以指定用户以及更改该用户的使能。设置新使能会覆盖当前的使能。要查看当前使能，请运行 `ceph auth get USERTYPE.USERID`。要添加使能，使用以下格式时还需要指定现有使能：

```
root # ceph auth caps USERTYPE.USERID daemon 'allow [r|w|x|*|...] \
     [pool=pool-name] [namespace=namespace-name]' [daemon 'allow [r|w|x|*|...] \
     [pool=pool-name] [namespace=namespace-name]']
```

例如：

```
root # ceph auth get client.john
root # ceph auth caps client.john mon 'allow r' osd 'allow rw pool=prague'
root # ceph auth caps client.paul mon 'allow rw' osd 'allow rwx pool=prague'
root # ceph auth caps client.brian-manager mon 'allow *' osd 'allow *'
```

要删除某个使能，可重设置该使能。如果希望用户无权访问以前设置的特定守护进程，请指定一个空字符串：

```
root # ceph auth caps client.ringo mon ' ' osd ' '
```

5.2.2.5 删除用户

要删除用户，请使用 `ceph auth del`：

```
root # ceph auth del TYPE.ID
```

其中，`TYPE` 是 `client`、`osd`、`mon` 或 `mds` 之一，`ID` 是用户名或守护进程的 ID。

5.2.2.6 列显用户的密钥

要将用户的身份验证密钥列显到标准输出，请执行以下命令：

```
root # ceph auth print-key TYPE.ID
```

其中，`TYPE` 是 `client`、`osd`、`mon` 或 `mds` 之一，`ID` 是用户名或守护进程的 ID。

需要在客户端软件（例如 `libvirt`）中填充某个用户的密钥时，列显用户的密钥非常有帮助，如以下示例所示：

```
cephadm > sudo mount -t ceph host:/ mount_point \
-o name=client.user,secret=`ceph auth print-key client.user`
```

5.2.2.7 导入用户

要导入一个或多个用户，请使用 `ceph auth import` 并指定密钥环：

```
cephadm > sudo ceph auth import -i /etc/ceph/ceph.keyring
```

注意

Ceph 存储集群将添加新用户及其密钥和使能，并更新现有用户及其密钥和使能。

5.2.3 密钥环管理

当您通过 Ceph 客户端访问 Ceph 时，该客户端会查找本地密钥环。默认情况下，Ceph 会使用以下四个密钥环名称预设置密钥环设置，因此，除非您要覆盖默认值，否则无需在 Ceph 配置文件中设置这些名称：

```
/etc/ceph/cluster.name.keyring
/etc/ceph/cluster.keyring
/etc/ceph/keyring
/etc/ceph/keyring.bin
```

`cluster` 元变量是根据 Ceph 配置文件名称定义的 Ceph 集群名称。`ceph.conf` 表示集群名称为 `ceph`，因此密钥环名称为 `ceph.keyring`。`name` 元变量是用户类型和用户 ID（例如 `client.admin`），因此密钥环名称为 `ceph.client.admin.keyring`。

创建用户（例如 `client.ringo`）之后，必须获取密钥并将其添加到 Ceph 客户端上的密钥环，以使该用户能够访问 Ceph 存储集群。

第 5.2 节 “密钥管理”详细介绍了如何直接在 Ceph 存储集群中列出、获取、添加、修改和删除用户。不过，Ceph 还提供了 `ceph-authtool` 实用程序，可让您从 Ceph 客户端管理密钥环。

5.2.3.1 创建密钥环

当您执行第 5.2 节 “密钥管理”中的过程创建用户时，需要向 Ceph 客户端提供用户密钥，以使客户端能检索指定用户的密钥，并向 Ceph 存储集群进行身份验证。Ceph 客户端将访问钥环，以查找用户名并检索用户的密钥：

```
cephadm > sudo ceph-authtool --create-keyring /path/to/keyring
```

创建包含多个用户的密钥环时，我们建议使用集群名称（例如 `cluster`.keyring）作为密钥环文件名，并将其保存在 `/etc/ceph` 目录中，如此，您无需在 Ceph 配置文件的本地副本中指定文件名，密钥环配置默认设置就会选取正确的文件名。例如，可执行以下命令创建 `ceph.keyring`：

```
cephadm > sudo ceph-authtool -C /etc/ceph/ceph.keyring
```

创建包含单个用户的密钥环时，我们建议使用集群名称、用户类型和用户名，并将其保存在 `/etc/ceph` 目录中。例如，为 `client.admin` 用户创建 `ceph.client.admin.keyring`。

5.2.3.2 将用户添加到密钥环

将某个用户添加到 Ceph 存储集群时（请参见第 5.2.2.3 节 “添加用户”），可以检索该用户、密钥和使能，并将该用户保存到密钥环。

如果您只想对每个密钥环使用一个用户，可以结合 `-o` 选项使用 `ceph auth get` 命令以密钥环文件格式保存输出。例如，要为 `client.admin` 用户创建密钥环，请执行以下命令：

```
root # ceph auth get client.admin -o /etc/ceph/ceph.client.admin.keyring
```

想要将用户导入到密钥环时，可以使用 `ceph-authtool` 指定目标密钥环和源密钥环：

```
cephadm > sudo ceph-authtool /etc/ceph/ceph.keyring \
  --import-keyring /etc/ceph/ceph.client.admin.keyring
```

5.2.3.3 创建用户

Ceph 提供 `ceph auth add` 命令用于直接在 Ceph 存储集群中创建用户。但是，您也可以直接在 Ceph 客户端密钥环中创建用户、密钥和使能。然后，可将用户导入到 Ceph 存储集群：

```
cephadm > sudo ceph-authtool -n client.ringo --cap osd 'allow rwx' \
  --cap mon 'allow rwx' /etc/ceph/ceph.keyring
```

您也可以在创建密钥环的同时将新用户添加到该密钥环：

```
cephadm > sudo ceph-authtool -C /etc/ceph/ceph.keyring -n client.ringo \
  --cap osd 'allow rwx' --cap mon 'allow rwx' --gen-key
```

在前面的方案中，新用户 `client.ringo` 仅存放在密钥环中。要将该新用户添加到 Ceph 存储集群，仍必须手动添加：

```
cephadm > sudo ceph auth add client.ringo -i /etc/ceph/ceph.keyring
```

5.2.3.4 修改用户

要修改密钥环中某条用户记录的使能，请指定该密钥环和用户，然后指定使能：

```
cephadm > sudo ceph-authtool /etc/ceph/ceph.keyring -n client.ringo \
  --cap osd 'allow rwx' --cap mon 'allow rwx'
```

要在 Ceph 集群环境中更新已修改的用户，必须将密钥环中的更改导入到 Ceph 集群中的用户项：

```
root # ceph auth import -i /etc/ceph/ceph.keyring
```

请参见第 5.2.2.7 节 “导入用户”，了解有关根据密钥环更新 Ceph 存储集群用户的详细信息。

5.2.4 命令行用法

`ceph` 命令支持以下与用户名和机密操作相关的选项：

`--id` 或 `--user`

Ceph 使用类型和 ID（`类型.ID`，例如 `client.admin` 或 `client.user1`）来标识用户。使用 `id`、`name` 和 `-n` 选项可以指定用户名的 ID 部分（例如 `admin` 或 `user1`）。可以使用 --id 指定用户，并省略类型。例如，要指定用户 client.foo，请输入以下命令：

```
root # ceph --id foo --keyring /path/to/keyring health
root # ceph --user foo --keyring /path/to/keyring health
```

`--name` 或 `-n`

Ceph 使用类型和 ID（`类型.ID`，例如 `client.admin` 或 `client.user1`）来标识用户。使用 `--name` 和 `-n` 选项可以指定完全限定的用户名。必须指定用户类型（通常是 `client`）和用户 ID：

```
root # ceph --name client.foo --keyring /path/to/keyring health
```

```
root # ceph -n client.foo --keyring /path/to/keyring health
```

`--keyring`

包含一个或多个用户名和机密的密钥环的路径。`--secret` 选项提供相同的功能，但它不适用于对象网关，该网关将 `--secret` 用于其他目的。可以使用 `ceph auth get-or-create` 检索密钥环并将其存储在本地。这是首选的做法，因为无需切换密钥环路径就能切换用户名：

```
cephadm > sudo rbd map --id foo --keyring /path/to/keyring mypool/myimage
```

6 存储的数据管理

CRUSH 算法通过计算数据存储位置来确定如何存储和检索数据。使用 CRUSH，Ceph 客户端无需通过中心服务器或中介程序，即可直接与 OSD 通讯。借助算法确定的数据存储和检索方法，Ceph 可避免单一故障点、性能瓶颈和可伸缩性物理限制。

CRUSH 需要获取集群的地图，它使用 CRUSH 地图以伪随机的方式在 OSD 中存储和检索数据，并以一致的方式在整个集群中分布数据。

CRUSH 地图包含一个 OSD 列表、一个用于将设备聚合到物理位置的“桶”列表，以及一个告知 CRUSH 应如何在 Ceph 集群存储池中复制数据的规则列表。通过反映安装的底层物理组织，CRUSH 可对相关设备故障的潜在根源建模，从而解决故障的根源。典型的根源包括物理接近、共用电源和共用网络。通过将这些信息编码到集群地图中，CRUSH 归置策略可将对象副本分隔在不同的故障域中，同时维持所需的分布方式。例如，为了消除可能的并发故障，可能需要确保数据副本位于使用不同机架、机柜、电源、控制器和/或物理位置的设备上。

部署 Ceph 集群后，将会生成默认的 CRUSH 地图。这种模式适合 Ceph 沙箱环境。但是，在部署大规模的数据集群时，强烈建议您考虑创建自定义 CRUSH 地图，因为这样做有助于管理 Ceph 集群、提高性能并确保数据安全。

例如，如果某个 OSD 停机，而您需要使用现场支持或更换硬件，则 CRUSH 地图可帮助您定位到发生 OSD 故障的主机所在的物理数据中心、机房、设备排和机柜。

同样，CRUSH 可以帮助您更快地确定故障。例如，如果特定机柜中的所有 OSD 同时停机，故障可能是由某个网络交换机或者机柜或网络交换机的电源所致，而不是发生在 OSD 自身上。

当与故障主机关联的归置组处于降级状态时，自定义 CRUSH 地图还可帮助您确定 Ceph 存储数据冗余副本的物理位置。

CRUSH 地图包括三个主要部分。

- 设备数由任何对象存储设备（即与 `ceph-osd` 守护进程对应的硬盘）组成。
- 桶由存储位置（例如设备排、机柜、主机等）及为其指定的权重的分层聚合组成。
- 规则组由桶选择方式组成。

6.1 设备数

为了将归置组映射到 OSD，CRUSH 地图需要 OSD 设备（OSD 守护进程的名称）的列表。设备列表显示在 CRUSH 地图的最前面。

```
#devices
device num osd.name
```

例如：

```
#devices
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3
```

一般而言，一个 OSD 守护进程映射到一个磁盘。

6.2 桶

CRUSH 地图包含 OSD 的列表，可将这些 OSD 组织成“桶”，以便将设备聚合到物理位置。

0	OSD	OSD 守护进程（osd.1、osd.2 等）。
1	主机	包含一个或多个 OSD 的主机名。
2	机箱	组成机柜的机箱。
3	机柜	计算机机柜。默认值为 `unknownrack`。
4	设备排	由一系列机柜组成的设备排。
5	Pdu	电源分配单元。
6	Pod	
7	机房	包含主机机柜和主机排的机房。
8	数据中心	包含机房的物理数据中心。

9	地区	
10	根	

提示

可以删除这些类型，并创建自己的桶类型。

Ceph 的部署工具可生成 CRUSH 地图，其中包含每个主机的桶，以及名为“default”的存储池（可用于默认的 `rbd` 池）。剩余的桶类型提供了一种存储有关节点/桶的物理位置信息的方法，当 OSD、主机或网络硬件发生故障，并且管理员需要访问物理硬件时，这种方法可大大简化集群管理工作。

桶具有类型、唯一的名称（字符串）、以负整数表示的唯一 ID、相对于其项目的总容量/功能的权重、桶算法（默认为 `straw`）和哈希（默认为 `0`，反映 CRUSH 哈希 `rjenkins1`）。一个桶可以包含一个或多个项目。项目可由其他桶或 OSD 组成。项目可能会有一个权重来反映该项目的相对权重。

```
[bucket-type] [bucket-name] {
  id [a unique negative numeric ID]
  weight [the relative capacity/capability of the item(s)]
  alg [the bucket type: uniform | list | tree | straw ]
  hash [the hash type: 0 by default]
  item [item-name] weight [weight]
}
```

下面的示例说明如何使用桶来聚合存储池，以及诸如数据中心、机房、机柜和设备排的物理位置。

```
host ceph-osd-server-1 {
        id -17
        alg straw
        hash 0
        item osd.0 weight 1.00
        item osd.1 weight 1.00
}

row rack-1-row-1 {
```

```
        id -16
        alg straw
        hash 0
        item ceph-osd-server-1 weight 2.00
}

rack rack-3 {
        id -15
        alg straw
        hash 0
        item rack-3-row-1 weight 2.00
        item rack-3-row-2 weight 2.00
        item rack-3-row-3 weight 2.00
        item rack-3-row-4 weight 2.00
        item rack-3-row-5 weight 2.00
}

rack rack-2 {
        id -14
        alg straw
        hash 0
        item rack-2-row-1 weight 2.00
        item rack-2-row-2 weight 2.00
        item rack-2-row-3 weight 2.00
        item rack-2-row-4 weight 2.00
        item rack-2-row-5 weight 2.00
}

rack rack-1 {
        id -13
        alg straw
        hash 0
        item rack-1-row-1 weight 2.00
        item rack-1-row-2 weight 2.00
        item rack-1-row-3 weight 2.00
        item rack-1-row-4 weight 2.00
        item rack-1-row-5 weight 2.00
}
```

```
room server-room-1 {
        id -12
        alg straw
        hash 0
        item rack-1 weight 10.00
        item rack-2 weight 10.00
        item rack-3 weight 10.00
}

datacenter dc-1 {
        id -11
        alg straw
        hash 0
        item server-room-1 weight 30.00
        item server-room-2 weight 30.00
}

pool data {
        id -10
        alg straw
        hash 0
        item dc-1 weight 60.00
        item dc-2 weight 60.00
}
```

6.3 规则组

CRUSH 地图支持“CRUSH 规则”概念，这些规则确定存储池的数据归置。对于大型集群，您可能会创建许多存储池，其中每个存储池各自可能具有自己的 CRUSH 规则组和规则。默认的 CRUSH 地图对每个存储池使用一个规则，并为每个默认存储池指定一个规则组。

注意

大多数情况下，无需修改默认规则。创建新存储池时，该存储池的默认规则组为 0。

规则采用以下格式：

```
rule rulename {

        ruleset ruleset
        type type
        min_size min-size
        max_size max-size
        step step

}
```

ruleset

一个整数。将规则分类，使其属于一个规则组。通过在存储池中设置规则组来激活。必须指定此选项。默认值是 `0`。

重要

需要从默认值 0 开始连续递增规则组编号，否则相关的监视器可能会崩溃。

type

一个字符串。描述硬盘 (replicated) 或 RAID 的规则。必须指定此选项。默认值为 `replicated`。

min_size

一个整数。如果归置组创建的副本数小于此数字，CRUSH 将不选择此规则。必须指定此选项。默认值是 `2`。

max_size

一个整数。如果归置组创建的副本数大于此数字，CRUSH 将不选择此规则。必须指定此选项。默认值是 `10`。

step take `bucket`

采用某个桶名称，并开始在树中向下迭代。必须指定此选项。有关在树中迭代的说明，请参见第 6.3.1 节 “在节点树中迭代”。

step `target mode num` type `bucket-type`

`target` 可以是 `choose` 或 `chooseleaf`。如果设置为 `choose`，则会选择许多桶。`chooseleaf` 直接从桶集的每个桶的子树中选择 OSD（叶节点）。

`mode` 可以是 `firstn` 或 `indep`。请参见第 6.3.2 节 “firstn 和 indep”。

选择给定类型的桶的数量。其中，N 是可用选项的数量，如果 `num` > 0 且 < N，则选择该数量的桶；如果 `num` < 0，则表示 N - `num`；如果 `num` == 0，则选择 N 个桶（全部可用）。跟在 `step take` 或 `step choose` 后使用。

step emit

输出当前值并清空堆栈。通常在规则的末尾使用，但也可在同一规则中用来构成不同的树。跟在 `step choose` 后使用。

重要

要将一个或多个具有共同规则组编号的规则构成某个存储池，请将规则组编号设置为该存储池。

6.3.1 在节点树中迭代

可采用节点树的形式来查看使用桶定义的结构。在此树中，桶是节点，OSD 是叶。

CRUSH 地图中的规则定义如何从此树中选择 OSD。规则从某个节点开始，然后在树中向下迭代，以返回一组 OSD。无法定义需要选择哪个分支。CRUSH 算法可确保 OSD 集能够满足复制要求并均衡分布数据。

使用 `step take bucket` 时，节点树中的迭代从给定的桶（而不是桶类型）开始。如果要返回树中所有分支上的 OSD，该桶必须是根桶。否则，以下步骤只会在子树中迭代。

完成 `step take` 后，接下来会执行规则定义中的一个或多个 `step choose` 项。每个 `step choose` 项从前面选定的上层节点中选择定义数量的节点（或分支）。

最后，使用 `step emit` 返回选定的 OSD。

`step chooseleaf` 是一个便捷函数，可直接从给定桶的分支中选择 OSD。

图 6.1 “示例树”中提供了说明如何使用 `step` 在树中迭代的示例。在下面的规则定义中，橙色箭头和数字与 `example1a` 和 `example1b` 对应，蓝色箭头和数字与 `example2` 对应。

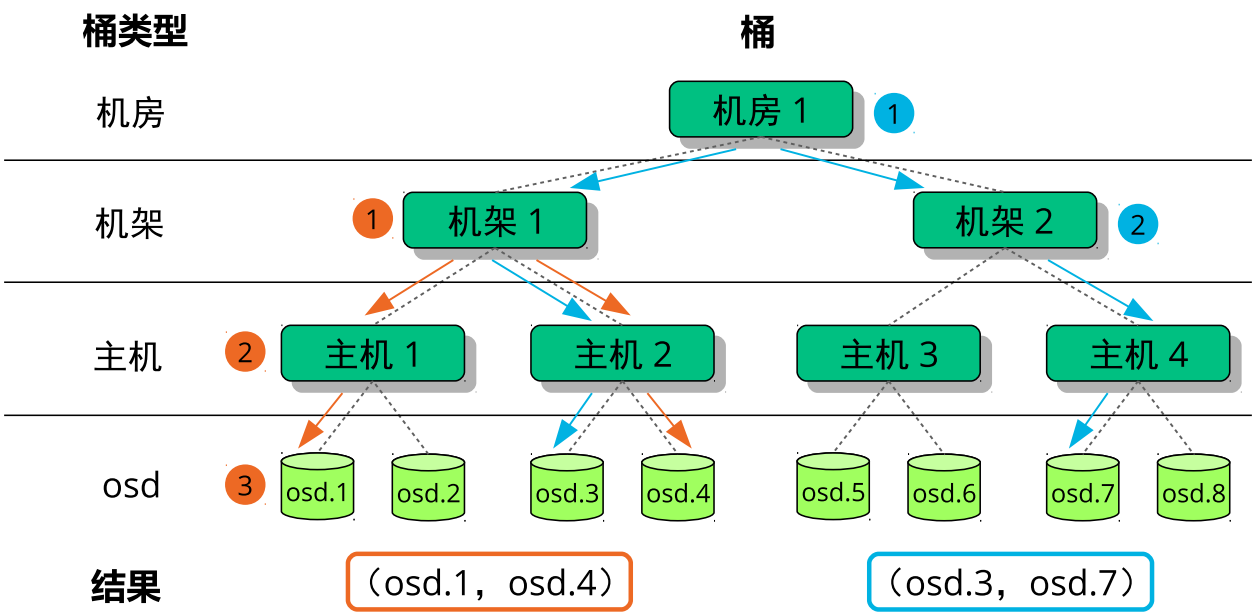


图 6.1： 示例树

```
# orange arrows
rule example1a {
        ruleset 0
        type replicated
        min_size 2
        max_size 10
        # orange (1)
        step take rack1
        # orange (2)
        step choose firstn 0 host
        # orange (3)
        step choose firstn 1 osd
        step emit
}

rule example1b {
        ruleset 0
        type replicated
        min_size 2
        max_size 10
        # orange (1)
```

```
        step take rack1
        # orange (2) + (3)
        step chooseleaf firstn 0 host
        step emit
}

# blue arrows
rule example2 {
        ruleset 0
        type replicated
        min_size 2
        max_size 10
        # blue (1)
        step take room1
        # blue (2)
        step chooseleaf firstn 0 rack
        step emit
}
```

6.3.2 firstn 和 indep

CRUSH 规则定义有故障节点或 OSD 的替换项（请参见第 6.3 节 “规则组”）。关键字 `step` 要求使用 `firstn` 或 `indep` 参数。图 6.2 “节点替换方法”提供了示例。

`firstn` 将替换节点添加到活动节点列表的末尾。如果某个节点发生故障，其后的正常节点会移位到左侧，以填充有故障节点留下的空缺。这是副本池的默认方法，也是需要采取的方法，因为次要节点已包含所有数据，因此可立即接管主要节点的职责。

`indep` 为每个活动节点选择固定的替换节点。替换有故障节点不会更改剩余节点的顺序。这对于纠删码池而言是所需的行为。在纠删码池中，节点上存储的数据取决于在选择节点时它所在的位置。如果节点的顺序发生变化，受影响节点上的所有数据都需要重新放置。

注意：纠删池

确保针对每个纠删码池设置使用 `indep` 的规则。

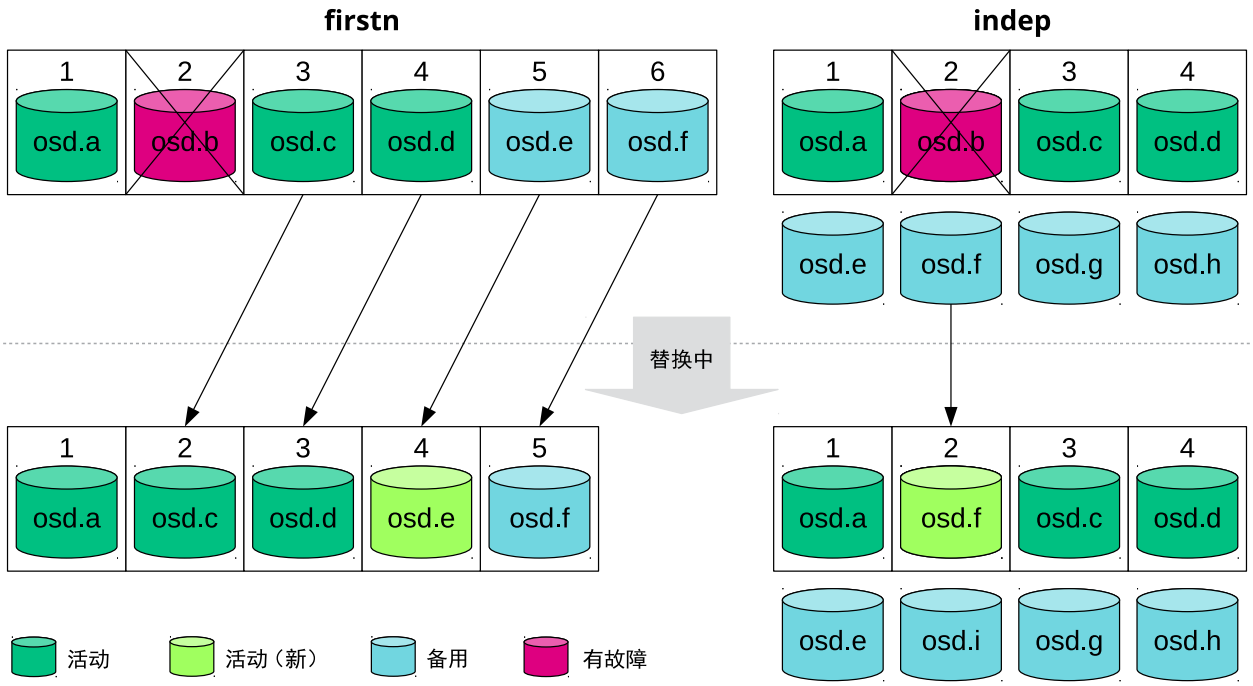


图 6.2： 节点替换方法

6.4 CRUSH 地图操作

本节介绍基本的 CRUSH 地图操作方法，例如编辑 CRUSH 地图、更改 CRUSH 地图参数，以及添加/移动/删除 OSD。

6.4.1 编辑 CRUSH 地图

要编辑现有的 CRUSH 地图，请执行以下操作：

1. 获取 CRUSH 地图。要获取集群的 CRUSH 地图，请执行以下命令：

```
root # ceph osd getcrushmap -o compiled-crushmap-filename
```

Ceph 会将编译的 CRUSH 地图输出 (`-o`) 到您指定名称的文件。由于该 CRUSH 地图采用编译格式，您必须先将其反编译，然后才能对其进行编辑。

2. 反编译 CRUSH 地图。要反编译 CRUSH 地图，请执行以下命令：

```
cephadm > crushtool -d compiled-crushmap-filename \
 -o decompiled-crushmap-filename
```

Ceph 将对已编译的 CRUSH 地图进行反编译 (`-d`)，并将其输出 (`-o`) 到您指定名称的文件。

3. 至少编辑“设备”、“桶”和“规则”中的其中一个参数。
4. 编译 CRUSH 地图。要编译 CRUSH 地图，请执行以下命令：

```
cephadm > crushtool -c decompiled-crush-map-filename \
 -o compiled-crush-map-filename
```

Ceph 会将编译的 CRUSH 地图存储到您指定名称的文件。

5. 设置 CRUSH 地图。要设置集群的 CRUSH 地图，请执行以下命令：

```
root # ceph osd setcrushmap -i compiled-crushmap-filename
```

Ceph 将输入您所指定文件名的已编译 CRUSH 地图，作为集群的 CRUSH 地图。

6.4.2 添加/移动 OSD

要在运行中集群的 CRUSH 地图中添加或移动 OSD，请执行以下命令：

```
root # ceph osd crush set id_or_name weight root=pool-name
bucket-type=bucket-name ...
```

id

一个整数。OSD 的数字 ID。必须指定此选项。

name

一个字符串。OSD 的全名。必须指定此选项。

weight

一个双精度值。OSD 的 CRUSH 权重。必须指定此选项。

pool

一个键/值对。默认情况下，CRUSH 层次结构包含 default 存储池作为根。必须指定此选项。

bucket-type

键/值对。可在 CRUSH 层次结构中指定 OSD 的位置。

下面的示例将 `osd.0` 添加到层次结构，或移动之前某个位置的 OSD。

```
root # ceph osd crush set osd.0 1.0 root=data datacenter=dc1 room=room1 \
row=foo rack=bar host=foo-bar-1
```

6.4.3 调整 OSD 的 CRUSH 权重

要在运行中集群的 CRUSH 地图中调整 OSD 的 CRUSH 权重，请执行以下命令：

```
root # ceph osd crush reweight name weight
```

name

一个字符串。OSD 的全名。必须指定此选项。

weight

一个双精度值。OSD 的 CRUSH 权重。必须指定此选项。

6.4.4 删除 OSD

要从运行中集群的 CRUSH 地图中删除 OSD，请执行以下命令：

```
root # ceph osd crush remove name
```

name

一个字符串。OSD 的全名。必须指定此选项。

6.4.5 移动桶

要将某个桶移到 CRUSH 地图层次结构中的不同位置，请执行以下命令：

```
root # ceph osd crush move bucket-name bucket-type=bucket-name, ...
```

bucket-name

一个字符串。要移动/重新定位的桶的名称。必须指定此选项。

bucket-type

键/值对。可在 CRUSH 层次结构中指定桶的位置。

6.5 整理 (Scrub)

除了复制对象的多个副本外，Ceph 还需通过整理 (scrub) 归置组来确保数据完整性。Ceph 的整理 (scrub) 类似于在对象存储层运行 `fsck`。对于每个归置组，Ceph 都会生成一个包含所有对象的编目，并比较每个主对象及其副本，以确保不会有缺失或不匹配的对象。每天的浅层整理 (light scrub) 会检查对象大小和属性，而每周的深层整理 (deep scrub) 则会读取数据并使用校验和来确保数据完整性。

整理 (scrub) 对于维护数据完整性非常重要，但该操作可能会降低性能。您可以通过调整以下设置来增加或减少整理 (scrub) 操作：

`osd max scrubs`

针对 Ceph OSD 同时执行的最大整理 (scrub) 操作数量。默认值是 1。

`osd scrub begin hour`、`osd scrub end hour`

按小时定义一天内可以执行整理 (scrub) 的时间段（0 到 24）。默认开始时间为 0，结束时间为 24。

重要

如果归置组的整理 (scrub) 间隔超出 `osd scrub max interval` 设置的值，则无论您定义的整理 (scrub) 时间段为何，都将执行整理 (scrub)。

`osd scrub during recovery`

允许恢复期间执行整理 (scrub)。如果将此选项设置为“false”，则当存在活动的恢复进程时，将禁止安排新的整理 (scrub)。已在运行的整理 (scrub) 将继续执行。此选项有助于降低忙碌集群上的负载。默认值为“true”。

`osd scrub thread timeout`

整理 (scrub) 线程超时前的最长时间（以秒为单位）。默认值是 60。

`osd scrub finalize thread timeout`

整理 (scrub) 完成线程超时前的最长时间（以秒为单位）。默认值为 60*10。

`osd scrub load threshold`

规范化的最大负载。当系统负载（由 `getloadavg()` 与 `online cpus` 数量之比定义）高于此数字时，Ceph 将不会执行整理 (scrub)。默认值是 0.5。

`osd scrub min interval`

当 Ceph 集群负载较低时整理 (scrub) Ceph OSD 的最短间隔（以秒为单位）。默认值为 60*60*24（一天一次）。

`osd scrub max interval`

无论集群负载如何都整理 (scrub) Ceph OSD 的最长间隔（以秒为单位）。7*60*60*24（一周一次）。

`osd scrub chunk min`

单个操作期间要整理 (scrub) 的对象存储块的最小数量。整理 (scrub) 期间，Ceph 会阻止向单个块写入数据。默认值是 5。

`osd scrub chunk max`

单个操作期间要整理 (scrub) 的对象存储块的最大数量。默认值是 25。

`osd scrub sleep`

整理 (scrub) 下一组存储块之前休眠的时间。增大此值会降低整个整理 (scrub) 操作的速度，但对客户端操作的影响较小。默认值是 0。

`osd deep scrub interval`

深层整理 (deep scrub)（完整读取所有数据）的间隔。`osd scrub load threshold` 选项不会影响此设置。默认值为 60*60*24*7（一周一次）。

`osd scrub interval randomize ratio`

在安排归置组的下一次整理 (scrub) 作业时，为 `osd scrub min interval` 值增加一个随机延迟。该延迟为一个随机的值，小于 `osd scrub min interval` * `osd scrub interval randomized ratio` 所得结果。因此，该默认设置实际上是将整理 (scrub) 随机地安排在允许的时间段 [1, 1.5] * `osd scrub min interval` 内执行。默认值为 0.5

`osd deep scrub stride`

执行深层整理 (deep scrub) 时读取的大小。默认值为 524288 (512 kB)。

6.6 在同一个节点上混用 SSD 和 HDD

有时，用户可能需要配置这样一个 Ceph 集群：在每个节点上混用 SSD 和 HDD，并将一个存储池放在速度较快的 SSD 上，将另一个存储池放在速度较慢的 HDD 上。要实现此目的，需要编辑 CRUSH 地图。

默认的 CRUSH 地图采用简单的层次结构，其中，默认根包含主机，而主机包含 OSD，例如：

```
root # ceph osd tree
 ID CLASS WEIGHT   TYPE NAME      STATUS REWEIGHT PRI-AFF
 -1       83.17899 root default
 -4       23.86200     host cpach
 2   hdd  1.81898         osd.2      up  1.00000 1.00000
 3   hdd  1.81898         osd.3      up  1.00000 1.00000
 4   hdd  1.81898         osd.4      up  1.00000 1.00000
 5   hdd  1.81898         osd.5      up  1.00000 1.00000
 6   hdd  1.81898         osd.6      up  1.00000 1.00000
 7   hdd  1.81898         osd.7      up  1.00000 1.00000
 8   hdd  1.81898         osd.8      up  1.00000 1.00000
 15  hdd  1.81898         osd.15     up  1.00000 1.00000
 10  nvme 0.93100         osd.10     up  1.00000 1.00000
 0   ssd  0.93100         osd.0      up  1.00000 1.00000
 9   ssd  0.93100         osd.9      up  1.00000 1.00000
```

这样，就无法区分磁盘类型。要将 OSD 分为 SSD 和 HDD，需在 CRUSH 地图中创建另一个层次结构：

```
root # ceph osd crush add-bucket ssd root
```

为 SSD 创建新根后，需在此根中添加主机。这意味着需要创建新的主机项。但是，由于同一个主机名不能在 CRUSH 地图中出现多次，此处使用了虚设的主机名。这些虚设的主机名无需由 DNS 解析。CRUSH 不关心主机名是什么，只需创建适当的层次结构即可。要支持虚设的主机名，真正需要进行的一项更改就是必须设置

```
osd crush update on start = false
```

（在 /srv/salt/ceph/configuration/files/ceph.conf.d/global.conf 文件中），然后运行 DeepSea 阶段 3 以分发该项更改（有关详细信息，请参见第 1.11 节 “自定义 ceph.conf 文件”）：

```
root@master # salt-run state.orch ceph.stage.3
```

否则，您移动的 OSD 稍后将重设置到其在默认根中的原始位置，并且集群不会按预期方式工作。

更改该设置后，请将新的虚设主机添加到 SSD 的根中：

```
root # ceph osd crush add-bucket node1-ssd host
root # ceph osd crush move node1-ssd root=ssd
root # ceph osd crush add-bucket node2-ssd host
root # ceph osd crush move node2-ssd root=ssd
root # ceph osd crush add-bucket node3-ssd host
root # ceph osd crush move node3-ssd root=ssd
```

最后，针对每个 SSD OSD，将 OSD 移到 SSD 的根中。在本示例中，我们假设 osd.0、osd.1 和 osd.2 实际托管在 SSD 上：

```
root # ceph osd crush add osd.0 1 root=ssd
root # ceph osd crush set osd.0 1 root=ssd host=node1-ssd
root # ceph osd crush add osd.1 1 root=ssd
root # ceph osd crush set osd.1 1 root=ssd host=node2-ssd
root # ceph osd crush add osd.2 1 root=ssd
root # ceph osd crush set osd.2 1 root=ssd host=node3-ssd
```

CRUSH 层次结构现在应类似下方所示：

```
root # ceph osd tree
ID WEIGHT  TYPE NAME                 UP/DOWN REWEIGHT PRIMARY-AFFINITY
-5 3.00000 root ssd
-6 1.00000     host node1-ssd
 0 1.00000         osd.0                  up  1.00000          1.00000
-7 1.00000     host node2-ssd
 1 1.00000         osd.1                  up  1.00000          1.00000
-8 1.00000     host node3-ssd
 2 1.00000         osd.2                  up  1.00000          1.00000
```

```
-1 0.11096 root default
-2 0.03699     host node1
 3 0.01849         osd.3                  up  1.00000          1.00000
 6 0.01849         osd.6                  up  1.00000          1.00000
-3 0.03699     host node2
 4 0.01849         osd.4                  up  1.00000          1.00000
 7 0.01849         osd.7                  up  1.00000          1.00000
-4 0.03699     host node3
 5 0.01849         osd.5                  up  1.00000          1.00000
 8 0.01849         osd.8                  up  1.00000          1.00000
```

现在，创建一个针对 SSD 根的 CRUSH 规则：

```
root # ceph osd crush rule create-simple ssd_replicated_ruleset ssd host
```

原始默认规则组 `replicated_ruleset`（ID 为 0）针对的是 HDD。新规则组 `ssd_replicated_ruleset`（ID 为 1）针对的是 SSD。

所有现有存储池仍会使用 HDD，因为它们位于 CRUSH 地图的默认层次结构中。可以创建一个仅使用 SSD 的新存储池：

```
root # ceph osd pool create ssd-pool 64 64
root # ceph osd pool set ssd-pool crush_rule ssd_replicated_ruleset
```

7 管理存储池

Ceph 将数据存储在存储池中。存储池是用于存储对象的逻辑组。如果您先部署集群而不创建存储池，Ceph 会使用默认存储池来存储数据。存储池为您提供：

- 恢复能力：您可以设置允许多少个 OSD 发生故障而不会丢失数据。对于副本池，它是对象的所需副本数。创建新存储池时，会将默认副本数设置为 3。因为典型配置会存储一个对象和一个额外的副本，所以您需要将副本数设置为 2。对于纠删码池，该计数为编码块数（在纠删码配置中，设置为 m=2）。
- 归置组：用于跨 OSD 将数据存储在某个存储池中的内部数据结构。CRUSH 地图中定义了 Ceph 将数据存储到 PG 中的方式。您可以设置存储池的归置组数。典型配置为每个 OSD 使用约 100 个归置组，以提供最佳平衡而又不会耗费太多计算资源。设置多个存储池时，请务将存储池和集群作为整体考虑，确保设置合理的归置组数。
- CRUSH 规则：在存储池中存储数据时，映射到存储池的 CRUSH 规则组可让 CRUSH 识别将对象及其副本（对于纠删码池，则为块）放置在集群中的规则。您可为存储池创建自定义 CRUSH 规则。
- 快照：使用 `ceph osd pool mksnap` 创建快照时，可高效创建特定存储池的快照。
- 设置所有权：您可将某用户 ID 设置为存储池的所有者。

要将数据组织到存储池中，可以列出、创建和删除存储池。您还可以查看每个存储池的用量统计数字。

7.1 将存储池与应用关联

在使用存储池之前，需要将它们与应用关联。将与 CephFS 搭配使用或由对象网关自动创建的存储池会自动关联。需要使用 `rbd` 工具初始化要与 RBD 搭配使用的存储池（有关详细信息，请参见第 8.1 节 “块设备命令”）。

对于其他情况，可以手动将自由格式的应用名称与存储池关联：

```
root # ceph osd pool application enable pool_name application_name
```

提示：默认应用名称

CephFS 使用应用名称 `cephfs`，RADOS 块设备使用 `rbd`，对象网关使用 `rgw`。

一个存储池可以与多个应用关联，每个应用都可具有自己的元数据。可使用以下命令显示给定存储池的应用元数据：

```
root # ceph osd pool application get pool_name
```

7.2 操作存储池

本节介绍对存储池执行基本任务的特定信息。您可以了解如何列出、创建和删除存储池，以及如何显示存储池统计数字或管理存储池快照。

7.2.1 列出存储池

要列出集群的存储池，请执行以下命令：

```
root # ceph osd lspools
0 rbd, 1 photo_collection, 2 foo_pool,
```

7.2.2 创建存储池

要创建副本存储池，请执行以下命令：

```
root # ceph osd pool create pool_name pg_num pgp_num
 replicated crush_ruleset_name \
expected_num_objects
```

要创建纠删码池，请执行以下命令：

```
root # ceph osd pool create pool_name pg_num pgp_num
 erasure erasure_code_profile \
```

```
crush_ruleset_name expected_num_objects
```

如果超出每个 OSD 的归置组限制，则 `ceph osd pool create` 可能会失败。该限制通过 `mon_max_pg_per_osd` 选项设置。

pool_name
: 存储池的名称，必须唯一。必须指定此选项。

pg_num
: 存储池的归置组总数。必须指定此选项。默认值是 8。

pgp_num
: 用于归置数据的归置组总数。此数量应该与归置组总数相等，归置组拆分情况除外。必须指定此选项。默认值是 8。

pgp_type
: 存储池类型，可以是 replicated（用于保留对象的多个副本，以便从失败的 OSD 恢复）或 erasure（用于获得某种通用 RAID5 功能）。副本池需要的原始存储较多，但可实现所有 Ceph 操作。纠删码池需要的原始存储较少，但只实现一部分可用的操作。默认值是“replicated”。

crush_ruleset_name
: 此存储池的 crush 规则组的名称。如果所指定的规则组不存在，则创建副本池的操作将会失败，并显示 -ENOENT。但副本池将使用指定的名称创建新的纠删规则组。对于纠删码池，默认值是“erasure-code”。对于副本池，将选取 Ceph 配置变量 `osd_osd_pool_default_crush_replicated_ruleset`。

erasure_code_profile=profile
: 仅适用于纠删码池。使用纠删码配置。该配置必须是 `osd erasure-code-profile set` 所定义的现有配置。

  创建存储池时，请将归置组数设置为合理的值（例如 100）。还需考虑每个 OSD 的归置组总数。归置组在计算方面的开销很高，因此如果您的许多存储池都包含很多归置组（例如有 50 个池，每个池各有 100 个归置组），性能将会下降。下降点的恢复视 OSD 主机性能而定。

  有关计算存储池的合适归置组数量的详细信息，请参见归置组 (http://docs.ceph.com/docs/master/rados/operations/placement-groups/)↗。

expected_num_objects

此存储池的预期对象数。如果设置此值，PG 文件夹拆分发生于存储池创建时。这可避免因运行时文件夹拆分导致的延迟影响。

7.2.3 设置存储池配额

您可以设置存储池配额，限定每个存储池的最大字节数和/或最大对象数。

```
root # ceph osd pool set-quota pool-name max_objects obj-count max_bytes bytes
```

例如：

```
root # ceph osd pool set-quota data max_objects 10000
```

要删除配额，请将其值设置为 0。

7.2.4 删除存储池

警告：删除存储池的操作不可逆

存储池中可能包含重要数据。删除存储池会导致存储池中的所有数据消失，且无法恢复。

不小心删除存储池十分危险，因此 Ceph 实施了两个机制来防止删除存储池。要删除存储池，必须先禁用这两个机制。

第一个机制是 `NODELETE` 标志。每个存储池都有这个标志，其默认值是“false”。要确定某个存储池的此标志值，请运行以下命令：

```
root # ceph osd pool get pool_name nodelete
```

如果命令输出 `nodelete: true`，则只有在使用以下命令更改该标志后，才能删除存储池：

```
ceph osd pool set pool_name nodelete false
```

第二个机制是集群范围的配置参数 `mon allow pool delete`，其默认值为“false”。这表示默认不能删除存储池。显示的错误讯息是：

```
Error EPERM: pool deletion is disabled; you must first set the
mon_allow_pool_delete config option to true before you can destroy a pool
```

若要规避此安全设置删除存储池，可以临时将 `mon allow pool delete` 设置为“true”，删除存储池，然后将该参数恢复为“false”：

```
root # ceph tell mon.* injectargs --mon-allow-pool-delete=true
root # ceph osd pool delete pool_name pool_name --yes-i-really-really-mean-it
root # ceph tell mon.* injectargs --mon-allow-pool-delete=false
```

`injectargs` 命令会显示以下讯息：

```
injectargs:mon_allow_pool_delete = 'true' (not observed, change may require
 restart)
```

这主要用于确认该命令已成功执行。它不是错误。

如果为您创建的存储池创建了自己的规则组和规则，则应该考虑在不再需要该存储池时删除规则组和规则。如果您创建了仅对不再存在的存储池具有许可权限的用户，则应该考虑也删除那些用户。

7.2.5 重命名存储池

要重命名存储池，请执行以下命令：

```
root # ceph osd pool rename current-pool-name new-pool-name
```

如果重命名了存储池，且为经过身份验证的用户使用了按存储池功能，则必须用新的存储池名称更新用户的功能。

7.2.6 显示存储池统计数字

要显示存储池的用量统计数字，请执行以下命令：

```
root # rados df
pool name  category  KB  objects   lones  degraded  unfound  rd  rd KB  wr  wr
 KB
```

```
cold-storage     -    228    1          0       0           0        0    0       1    228
data             -      1    4          0       0           0        0    0       4      4
hot-storage      -      1    2          0       0           0       15   10       5    231
metadata         -      0    0          0       0           0        0    0       0      0
pool1            -      0    0          0       0           0        0    0       0      0
rbd              -      0    0          0       0           0        0    0       0      0
total used           266268          7
total avail        27966296
total space        28232564
```

7.2.7 设置存储池的值

要设置存储池的值，请执行以下命令：

```
root # ceph osd pool set pool-name key value
```

您可以设置以下键的值：

size

设置存储池中对象的副本数。有关更多详细信息，请参见第 7.2.9 节 “设置对象副本数”。仅用于副本池。

min_size

设置 I/O 所需的最小副本数。有关更多详细信息，请参见第 7.2.9 节 “设置对象副本数”。仅用于副本池。

crash_replay_interval

允许客户端重放已确认但未提交的请求的秒数。

pg_num

存储池的归置组数。如果将 OSD 添加到集群，则应该提高归置组的值，有关详细信息，请参见第 7.2.11 节 “增加归置组数”。

pgp_num

计算数据归置时要使用的归置组的有效数量。

crush_ruleset

用于在集群中映射对象归置的规则组。

hashpspool

为给定存储池设置 (1) 或取消设置 (0) HASHPSPOOL 标志。启用此标志会更改算法，以采用更佳的方式将 PG 分配到 OSD 之间。对之前 HASHPSPOOL 标志设为 0 的存储池启用此标志后，集群会开始回填，以使所有 PG 都可再次正确归置。请注意，这可能会在集群上产生相当高的 I/O 负载，因此对高负载生产集群必须进行妥善规划。

nodelete

防止删除存储池。

nopgchange

防止更改存储池的 `pg_num` 和 `pgp_num`。

nosizechange

防止更改存储池的大小。

write_fadvise_dontneed

对给定存储池设置/取消设置 `WRITE_FADVISE_DONTNEED` 标志。

noscrub、nodeep-scrub

禁用（深层）整理 (scrub) 特定存储池的数据以解决临时高 I/O 负载问题。

hit_set_type

对快速缓存池启用命中集跟踪。请参见布隆过滤器 (http://en.wikipedia.org/wiki/Bloom_filter)↗以了解更多信息。此选项可用的值如下：`bloom`、`explicit_hash`、`explicit_object`。默认值是 `bloom`，其他值仅用于测试。

hit_set_count

要为快速缓存池存储的命中集数。该数值越高，`ceph-osd` 守护进程耗用的 RAM 越多。默认值是 `0`。

hit_set_period

快速缓存池的命中集期间的时长（以秒为单位）。该数值越高，`ceph-osd` 守护进程耗用的 RAM 越多。

hit_set_fpp

布隆命中集类型的误报率。请参见布隆过滤器 (http://en.wikipedia.org/wiki/Bloom_filter)↗以了解更多信息。有效范围是 0.0 - 1.0，默认值是 `0.05`

use_gmt_hitset

为快速缓存分层创建命中集时，强制 OSD 使用 GMT（格林威治标准时间）时戳。这可确保在不同时区中的节点返回相同的结果。默认值是 `1`。不应该更改此值。

cache_target_dirty_ratio

在快速缓存分层代理将已修改（脏）对象清理到后备存储池之前，包含此类对象的快速缓存池百分比。默认值是 `.4`

cache_target_dirty_high_ratio

在快速缓存分层代理将已修改（脏）对象清理到速度更快的后备存储池之前，包含此类对象的快速缓存池百分比。默认值是 `.6`。

cache_target_full_ratio

在快速缓存分层代理将未修改（干净）对象从快速缓存池逐出之前，包含此类对象的快速缓存池百分比。默认值是 `.8`

target_max_bytes

触发 `max_bytes` 阈值后，Ceph 将会开始清理或逐出对象。

target_max_objects

触发 `max_objects` 阈值时，Ceph 将开始清理或逐出对象。

hit_set_grade_decay_rate

两次连续的 `hit_set` 之间的温度降低率。默认值是 `20`。

hit_set_search_last_n

计算温度时在 `hit_set` 中对出现的项最多计 `N` 次。默认值是 `1`。

cache_min_flush_age

在快速缓存分层代理将对象从快速缓存池清理到存储池之前的时间（秒）。

cache_min_evict_age

在快速缓存分层代理将对象从快速缓存池中逐出之前的时间（秒）。

fast_read

如果对纠删码池启用此标志，则读取请求会向所有分片发出子读取命令，并一直等到接收到足够解码的分片，才会为客户端提供服务。对于 jerasure 和 isa 纠删插件，前 `K` 个副本返回时，就会使用从这些副本解码的数据立即处理客户端的请求。这有助于获得一些资源以提高性能。目前，此标志仅支持用于纠删码池。默认值是 `0`。

scrub_min_interval

集群负载低时整理 (scrub) 存储池的最小间隔（秒）。默认值 `0` 表示使用来自 Ceph 配置文件的 `osd_scrub_min_interval` 值。

scrub_max_interval

不论集群负载如何都整理 (scrub) 存储池的最大间隔（秒）。默认值 `0` 表示使用来自 Ceph 配置文件的 `osd_scrub_max_interval` 值。

deep_scrub_interval

深层整理 (scrub) 存储池的间隔（秒）。默认值 `0` 表示使用来自 Ceph 配置文件的 `osd_deep_scrub` 值。

7.2.8 获取存储池的值

要获取存储池中的值，请执行以下命令：

```
root # ceph osd pool get pool-name key
```

您可以获取第 7.2.7 节 “设置存储池的值”中所列键以及下列键的值：

pg_num

存储池的归置组数。

pgp_num

计算数据归置时要使用的归置组的有效数量。有效范围小于或等于 `pg_num`。

7.2.9 设置对象副本数

要设置副本存储池上的对象副本数，请执行以下命令：

```
root # ceph osd pool set poolname size num-replicas
```

`num-replicas` 包括对象本身。例如，如果您想用对象和对象的两个副本组成对象的三个实例，请指定 3。

如果将 `num-replicas` 设置为 2，数据将只有一个副本。例如，如果您丢失了一个对象实例，则需要在恢复期间确定自上次整理 (scrub) (http://ceph.com/docs/master/rados/configuration/osd-config-ref/#scrubbing) 后，另一个副本没有损坏。

将存储池设置为具有一个副本意味着存储池中的数据对象只有一个实例。如果 OSD 发生故障，您将丢失数据。如果要短时间存储临时数据，可能就会用到只有一个副本的存储池。

为存储池设置三个以上副本只能小幅提高可靠性，但在极少数情况下可能适用。请记住，副本越多，存储对象副本所需的磁盘空间就越多。如果您需要终极数据安全性，则建议使用纠删码池。有关详细信息，请参见第 9 章 “纠删码池”。

警告：建议使用两个以上副本

强烈建议不要只使用 2 个副本。如果一个 OSD 发生故障，恢复期间的高负载很可能会导致第二个 OSD 也发生故障。

例如：

```
root # ceph osd pool set data size 3
```

可针对每个存储池执行此命令。

注意

对象可以接受降级模式下副本数量低于 `pool size` 的 I/O。要设置 I/O 所需副本的最小数目，应该使用 `min_size` 设置。例如：

```
root # ceph osd pool set data min_size 2
```

这可确保数据池中没有对象会接收到副本数量低于 `min_size` 的 I/O。

7.2.10 获取对象副本数

要获取对象副本数，请执行以下命令：

```
root # ceph osd dump | grep 'replicated size'
```

Ceph 将列出存储池，并高亮显示 `replicated size` 属性。Ceph 默认会创建对象的两个副本（共三个副本，或者大小为 3）。

7.2.11 增加归置组数

创建新存储池时，需指定存储池的归置组数（请参见第 7.2.2 节 “创建存储池”）。将更多 OSD 添加至集群后，出于性能和数据持久性原因，通常还需要增加归置组数。对于每个归置组，OSD 和监视器节点始终都需要用到内存、网络和 CPU，在恢复期间需求量甚至更大。因此，最大限度地减少归置组数可节省相当大的资源量。

警告：pg_num 的值过高

更改存储池的 `pg_num` 值时，新的归置组数有可能会超出允许的限制。例如

```
root # ceph osd pool set rbd pg_num 4096
 Error E2BIG: specified pg_num 3500 is too large (creating 4096 new PGs \
 on ~64 OSDs exceeds per-OSD max of 32)
```

该限制可防止归置组过度拆分，它从 `mon_osd_max_split_count` 值衍生。

为已调整大小的集群确定合适的新归置组数是一项复杂的任务。一种方法是不断增加归置组数，直到达到集群性能的最佳状态。要确定增加后的新归置组数，需要获取 `mon_osd_max_split_count` 参数的值，并将它与当前的归置组数相加。要了解基本原理，请查看下面的脚本：

```
cephadm > max_inc=`ceph daemon mon.a config get mon_osd_max_split_count 2>&1 \
  | tr -d '\n ' | sed 's/.*"\([[:digit:]]\+\)".*/\1/'`
cephadm > pg_num=`ceph osd pool get rbd pg_num | cut -f2 -d: | tr -d ' '`
cephadm > echo "current pg_num value: $pg_num, max increment: $max_inc"
cephadm > next_pg_num="$(($pg_num+$max_inc))"
cephadm > echo "allowed increment of pg_num: $next_pg_num"
```

确定新的归置组数之后，使用以下命令来增加该数量：

```
root # ceph osd pool set pool_name pg_num next_pg_num
```

7.2.12 添加存储池

在您首次部署集群之后，Ceph 会使用默认存储池来存储数据。之后，您可以使用以下命令创建新的存储池：

```
root # ceph osd pool create
```

有关创建集群存储池的详细信息，请参见第 7.2.2 节 “创建存储池”。

7.3 存储池迁移

创建存储池（请参见第 7.2.2 节 “创建存储池”）时，您需要指定存储池的初始参数，例如存储池类型或归置组数量。如果您在存储池内放置数据后，又决定更改任何初始参数，则需要将存储池数据迁移到参数适合您的部署的另一个存储池中。

迁移存储池的方法有多种。建议使用快速缓存层，因为该方法是透明的，能够减少集群停机时间并避免复制所有存储池的数据。

7.3.1 使用快速缓存层迁移

该方法的原理十分简单，只需将需要迁移的存储池按相反的顺序加入快速缓存层中即可。有关快速缓存层的详细信息，请参见第 10 章 “快速缓存分层”。例如，要将名为“testpool”的副本池迁移到纠删码池，请执行以下步骤：

过程 7.1： 将副本池迁移到纠删码池

1. 创建一个名为“newpool”的新纠删码池：

   ```
   root@minion > ceph osd pool create newpool 4096 4096 erasure default
   ```

   您现在有两个池，即装满数据的原始副本池“testpool”和新的空纠删码池“newpool”：

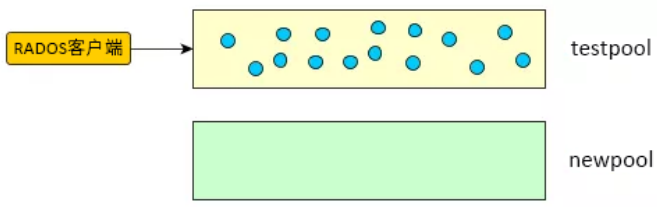


   图 7.1： 迁移前的存储池

2. 设置快速缓存层，并将副本池“testpool”配置为快速缓存池：

```
root@minion > ceph osd tier add newpool testpool --force-nonempty
root@minion > ceph osd cache-mode testpool forward
```

自此之后，所有新对象都将创建在新池中：

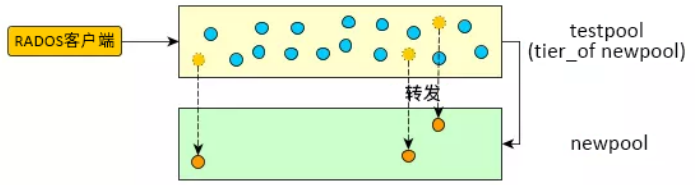


图 7.2： 快速缓存层设置

3. 强制快速缓存池将所有对象移到新池中：

```
root@minion > rados -p testpool cache-flush-evict-all
```

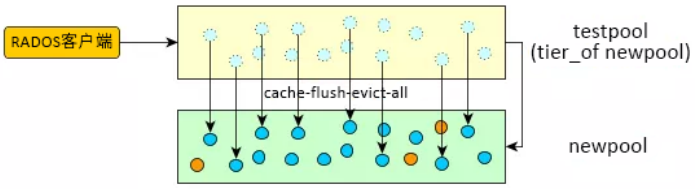


图 7.3： 数据清理

4. 将所有客户端切换到新池。您需要指定一个覆盖层，以便在旧池中搜索对象，直到所有数据都已清理到新的纠删码池。

```
root@minion > ceph osd tier set-overlay newpool testpool
```

有了覆盖层，所有操作都会转到旧的副本池“testpool”：

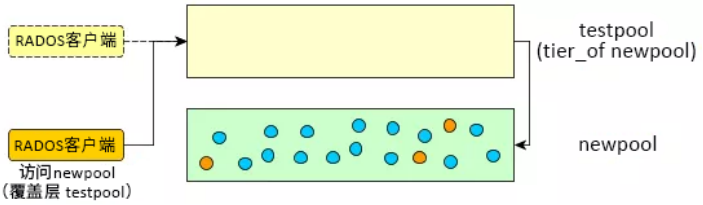


图 7.4： 设置覆盖层

现在，您可以将所有客户端都切换为访问新池中的对象。

5. 所有数据都迁移到纠删码池“newpool”后，删除覆盖层和旧超速缓冲池“testpool”：

```
root@minion > ceph osd tier remove-overlay newpool
root@minion > ceph osd tier remove newpool testpool
```

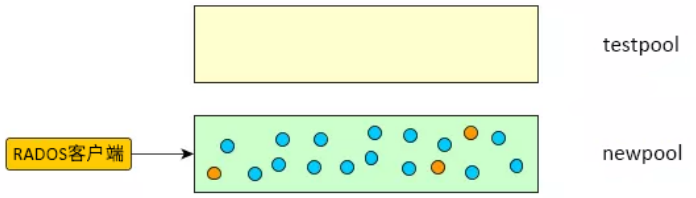


图 7.5： 迁移完成

7.4 存储池快照

存储池快照是整个 Ceph 存储池的状态快照。通过存储池快照，可以保留存储池状态的历史。创建存储池快照可能需要大量存储空间，具体取决于存储池的大小。在创建存储池快照之前，始终需要检查相关存储是否有足够的磁盘空间。

7.4.1 创建存储池快照

要创建存储池快照，请执行以下命令：

```
root # ceph osd pool mksnap pool-name snap-name
```

例如：

```
root # ceph osd pool mksnap pool1 snapshot1
created pool pool1 snap snapshot1
```

7.4.2 删除存储池快照

要删除存储池快照，请执行以下命令：

```
root # ceph osd pool rmsnap pool-name snap-name
```

7.5 数据压缩

从 SUSE Enterprise Storage 5 开始，BlueStore 提供即时数据压缩，以节省磁盘空间。

7.5.1 启用压缩

可使用以下命令启用存储池的数据压缩：

```
root # ceph osd pool set POOL_NAME ompression_algorithm snappy
root # ceph osd pool set POOL_NAME compression_mode aggressive
```

将 `POOL_NAME` 替换为要启用压缩的存储池。

7.5.2 存储池压缩选项

完整的压缩设置列表：

compression_algorithm

值：`none`、`zstd`、`snappy`。默认值：`snappy`。

使用的压缩算法取决于特定使用情形。下面是几点建议：

- 不要使用 `zlib`，其余几种算法更好。
- 如果需要较好的压缩率，请使用 `zstd`。注意，由于 `zstd` 在压缩少量数据时 CPU 开销较高，建议不要将其用于 BlueStore。
- 如果需要较低的 CPU 使用率，请使用 `lz4` 或 `snappy`。
- 针对实际数据的样本运行这些算法的基准测试，观察集群的 CPU 和内存使用率。

compression_mode

值：{`none`、`aggressive`、`passive`、`force`}。默认值：`none`。

- `none`：从不压缩
- `passive`：如果提示 `COMPRESSIBLE`，则压缩
- `aggressive`：除非提示 `INCOMPRESSIBLE`，才压缩
- `force`：始终压缩

有关如何设置 `COMPRESSIBLE` 或 `INCOMPRESSIBLE` 标志的信息，请参见 http://docs.ceph.com/docs/doc-12.2.0-major-changes/rados/api/librados/#rados_set_alloc_hint 。

compression_required_ratio

值：双精度型，比例 = SIZE_COMPRESSED / SIZE_ORIGINAL。默认值：`.875`

由于净增益低，存储高于此比例的对象时不会压缩。

compression_max_blob_size

值：无符号整数，大小以字节为单位。默认值：`0`

所压缩对象的最大大小。

compression_min_blob_size

值：无符号整数，大小以字节为单位。默认值：`0`

所压缩对象的最小大小。

7.5.3 全局压缩选项

可在 Ceph 配置中设置以下配置选项，并将其应用于所有 OSD 而不仅仅是单个存储池。第 7.5.2 节 “存储池压缩选项”中列出的存储池特定配置优先。

bluestore_compression_algorithm

值：`none`、`zstd`、`snappy`、`zlib`。默认值：`snappy`。

使用的压缩算法取决于特定使用情形。下面是几点建议：

- 不要使用 `zlib`，其余几种算法更好。
- 如果需要较好的压缩率，请使用 `zstd`。注意，由于 `zstd` 在压缩少量数据时 CPU 开销较高，建议不要将其用于 BlueStore。

- 如果需要较低的 CPU 使用率，请使用 `lz4` 或 `snappy`。
- 针对实际数据的样本运行这些算法的基准测试，观察集群的 CPU 和内存使用率。

bluestore_compression_mode

值：{`none`、`aggressive`、`passive`、`force`}。默认值：`none`。

- `none`：从不压缩
- `passive`：如果提示 `COMPRESSIBLE`，则压缩。
- `aggressive`：除非提示 `INCOMPRESSIBLE`，才压缩
- `force`：始终压缩

有关如何设置 `COMPRESSIBLE` 或 `INCOMPRESSIBLE` 标志的信息，请参见 http://docs.ceph.com/docs/doc-12.2.0-major-changes/rados/api/librados/#rados_set_alloc_hint ↗。

bluestore_compression_required_ratio

值：双精度型，比例 = SIZE_COMPRESSED / SIZE_ORIGINAL。默认值：`.875`

由于净增益低，存储高于此比例的对象时不会压缩。

bluestore_compression_min_blob_size

值：无符号整数，大小以字节为单位。默认值：`0`

所压缩对象的最小大小。

bluestore_compression_max_blob_size

值：无符号整数，大小以字节为单位。默认值：`0`

所压缩对象的最大大小。

bluestore_compression_min_blob_size_ssd

值：无符号整数，大小以字节为单位。默认值：`8K`

压缩并存储在固态硬盘上的对象的最小大小。

bluestore_compression_max_blob_size_ssd

值：无符号整数，大小以字节为单位。默认值：`64K`

压缩并存储在固态硬盘上的对象的最大大小。

bluestore_compression_min_blob_size_hdd

值：无符号整数，大小以字节为单位。默认值：`128K`

压缩并存储在普通硬盘上的对象的最小大小。

bluestore_compression_max_blob_size_hdd

值：无符号整数，大小以字节为单位。默认值：512K

压缩并存储在普通硬盘上的对象的最大大小。

8 RADOS 块设备

块是一个字节序列，例如 512 字节的数据块。基于块的存储接口是使用旋转媒体（例如硬盘、CD、软盘）存储数据最常见的方式。块设备接口的普及，也使得虚拟块设备成为与大量数据存储系统（例如 Ceph）进行交互的理想选择。

Ceph 块设备允许共享物理资源，并且可以调整大小。它们会在 Ceph 集群中的多个 OSD 上等量存储数据。Ceph 块设备会利用 RADOS 功能，例如创建快照、复制和一致性。Ceph 的 RADOS 块设备 (RBD) 使用内核模块或 `librbd` 库与 OSD 交互。

图 8.1： RADOS 协议

Ceph 的块设备为内核模块提供高性能及无限的可扩展性。它们支持虚拟化解决方案（例如 QEMU）或依赖于 `libvirt` 的基于云的计算系统（例如 OpenStack）。您可以使用同一个集群来同时操作对象网关、CephFS 和 RADOS 块设备。

8.1 块设备命令

`rbd` 命令可让您创建、列出、内省和删除块设备映像。您还可以使用它来执行其他操作，例如，克隆映像、创建快照、将映像回滚到快照或查看快照。

提示：访问集群

要使用 RADOS 块设备命令，您必须拥有对运行中 Ceph 集群的访问权限。

8.1.1 创建块设备映像

您必须先在集群中为块设备创建映像，然后才能将其添加到节点。要创建块设备映像，请执行以下命令：

```
root # rbd create --size megabytes pool-name/image-name
```

例如，要创建名为“bar”的 1GB 映像，并将信息存储在名为“swimmingpool”的存储池中，请执行以下命令：

```
root # rbd create --size 1024 swimmingpool/bar
```

提示：默认存储池

如果您在创建映像时不指定存储池，映像将存储在默认存储池“rbd”中。

注意：首先创建存储池

您需要先创建存储池，然后才能将它指定为来源。有关详细信息，请参见第 7 章 “管理存储池”。

8.1.2 在纠删码池中创建块设备映像

自 SUSE Enterprise Storage 5 起，可以将块设备映像的数据存储在纠删码池中。只能将 RBD 映像的“数据”部分存储在纠删码池中。另外，纠删码池必须将“overwrite”标志设置为 true，而只有在所有 OSD 都使用 BlueStore 的情况下，才能将此标志设置为 true。

映像元数据不能驻留在纠删码池中。元数据可以驻留在默认的“rbd”池中，或用户使用参数 `--pool=` 在 `rbd create` 命令中明确指定的存储池中。

注意：需要 BlueStore

所有节点都需要 BlueStore 才能使用纠删码池存储块设备映像。

使用以下步骤可在纠删码池中创建 RBD 映像：

```
root # ceph osd pool create POOL_NAME 12 12 erasure
root # ceph osd pool set POOL_NAME allow_ec_overwrites true

# Metadata will reside in pool "rbd", and data in pool "POOL_NAME"
root # rbd create IMAGE_NAME --size=1G --data-pool POOL_NAME
```

```
#Metadata will reside in pool "OTHER_POOL", and data in pool "POOL_NAME"
root # rbd create IMAGE_NAME --size=1G --data-pool POOL_NAME --pool=OTHER_POOL
```

8.1.3 列出块设备映像

要列出“rbd”池中的块设备，请执行以下命令（“rbd”是默认池名称）：

```
root # rbd ls
```

要列出名为“swimmingpool”的存储池中的块设备，请执行以下命令：

```
root # rbd ls swimmingpool
```

8.1.4 检索映像信息

要检索名为“swimmingpool”的存储池内映像“bar”中的信息，请运行以下命令：

```
root # rbd info swimmingpool/bar
```

8.1.5 调整块设备映像的大小

RADOS 块设备映像是瘦配置 — 在您开始将数据保存到这些映像之前，它们实际上并不会使用任何物理存储。但是，这些映像具有您使用 `--size` 选项设置的最大容量。如果您要增大（或减小）映像的最大大小，请运行以下命令：

```
root # rbd resize --size 2048 foo # to increase
rbd resize --size 2048 foo --allow-shrink # to decrease
```

8.1.6 删除块设备映像

要删除与名为“swimmingpool”的存储池内映像“bar”对应的块设备，请运行以下命令：

```
root # rbd rm swimmingpool/bar
```

8.2 挂载和卸载 RBD 映像

创建 RADOS 块设备之后，便可以格式化并挂载它以便能够交换文件，然后在完成时将其卸载。

1. 确保您的 Ceph 集群包括要挂载的磁盘映像所在的存储池。假设存储池名为 `mypool`，映像名为 `myimage`。

```
rbd list mypool
```

2. 将映像映射到新的块设备。

```
root # rbd map --pool mypool myimage
```

提示：用户名和身份验证

要指定用户名，请使用 `--id 用户名`。此外，如果您使用了 `cephx` 身份验证，则还必须指定机密。该机密可能来自密钥环，或某个包含机密的文件：

```
root # rbd map --pool rbd myimage --id admin --keyring /path/to/
keyring
```

或者

```
root # rbd map --pool rbd myimage --id admin --keyfile /path/to/file
```

3. 列出所有映射的设备：

```
root # rbd showmapped
 id pool   image   snap device
 0  mypool myimage -    /dev/rbd0
```

我们要使用的设备是 `/dev/rbd0`。

4. 在 `/dev/rbd0` 设备上创建 XFS 文件系统。

```
root # mkfs.xfs /dev/rbd0
 log stripe unit (4194304 bytes) is too large (maximum is 256KiB)
```

```
log stripe unit adjusted to 32KiB
meta-data=/dev/rbd0              isize=256    agcount=9, agsize=261120
blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=2097152, imaxpct=25
         =                       sunit=1024   swidth=1024 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=8 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

5. 挂载设备并检查它是否已正确挂载。将 `/mnt` 替换为您的挂载点。

```
root # mount /dev/rbd0 /mnt
root # mount | grep rbd0
/dev/rbd0 on /mnt type xfs (rw,relatime,attr2,inode64,sunit=8192,...
```

现在，您便可以将数据移入/移出设备，就如同它是本地目录一样。

提示：增大 RBD 设备的大小

如果您发现 RBD 设备的大小不再够用，可以轻松增大大小。

1. 增大 RBD 映像的大小，例如增大到 10GB。

```
root # rbd resize --size 10000 mypool/myimage
 Resizing image: 100% complete...done.
```

2. 扩大文件系统以填入设备的新大小。

```
root # xfs_growfs /mnt
 [...]
 data blocks changed from 2097152 to 2560000
```

6. 当您访问过设备后，可以将其卸载。

```
root # unmount /mnt
```

提示：手动挂载（卸载）

因为在引导之后手动映射和挂载 RBD 映像以及在关机之前卸载和取消映射会非常麻烦，我们提供了 `rbdmap` 脚本和 `systemd` 单元。请参考第 8.4 节 “rbdmap：在引导时映射 RBD 设备”。

8.3 块设备快照

RBD 快照是 RADOS 块设备映像的快照。通过快照，您可以保留映像状态的历史。Ceph 还支持快照分层，这可让您轻松快速地克隆 VM 映像。Ceph 使用 `rbd` 命令和许多高级接口（包括 QEMU、`libvirt`、OpenStack 和 CloudStack）支持块设备快照。

注意

在创建映像快照之前，请停止输入和输出操作。如果映像包含文件系统，则在创建快照之前，文件系统必须处于一致状态。

8.3.1 Cephx 注意事项

如果 `cephx` 处于启用状态（有关更多信息，请参见http://ceph.com/docs/master/rados/configuration/auth-config-ref/ ），您必须指定用户名或 ID 以及包含用户的相应密钥的密钥环路径。有关更多详细信息，请参见用户管理 (http://ceph.com/docs/master/rados/operations/user-management/) 。您还可以添加 `CEPH_ARGS` 环境变量，以免重新输入以下参数。

```
root # rbd --id user-ID --keyring=/path/to/secret commands
root # rbd --name username --keyring=/path/to/secret commands
```

例如：

```
root # rbd --id admin --keyring=/etc/ceph/ceph.keyring commands
root # rbd --name client.admin --keyring=/etc/ceph/ceph.keyring commands
```

提示

将用户和机密添加到 `CEPH_ARGS` 环境变量，如此您便无需每次都输入它们。

8.3.2 快照基础知识

下面的过程说明如何在命令行上使用 `rbd` 命令创建、列出和删除快照。

8.3.2.1 创建快照

要使用 `rbd` 创建快照，请指定 `snap create` 选项、存储池名称和映像名称。

```
root # rbd --pool pool-name snap create --snap snap-name image-name
root # rbd snap create pool-name/image-name@snap-name
```

例如：

```
root # rbd --pool rbd snap create --snap snapshot1 image1
root # rbd snap create rbd/image1@snapshot1
```

8.3.2.2 列出快照

要列出映像的快照，请指定存储池名称和映像名称。

```
root # rbd --pool pool-name snap ls image-name
root # rbd snap ls pool-name/image-name
```

例如：

```
root # rbd --pool rbd snap ls image1
root # rbd snap ls rbd/image1
```

8.3.2.3 回滚快照

要使用 `rbd` 回滚快照，请指定 `snap rollback` 选项、存储池名称、映像名称和快照名称。

```
root # rbd --pool pool-name snap rollback --snap snap-name image-name
root # rbd snap rollback pool-name/image-name@snap-name
```

例如：

```
root # rbd --pool pool1 snap rollback --snap snapshot1 image1
root # rbd snap rollback pool1/image1@snapshot1
```

注意

将映像回滚到快照意味着会使用快照中的数据重写当前版本的映像。执行回滚所需的时间将随映像大小的增加而延长。从快照克隆较快，而从映像到快照的回滚较慢，因此克隆是返回先前存在状态的首选方法。

8.3.2.4 删除快照

要使用 `rbd` 删除快照，请指定 `snap rm` 选项、存储池名称、映像名称和用户名。

```
root # rbd --pool pool-name snap rm --snap snap-name image-name
root # rbd snap rm pool-name/image-name@snap-name
```

例如：

```
root # rbd --pool pool1 snap rm --snap snapshot1 image1
root # rbd snap rm pool1/image1@snapshot1
```

注意

Ceph OSD 会以异步方式删除数据，因此删除快照不能立即释放磁盘空间。

8.3.2.5 清除快照

要使用 `rbd` 删除映像的所有快照，请指定 `snap purge` 选项和映像名称。

```
root # rbd --pool pool-name snap purge image-name
```

```
root # rbd snap purge pool-name/image-name
```

例如：

```
root # rbd --pool pool1 snap purge image1
root # rbd snap purge pool1/image1
```

8.3.3 分层

Ceph 支持创建许多块设备快照的写入时复制 (COW) 克隆的能力。快照分层可让 Ceph 块设备客户端能够极快地创建映像。例如，您可以创建块设备映像并将 Linux VM 写入其中，然后创建映像的快照、保护快照，并创建您所需数量的“写入时复制”克隆。快照是只读的，因此克隆快照简化了语义，如此可快速创建克隆。

注意

下面的命令行示例中提到的“父”和“子”这两个术语是指 Ceph 块设备快照（父）和从快照克隆的相应映像（子）。

每个克隆的映像（子）都存储了对其父映像的引用，这可让克隆的映像打开父快照并读取其内容。

快照的 COW 克隆的行为方式与任何其他 Ceph 块设备映像完全相同。可针对克隆的映像执行读取、写入、克隆和调整大小操作。系统对克隆的映像没有特殊限制。但是，快照的写入时复制克隆会引用快照，因此您必须在克隆快照之前保护快照。

注意

Ceph 只支持克隆格式 2 映像（即使用 `rbd create --image-format 2` 创建的映像）。

8.3.3.1 分层入门

Ceph 块设备分层是一个简单的过程。您必须有一个映像。您必须创建映像的快照。您必须保护快照。在您执行这些步骤之后，就可以开始克隆快照了。

克隆的映像具有对父快照的引用，并且包含存储池 ID、映像 ID 和快照 ID。包含存储池 ID 意味着您可以将快照从一个存储池克隆到另一个存储池中的映像。

- 映像模板：一种常见的块设备分层用例是创建主映像和用作克隆模板的快照。例如，用户可为 Linux 发行套件（如 SUSE Linux Enterprise Server）创建映像并为它创建快照。用户可以定期更新映像和创建新的快照（例如，先执行 `zypper ref && zypper patch`，接着执行 `rbd snap create`）。随着映像日趋成熟，用户可以克隆任何一个快照。
- 扩展模板：更高级的用例包括扩展比基本映像提供的信息更多的模板映像。例如，用户可以克隆映像（VM 模板）并安装其他软件（例如，数据库、内容管理系统或分析系统），然后创建扩展映像的快照，这个扩展映像可以如基本映像一样更新。
- 模板池：使用块设备分层的一种方法是创建包含主映像（用作模板）的池，然后创建这些模板的快照。之后，您便可以扩大用户的只读特权，使他们可以克隆快照，却不能写入存储池或在存储池中执行。
- 映像迁移/恢复：使用块设备分层的一种方法是将数据从一个存储池迁移或恢复到另一个存储池。

8.3.3.2 保护快照

克隆会访问父快照。如果用户意外删除了父快照，则所有克隆都会损坏。为了防止数据丢失，您需要先保护快照，然后才能克隆它。

```
root # rbd --pool pool-name snap protect \
 --image image-name --snap snapshot-name
root # rbd snap protect pool-name/image-name@snapshot-name
```

例如：

```
root # rbd --pool pool1 snap protect --image image1 --snap snapshot1
root # rbd snap protect pool1/image1@snapshot1
```

注意

您无法删除受保护的快照。

8.3.3.3 克隆快照

要克隆快照，您需要指定父存储池、映像、快照、子存储池和映像名称。您需要先保护快照，然后才能克隆它。

```
root # rbd --pool pool-name --image parent-image \
 --snap snap-name --dest-pool pool-name \
 --dest child-image
root # rbd clone pool-name/parent-image@snap-name \
pool-name/child-image-name
```

例如：

```
root # rbd clone pool1/image1@snapshot1 pool1/image2
```

注意

您可以将快照从一个存储池克隆到另一个存储池中的映像。例如，可以在一个存储池中将只读映像和快照作为模板维护，而在另一个存储池中维护可写入克隆。

8.3.3.4 取消保护快照

必须先取消保护快照，然后才能删除它。另外，您无法删除克隆所引用的快照。您需要先平展快照的每个克隆，然后才能删除快照。

```
root # rbd --pool pool-name snap unprotect --image image-name \
 --snap snapshot-name
root # rbd snap unprotect pool-name/image-name@snapshot-name
```

例如：

```
root # rbd --pool pool1 snap unprotect --image image1 --snap snapshot1
root # rbd snap unprotect pool1/image1@snapshot1
```

8.3.3.5 列出快照的子项

要列出快照的子项，请执行以下命令：

```
root # rbd --pool pool-name children --image image-name --snap snap-name
root # rbd children pool-name/image-name@snapshot-name
```

例如：

```
root # rbd --pool pool1 children --image image1 --snap snapshot1
root # rbd children pool1/image1@snapshot1
```

8.3.3.6 平展克隆的映像

克隆的映像会保留对父快照的引用。删除子克隆对父快照的引用时，可通过将信息从快照复制到克隆，高效“平展”映像。平展克隆所需的时间随着映像大小的增加而延长。要删除快照，必须先平展子映像。

```
root # rbd --pool pool-name flatten --image image-name
root # rbd flatten pool-name/image-name
```

例如：

```
root # rbd --pool pool1 flatten --image image1
root # rbd flatten pool1/image1
```

注意

由于平展的映像包含快照中的所有信息，平展的映像占用的存储空间将比分层克隆多。

8.4 rbdmap：在引导时映射 RBD 设备

`rbdmap` 是一个外壳脚本，可针对一个或多个 RADOS 块设备映像自动执行 `rbd map` 和 `rbd unmap` 操作。虽然您随时都可以手动运行该脚本，但其主要用来在引导时自动映射和挂载 RBD 映像（以及在关机时卸载和取消映射），此操作由 Init 系统触发。`ceph-common` 包中随附了一个 `systemd` 单元文件 `rbdmap.service` 用于执行此操作。

该脚本使用单个自变量，可以是 `map` 或 `unmap`。使用任一自变量时，该脚本都会分析配置文件。它默认为 `/etc/ceph/rbdmap`，但可通过环境变量 `rbdmapFILE` 覆盖。该配置文件的每一行相当于一个要映射或取消映射的 RBD 映像。

配置文件采用以下格式：

```
image_specification rbd_options
```

image_specification

存储池中映像的路径。以 `pool_name`/`image_name` 格式指定。如果您省略 `pool_name`，则假设使用默认名称“rbd”。

rbd_options

要传递给底层 `rbd map` 命令的参数的可选列表。这些参数及其值应该以逗号分隔的字符串形式指定，例如：

```
PARAM1=VAL1,PARAM2=VAL2,...
```

该示例让 `rbdmap` 脚本运行以下命令：

```
rbd map pool_name/image_name --PARAM1 VAL1 --PARAM2 VAL2
```

以 `rbdmap map` 形式运行时，该脚本会分析配置文件，并且对于每个指定的 RBD 映像，它会尝试先映射映像（使用 `rbd map` 命令），再挂载映像。

以 `rbdmap unmap` 形式运行时，配置文件中列出的映像将卸载并取消映射。

`rbdmap unmap-all` 会尝试卸载然后取消映射所有当前已映射的 RBD 映像，而不论它们是否列在配置文件中。

如果成功，rbd map 操作会将映像映射到 /dev/rbdX 设备，此时会触发 udev 规则，以创建易记设备名称符号链接 `/dev/rbd/pool_name/image_name`，该链接指向实际映射的设备。

为了挂载和卸载成功，易记设备名称在 `/etc/fstab` 中需有对应项。写入 RBD 映像的 `/etc/fstab` 项时，指定“noauto”（或“nofail”）挂载选项。这可防止 Init 系统过早（尚未出现有问题的设备时）尝试挂载设备，因为 `rbdmap.service` 通常是在引导序列中相当靠后的时间触发。

有关 `rbd` 选项的完整列表，请参见 `rbd` 手册页 (`man 8 rbd`)。

有关 `rbdmap` 用法的示例，请参见 `rbdmap` 手册页 (`man 8 rbdmap`)。

8.5 RADOS 块设备镜像

RBD 映像可以在两个 Ceph 集群之间异步镜像。此功能使用 RBD 日记映像功能来确保集群之间的复制在崩溃时保持一致。镜像在对等集群中逐池配置，并且可以配置为自动镜像池中的所有映像或仅镜像特定的映像子集。镜像使用 `rbd` 命令进行配置。`rbd-mirror` 守护进程负责从远程对等集群提取映像更新，并将它们应用于本地集群中的映像。

重要：rbd-mirror 守护进程

要使用 RBD 镜像，您需要有两个 Ceph 集群，每个集群都运行 `rbd-mirror` 守护进程。

8.5.1 rbd-mirror 守护进程

两个 `rbd-mirror` 守护进程负责检查远程对等集群上的映像日记并针对本地集群重放日记事件。RBD 映像日记功能会按发生的顺序记录对映像进行的所有修改。如此可确保远程映像崩溃状态一致的镜像可在本地使用。

`rbd-mirror` 守护进程在 `rbd-mirror` 包中提供。在其中一个集群节点上安装、启用并启动它：

```
root@minion > zypper install rbd-mirror
root@minion > systemctl enable ceph-rbd-mirror@server_name.service
root@minion > systemctl start ceph-rbd-mirror@server_name.service
```

重要

每个 `rbd-mirror` 守护进程都必须能够同时连接到两个集群。

8.5.2 存储池配置

以下过程说明如何使用 `rbd` 命令来执行配置镜像的基本管理任务。镜像在 Ceph 集群中逐池进行配置。

您需要在两个对等集群上执行存储池配置步骤。为清楚起见，这些过程假设名为“local”和“remote”的两个集群可从单台主机访问。

有关如何连接到不同的 Ceph 集群的更多详细信息，请参见 `rbd` 手册页 (`man 8 rbd`)。

提示：多个集群

在下面的示例中，集群名称与同名的 Ceph 配置文件 `/etc/ceph/remote.conf` 相对应。有关如何配置多个集群的信息，请参见 ceph-conf (http://docs.ceph.com/docs/master/rados/configuration/ceph-conf/#running-multiple-clusters) 文档。

8.5.2.1 启用镜像

要针对存储池启用镜像，请指定 `mirror pool enable` 子命令、存储池名称和镜像模式。镜像模式可以是存储池或映像：

pool

将会镜像启用了日记功能的存储池中的所有映像。

image

需要针对每个映像明确启用镜像。有关详细信息，请参见第 8.5.3.2 节 “启用映像镜像”。

例如：

```
root # rbd --cluster local mirror pool enable image-pool pool
root # rbd --cluster remote mirror pool enable image-pool pool
```

8.5.2.2 禁用镜像

要对存储池禁用镜像，请指定 `mirror pool disable` 子命令和存储池名称。使用这种方法对存储池禁用镜像时，还会对已为其明确启用镜像的所有映像（该存储池中）禁用镜像。

```
root # rbd --cluster local mirror pool disable image-pool
root # rbd --cluster remote mirror pool disable image-pool
```

8.5.2.3 添加集群对等

为了让 `rbd-mirror` 守护进程发现它的对等集群，需要向存储池注册该对等集群。要添加镜像对等集群，请指定 `mirror pool peer add` 子命令、存储池名称和集群规格：

```
root # rbd --cluster local mirror pool peer add image-pool client.remote@remote
root # rbd --cluster remote mirror pool peer add image-pool client.local@local
```

8.5.2.4 删除集群对等

要删除镜像对等集群，请指定 `mirror pool peer remove` 子命令、存储池名称和对等 UUID（可通过 `rbd mirror pool info` 命令获得）：

```
root # rbd --cluster local mirror pool peer remove image-pool \
 55672766-c02b-4729-8567-f13a66893445
root # rbd --cluster remote mirror pool peer remove image-pool \
 60c0e299-b38f-4234-91f6-eed0a367be08
```

8.5.3 映像配置

与存储池配置不同，映像配置只需要针对单个镜像对等 Ceph 集群执行。

系统会将镜像的 RBD 映像指定为主要或非主要。这是映像的属性，而不是存储池的属性。不能修改指定为非主要的映像。

当首次对某个映像启用镜像时（如果存储池镜像模式是“存储池”并且映像已启用日记映像功能，则为隐式启用，或可通过 `rbd` 命令显式启用（请参见第 8.5.3.2 节 “启用映像镜像”）），映像会自动升级为主要映像。

8.5.3.1 启用映像日记支持

RBD 镜像使用 RBD 日记功能来确保复制的映像始终在崩溃时保持一致状态。在将映像镜像到对等集群之前，必须启用日记功能。可以在创建映像时通过将 `--image-feature exclusive-lock,journaling` 选项提供给 `rbd` 命令来启用该功能。

或者，日志功能可以针对预先存在的 RBD 映像动态启用。要启用日记，请指定 `feature enable` 子命令、存储池和映像名称以及功能名称：

```
root # rbd --cluster local feature enable image-pool/image-1 journaling
```

注意：选项依赖性

`journaling` 功能依赖于 `exclusive-lock` 功能。如果 `exclusive-lock` 功能尚未启用，则您需要先启用它，再启用 `journaling` 功能。

提示：针对所有新映像启用日记

默认情况下，可通过将下面一行添加到 Ceph 配置文件来针对所有新映像启用日记：

```
rbd default features = 125
```

8.5.3.2 启用映像镜像

如果以"映像"模式对映像的存储池配置镜像，则需要为存储池中的每个映像明确启用镜像。要为特定映像启用镜像，请指定 `mirror image enable` 子命令以及存储池和映像名称：

```
root # rbd --cluster local mirror image enable image-pool/image-1
```

8.5.3.3 禁用映像镜像

要为特定映像禁用镜像，请指定 `mirror image disable` 子命令以及存储池和映像名称：

```
root # rbd --cluster local mirror image disable image-pool/image-1
```

8.5.3.4 映像升级和降级

在需要将主要指定移动到对等集群中映像的故障转移情况下，您需要停止访问主要映像、降级当前主要映像、升级新的主要映像，然后继续访问替代集群上的映像。

要将特定映像降级为非主要映像，请指定 `mirror image demote` 子命令以及存储池和映像名称：

```
root # rbd --cluster local mirror image demote image-pool/image-1
```

要将存储池中的所有主要映像都降级为非主要映像，请指定 `mirror pool demote` 子命令以及存储池名称：

```
root # rbd --cluster local mirror pool demote image-pool
```

要将特定映像升级为主要映像，请指定 `mirror image promote` 子命令以及存储池和映像名称：

```
root # rbd --cluster remote mirror image promote image-pool/image-1
```

要将存储池中的所有非主要映像都升级为主要映像，请指定 `mirror pool promote` 子命令以及存储池名称：

```
root # rbd --cluster local mirror pool promote image-pool
```

提示：拆分 I/O 负载

因为主要或非主要状态都是针对每个映像指定的，所以可以将两个集群拆分为 IO 负载和阶段故障转移或故障回复。

注意：强制升级

可以使用 `--force` 选项强制升级。降级不能传播到对等集群时（例如，当集群发生故障或通讯中断时），就需要强制升级。这将导致两个对等集群之间出现节点分裂情况，并且映像不再同步，直到发出了 `resync` 子命令。

8.5.3.5 强制映像重新同步

如果 `rbd-mirror` 守护进程检测到分区事件，则在该情况解决之前，它不会尝试镜像受影响的映像。要继续镜像映像，请先降级确定过期的映像，然后请求与主要映像重新同步。要请求映像重新同步，请指定 `mirror image resync` 子命令以及存储池和映像名称：

```
root # rbd mirror image resync image-pool/image-1
```

8.5.4 镜像状态

系统会存储每个主要镜像映像的对等集群复制状态。此状态可使用 `mirror image status` 和 `mirror pool status` 子命令检索：

要请求镜像映像状态，请指定 `mirror image status` 子命令以及存储池和映像名称：

```
root # rbd mirror image status image-pool/image-1
```

要请求镜像存储池摘要状态，请指定 `mirror pool status` 子命令以及存储池名称：

```
root # rbd mirror pool status image-pool
```

提示：

将 `--verbose` 选项添加到 `mirror pool status` 子命令会额外地输出存储池中每个镜像映像的状态详细信息。

9 纠删码池

Ceph 提供了一种在存储池中正常复制数据的替代方案，称为纠删池或纠删码池。纠删池不能提供副本池的所有功能，但所需的原始存储空间更少。能够存储 1 TB 数据的默认纠删池需要 1.5 TB 原始存储空间。从这方面而言比副本池更好，因为后者需要 2 TB 的原始存储空间才能存储相同的数据量。

有关纠删码的背景信息，请参见 https://en.wikipedia.org/wiki/Erasure_code 。

注意

使用 FileStore 时，除非已配置快速缓存层，否则无法通过 RBD 接口访问纠删码池。有关详细信息或如何使用 Bluestore，请参见第 9.3 节 “纠删码池和快速缓存层”。

注意

确保纠删池的 CRUSH 规则对 `step` 使用 `indep` 。有关详细信息，请参见第 6.3.2 节 “firstn 和 indep”。

9.1 创建示例纠删码池

最简单的纠删码池相当于 RAID5，至少需要三个主机。以下过程介绍如何创建用于测试的存储池。

1. 命令 `ceph osd pool create` 用于创建类型为纠删的池。`12` 表示归置组的数量。使用默认参数时，该存储池能够处理一个 OSD 的故障。

   ```
   root # ceph osd pool create ecpool 12 12 erasure
   pool 'ecpool' created
   ```

2. 字符串 `ABCDEFGHI` 将写入名为 `NYAN` 的对象。

   ```
   cephadm > echo ABCDEFGHI | rados --pool ecpool put NYAN -
   ```

3. 为了进行测试，现在可以禁用 OSD，例如，断开其网络连接。

4. 要测试该存储池是否可以处理多台设备发生故障的情况，可以使用 `rados` 命令来访问文件的内容。

```
root # rados --pool ecpool get NYAN -
ABCDEFGHI
```

9.2 纠删码配置

调用 `ceph osd pool create` 命令来创建纠删池时，除非指定了其他配置，否则会使用默认的配置。配置定义数据冗余。要进行这种定义，可以设置随意命名为 `k` 和 `m` 的两个参数。k 和 m 定义要将数据片段拆分成多少个 `块`，以及要创建多少个编码块。然后，冗余块将存储在不同的 OSD 上。

纠删池配置所需的定义：

chunk
: 如果调用该编码函数，它会返回相同大小的块：可串联起来以重构造原始对象的数据块，以及可用于重构建丢失的块的编码块。

k
: 数据块的数量，即要将原始对象分割成的块数量。例如，如果 `k = 2`，则会将一个 10KB 对象分割成 `k` 个各为 5KB 的对象。

m
: 编码块的数量，即编码函数计算的额外块的数量。如果有 2 个编码块，则表示可以移出 2 个 OSD，而不会丢失数据。

crush-failure-domain
: 定义要将块分布到的设备。其值需要设置为某个桶类型。有关所有的桶类型，请参见第 6.2 节 “桶”。如果故障域为 `机柜`，则会将块存储在不同的机柜上，以提高机柜发生故障时的恢复能力。

借助第 9.1 节 “创建示例纠删码池”中所用的默认纠删码配置，当单个 OSD 发生故障时，您将不会丢失集群数据。因此，要存储 1 TB 数据，需要额外提供 0.5 TB 原始存储空间。这意味着，1 TB 数据需要 1.5 TB 原始存储空间。这相当于常见的 RAID 5 配置。相比之下，副本池需要 2 TB 原始存储空间来存储 1 TB 数据。

可使用以下命令显示默认配置的设置：

```
root # ceph osd erasure-code-profile get default
directory=.libs
k=2
m=1
plugin=jerasure
crush-failure-domain=host
technique=reed_sol_van
```

选择适当的配置非常重要，因为在创建存储池后便无法修改配置。需要创建使用不同配置的新存储池，并将之前的存储池中的所有对象移到新存储池。

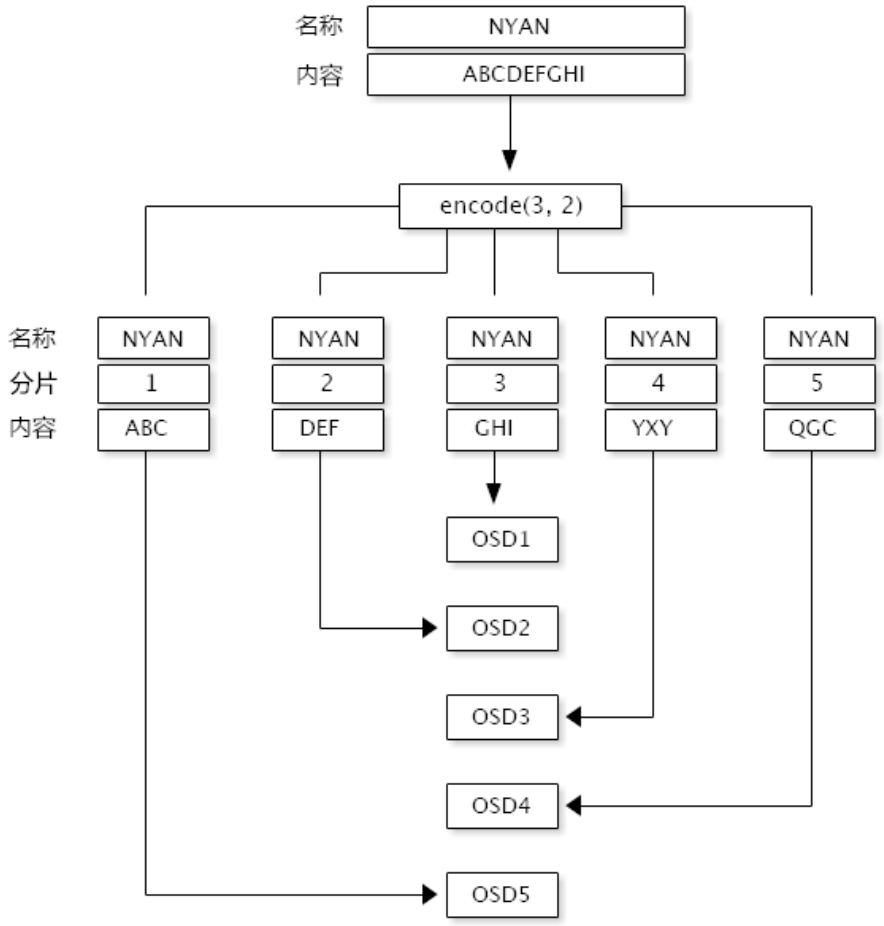
最重要的几个配置参数是 `k`、`m` 和 `crush-failure-domain`，因为它们定义存储开销和数据持久性。例如，如果在丢失两个机柜并且存储开销达到开销的 66% 时，必须能够维系所需的体系结构，您可定义以下配置：

```
root # ceph osd erasure-code-profile set myprofile \
   k=3 \
   m=2 \
   crush-failure-domain=rack
```

对于此新配置，可以重复第 9.1 节 “创建示例纠删码池”中的示例：

```
root # ceph osd pool create ecpool 12 12 erasure myprofile
cephadm > echo ABCDEFGHI | rados --pool ecpool put NYAN -
root # rados --pool ecpool get NYAN -
ABCDEFGHI
```

NYAN 对象将分割成三个 (`k=3`)，并将创建两个额外的块 (`m=2`)。`m` 值定义可以同时丢失多少个 OSD 而不会丢失任何数据。`crush-failure-domain=rack` 将创建一个 CRUSH 规则组，用于确保不会将两个块存储在同一个机柜中。



有关纠删码配置的详细信息，请参见 http://docs.ceph.com/docs/master/rados/operations/erasure-code-profile 。

9.3 纠删码池和快速缓存层

纠删码池所需的资源比副本池更多，并且缺少某些功能，例如部分写入。要克服这些限制，建议在纠删码池的前面设置一个快速缓存层。

例如，如果“热存储”池由高速存储设备构成，则可使用以下命令来加速第 9.2 节 “纠删码配置”中创建的“ecpool”：

```
root # ceph osd tier add ecpool hot-storage
root # ceph osd tier cache-mode hot-storage writeback
```

```
root # ceph osd tier set-overlay ecpool hot-storage
```

这会将用作 ecpool 层的“热存储”池置于写回模式，以便每次在 ecpool 中写入和读取时实际使用的都是热存储，并受益于热存储的灵活性和速度。

使用 FileStore 时，无法在纠删码池中创建 RBD 映像，因为此操作需要部分写入。但是，如果将副本池层设置为快速缓存层，则可以在纠删码池中创建 RBD 映像：

```
root # rbd --pool ecpool create --size 10 myvolume
```

有关快速缓存层的详细信息，请参见第 10 章 “快速缓存分层”。

9.4 含 RADOS 块设备的纠删码池

要将 EC 池标记为 RBD 池，请对其进行相应标记：

```
root # ceph osd pool application enable rbd ec_pool_name
```

RBD 可在 EC 池中存储映像数据。但是，映像报头和元数据仍需要存储在副本池中。为此，假设您的存储池命名为“rbd”：

```
root # rbd create rbd/image_name --size 1T --data-pool ec_pool_name
```

您可以像使用任何其他映像一样正常使用该映像，只不过所有数据都将存储在 `ec_pool_name` 池而非“rbd”池中。

10 快速缓存分层

快速缓存层是在客户端与标准存储之间实施的附加存储层。该层用于加快访问低速硬盘中存储的存储池以及纠删码池的速度。

通常，快速缓存分层涉及到创建一个配置为充当快速缓存层的相对快速且昂贵的存储设备（例如 SSD 驱动器）池，以及一个配置为充当存储层的较慢但较便宜的设备后备池。

10.1 分层存储的相关术语

快速缓存分层识别两种类型的池：快速缓存池和存储池。

提示

有关池的一般信息，请参见第 7 章 “管理存储池”。

存储池

在 Ceph 存储集群中存储一个对象的多个副本的标准副本池，或纠删码池（请参见第 9 章 “纠删码池”）。

存储池有时称为“后备”存储或“冷”存储。

快速缓存池

标准副本池，存储在相对较小但速度较快的存储设备上，在 CRUSH 地图中具有自己的规则组。

快速缓存池也称为“热”存储。

10.2 需考虑的要点

快速缓存分层可能会降低特定工作负载的集群性能。以下几点指出了您需要考虑的有关快速缓存分层的几个方面：

- 取决于工作负载：快速缓存能否提高性能取决于工作负载。由于将对象移入或移出快速缓存会耗费资源，因此，如果大多数请求只涉及到少量的对象，则使用快速缓存可能更高效。快速缓存池的大小应该足以接收工作负载的工作集，以避免性能大幅波动。
- 难以进行基准测试：大多数性能基准测试可能会反映使用快速缓存分层时性能会较低。原因在于，这些基准测试请求了大量的对象，而快速缓存的“预热”需要较长时间。
- 性能可能较低：对于不适合进行快速缓存分层的工作负载而言，其性能往往比不启用快速缓存分层的普通副本池更低。
- `librados` 对象枚举：如果应用直接使用 `librados` 并依赖于对象枚举，则快速缓存分层可能无法按预期工作（对于对象网关、RBD 或 CephFS 而言，这不会造成问题）。

10.3 何时使用快速缓存分层

在以下情况下，请考虑使用快速缓存分层：

- 需要通过 RADOS 块设备 (RBD) 访问纠删码池。
- 需要通过 iSCSI（因为它沿袭了 RBD 的限制）访问纠删码池。有关 iSCSI 的详细信息，请参见第 12 章 “Ceph iSCSI 网关”。
- 您的高性能存储数量有限，而低性能存储众多，您需要更快地访问存储的数据。

10.4 快速缓存模式

快速缓存分层代理可处理快速缓存层与后备存储层之间的数据迁移。管理员可以配置如何进行这种迁移。主要有两种方案：

写回模式

在写回模式下，Ceph 客户端将数据写入快速缓存层，并从快速缓存层接收确认响应。一段时间后，写入快速缓存层的数据将迁移到存储层，并从快速缓存层中清除。从概念上讲，快速缓存层叠加在后备存储层的“前面”。当 Ceph 客户端需要驻留在存储层中的数据时，快速缓存分层代理会在读取时将数据迁移到快速缓存层，然后，数据将发送到 Ceph 客户端。因此，在数据变为不活动状态前，Ceph 客户端可以使用快速缓存层执行 I/O。这种做法非常适合可变的数据，例如，编辑的照片或视频，或事务数据。

只读模式

在只读模式下，Ceph 客户端将数据直接写入后备层。在读取时，Ceph 将请求的对象从后备层复制到快速缓存层。过时对象将会根据定义的策略从快速缓存层中删除。这种做法非常适合不可变的数据，例如，在社交网络上呈现的图片或视频、DNA 数据或 X 光成像，因为从可能包含过时数据的快速缓存池中读取数据会导致一致性很差。不要对可变的数据使用只读模式。

10.5 命中集

10.5.1 概述

使用命中集参数可以优化快速缓存池。Ceph 中的命中集通常是布隆过滤器，提供节省内存用量的方式来跟踪已存放于快速缓存池的对象。

命中集是一个位数组，用于存储针对对象名称应用的一组哈希函数的结果。最初，所有的位都设为 `0`。将对象添加到命中集后，该对象的名称将经过哈希处理，结果将映射在命中集中的不同位置，那时，位的值便会设置为 `1`。

为了确定某个对象是否存在于快速缓存中，将会再次对对象名称进行哈希处理。如果有任何位是 `0`，则表示该对象肯定不在快速缓存中，需要从冷存储中检索它。

不同对象的结果可能会存储在命中集的同一位置。在巧合的情况下，可能所有位都是 `1`，而对象却不在快速缓存中。因此，处理布隆过滤器的命中集只能确定某个对象是否一定不在快速缓存中，需要从冷存储检索它。

一个快速缓存池可以使用多个命中集来跟踪各时间段的文件访问。设置 `hit_set_count` 定义所用的命中集数量，`hit_set_period` 定义每个命中集已使用了多长时间。该期限过期后，将使用下一个命中集。如果用尽了命中集，将会释放最旧命中集的内存，并创建新的命中集。将 `hit_set_count` 和 `hit_set_period` 的值相乘可定义已跟踪对象访问的整个时间范围。

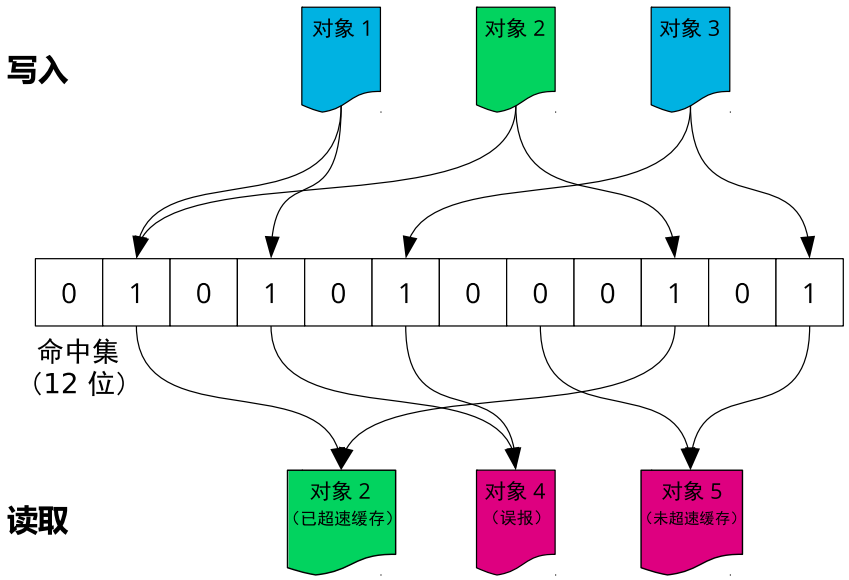


图 10.1： 包含 3 个存储对象的布隆过滤器

与哈希处理对象的数量相比，基于布隆过滤器的命中集非常地节省内存用量。只需使用不到 10 个位即可将误报率降低到 1% 以下。可以使用 `hit_set_fpp` 定义误报率。Ceph 可根据归置组中的对象数量及误报率自动计算命中集的大小。

可以使用 `min_write_recency_for_promote` 和 `min_read_recency_for_promote` 限制快速缓存池中所需的存储。如果将值设置为 `0`，则所有对象在被读取或写入后，会立即提升到快速缓存池，并且在被逐出之前会一直保持这种模式。使用大于 `0` 的任何值可定义在其中搜索对象的命中集（已按期限排序）的数量。如果在某个命中集中找到了该对象，会将该对象提升到快速缓存池。

10.5.2 示例

10.5.2.1 大型快速缓存池和少量内存

如果有大量的存储，但只有少量的 RAM 可用，则所有对象在被访问后可立即提升到快速缓存池。命中集保持较小的规模。下面是一组示例配置值：

```
hit_set_count = 1
hit_set_period = 3600
hit_set_fpp = 0.05
```

```
min_write_recency_for_promote = 0
min_read_recency_for_promote = 0
```

10.5.2.2 小型快速缓存池和大量内存

如果只有少量的存储，但可用的内存量相对较大，则可以将快速缓存层配置为将有限数量的对象提升到快速缓存池。如果有 12 个命中集，并且在 14,400 秒期限内可以使用其中每个命中集，则这些命中集总共可提供 48 小时的跟踪。如果在过去 8 小时内访问了某个对象，该对象将提升到快速缓存池。这种情况的一组示例配置值如下：

```
hit_set_count = 12
hit_set_period = 14400
hit_set_fpp = 0.01
min_write_recency_for_promote = 2
min_read_recency_for_promote = 2
```

10.6 设置示例分层存储

本节举例说明如何在标准硬盘（冷存储）的前面设置一个高速 SSD 快速缓存层（热存储）。

提示

下面的示例仅用于说明目的，其中的设置包含单个 Ceph 节点上存在的 SSD 部件的一个根及一条规则。

在生产环境中，集群设置通常包含热存储以及混合节点（配有 SSD 和 SATA 磁盘）的更多根项和规则项。

1. 准备一台配有 SSD 等高速驱动器的主机。此集群节点将充当高速缓存层。
2. 使用 DeepSea 将该计算机转变成 Ceph 节点。根据第 1.1 节 “添加新的集群节点”中所述安装软件并配置主机计算机。我们假设此节点名为 `node-4`。此节点需要有 4 个 OSD 磁盘。

   这可能会在 CRUSH 地图中生成类似下方所示的项：

```
[...]
host node-4 {
   id -5  # do not change unnecessarily
   # weight 0.012
   alg straw
   hash 0  # rjenkins1
   item osd.6 weight 0.003
   item osd.7 weight 0.003
   item osd.8 weight 0.003
   item osd.9 weight 0.003
}
[...]
```

3. 编辑热存储池（已映射到基于高速 SSD 驱动器的 OSD）的 CRUSH 地图。定义另一个包含 SSD 的根节点的层次结构（命令为“root ssd”）。此外，请更改 SSD 的权重和 CRUSH 规则。有关 CRUSH 地图的详细信息，请参见 http://docs.ceph.com/docs/master/rados/operations/crush-map/ 。

   使用 `getcrushmap` 和 `crushtool` 等命令行工具直接编辑 CRUSH 地图。

   a. 检索当前地图，并将其另存为 `c.map`：

      ```
      cephadm > sudo ceph osd getcrushmap -o c.map
      ```

   b. 反编译 `c.map`，并将其另存为 `c.txt`：

      ```
      cephadm > crushtool -d c.map -o c.txt
      ```

   c. 编辑 `c.txt`：

      ```
      [...]
      host node-4 {
              id -5  # do not change unnecessarily
              # weight 4.000
              alg straw
              hash 0  # rjenkins1
              item osd.6 weight 1.000
              item osd.7 weight 1.000
      ```

```
        item osd.8 weight 1.000
        item osd.9 weight 1.000
}
root ssd {    # newly added root for the SSD hot-storage
        id -6
        alg straw
        hash 0
        item node-4 weight 4.00
}
rule ssd {
        ruleset 4
        type replicated
        min_size 0
        max_size 4
        step take ssd
        step chooseleaf firstn 0 type host
        step emit
}
[...]
```

d. 编译已编辑的 `c.txt` 文件，并将其另存为 `ssd.map`：

```
cephadm > crushtool -c c.txt -o ssd.map
```

e. 最后，安装 `ssd.map`，作为新的 CRUSH 地图：

```
cephadm > sudo ceph osd setcrushmap -i ssd.map
```

4. 创建用于快速缓存分层的热存储池。对该池使用新的“ssd”规则：

```
cephadm > sudo ceph osd pool create hot-storage 100 100 replicated ssd
```

5. 使用默认的“replicated_ruleset”规则创建冷存储池：

```
cephadm > sudo ceph osd pool create cold-storage 100 100 replicated
 replicated_ruleset
```

6. 然后，设置快速缓存层的过程涉及到将后备存储池关联到快速缓存池，在本例中，需要将冷存储（即存储池）关联到热存储（即快速缓存池）：

```
cephadm > sudo ceph osd tier add cold-storage hot-storage
```

7. 要将快速缓存模式设置为“写回”，请执行以下命令：

```
cephadm > sudo ceph osd tier cache-mode hot-storage writeback
```

有关快速缓存模式的详细信息，请参见第 10.4 节 “快速缓存模式”。
写回快速缓存层叠加在后备存储池上，因此需要执行一个额外的步骤：必须将来自存储池的所有客户端流量定向到快速缓存池。例如，要将客户端流量直接定向到快速缓存池，请执行以下命令：

```
cephadm > sudo ceph osd tier set-overlay cold-storage hot-storage
```

10.6.1 配置快速缓存层

可以使用多个选项来配置快速缓存层。使用以下语法：

```
cephadm > sudo ceph osd pool set cachepool key value
```

10.6.1.1 目标大小和类型

以下步骤说明如何使用第 10.5.2.2 节 “小型快速缓存池和大量内存”中提供的值配置快速缓存池

Ceph 的生产快速缓存层针对 `hit_set_type` 使用布隆过滤器：

```
cephadm > sudo ceph osd pool set cachepool hit_set_type bloom
```

`hit_set_count` 和 `hit_set_period` 定义每个命中集应该经历的时长，以及要存储多少个这样的命中集。

```
cephadm > sudo ceph osd pool set cachepool hit_set_count 12
cephadm > sudo ceph osd pool set cachepool hit_set_period 14400
cephadm > sudo ceph osd pool set cachepool target_max_bytes 1000000000000
```

注意

`hit_set_count` 越大，`ceph-osd` 进程耗费的 RAM 就越多。

`min_read_recency_for_promote` 定义在处理读取操作时，要在多少个命中集中检查某个对象是否存在。检查结果用于确定是否要以异步方式提升该对象。此参数的值应介于 0 到 `hit_set_count` 之间。如果设置为 0，则始终提升该对象。如果设置为 1，则检查当前命中集。如果此对象在当前命中集中，则提升此对象。否则，将不提升。如果设置为其他值，则检查相应数量的存档命中集。如果在任何最近的 `min_read_recency_for_promote` 命中集中找到了该对象，则提升该对象。

可以针对写入操作设置类似的参数，即 `min_write_recency_for_promote`：

```
cephadm > sudo ceph osd pool set cachepool min_read_recency_for_promote 2
cephadm > sudo ceph osd pool set cachepool min_write_recency_for_promote 2
```

注意

期限越长，`min_read_recency_for_promote` 和 `min_write_recency_for_promote` 的值越大，`ceph-osd` 守护进程耗费的 RAM 就越多。特别是，当代理正在清理或逐出快速缓存对象时，所有 `hit_set_count` 命中集都会加载到 RAM 中。

10.6.1.2 快速缓存大小调整

快速缓存分层代理执行两项主要功能：

清理

代理识别已修改的（脏）对象，并将其转发到存储池长期存储。

逐出

代理识别未曾修改的（干净）对象，并将其中最近用得最少的对象从快速缓存中逐出。

10.6.1.2.1 绝对大小调整

快速缓存分层代理可根据字节总数或对象总数来清理或逐出对象。要指定最大字节数，请执行以下命令：

```
cephadm > sudo ceph osd pool set cachepool target_max_bytes num_of_bytes
```

要指定最大对象数，请执行以下命令：

```
cephadm > sudo ceph osd pool set cachepool target_max_objects num_of_objects
```

注意

Ceph 无法自动确定快速缓存池的大小，因此，便需要配置绝对大小。否则，清理和逐出功能将无法正常工作。如果这两项限制都指定，则一旦触发任一阈值，快速缓存分层代理就会开始执行清理或逐出。

注意

仅当达到 `target_max_bytes` 或 `target_max_objects` 时，才会阻止所有客户端请求。

10.6.1.2.2 相对大小调整

快速缓存分层代理可以根据快速缓存池的相对大小（通过第 10.6.1.2.1 节 “绝对大小调整”中所述的 `target_max_bytes` 或 `target_max_objects` 指定）清理或逐出对象。当快速缓存池中的已修改（脏）对象达到特定百分比时，快速缓存分层代理会将这些对象清理到存储池。要设置 `cache_target_dirty_ratio`，请执行以下命令：

```
cephadm > sudo ceph osd pool set cachepool cache_target_dirty_ratio 0.0...1.0
```

例如，如果将值设置为 0.4，则当已修改（脏）对象的大小达到快速缓存池容量的 40% 时，就会开始清理这些对象。

```
cephadm > sudo ceph osd pool set hot-storage cache_target_dirty_ratio 0.4
```

当脏对象的大小达到容量的特定百分比时，将以更高的速度清理。使用 `cache_target_dirty_high_ratio`：

```
cephadm > sudo ceph osd pool set cachepool
 cache_target_dirty_high_ratio 0.0..1.0
```

当快速缓存池的大小达到其容量的特定百分比时，快速缓存分层代理会逐出对象，以维持可用容量。要设置 `cache_target_full_ratio`，请执行以下命令：

```
cephadm > sudo ceph osd pool set cachepool cache_target_full_ratio 0.0..1.0
```

10.6.1.3 快速缓存期限

您可以指定在快速缓存分层代理将最近已修改的（脏）对象清理到后备存储池之前，这些对象至少可保留的期限：

```
cephadm > sudo ceph osd pool set cachepool cache_min_flush_age num_of_seconds
```

您可以指定在将某个对象逐出快速缓存层之前，该对象至少可保留的期限：

```
cephadm > sudo ceph osd pool set cachepool cache_min_evict_age num_of_seconds
```

10.6.1.4 对命中集使用 GMT

快速缓存层设置包含一个称作命中集的布隆过滤器。该过滤器测试某个对象是属于一组热对象还是冷对象。对象将添加到命中集，其名称后面附有时戳。

如果集群计算机位于不同的时区，且时戳根据当地时间派生，则命中集中对象的名称可能包含将来或过去的时戳，致使用户产生误解。在最坏的情况下，对象可能根本不在命中集中。

为防止这种问题发生，在新建的快速缓存层设置中，`use_gmt_hitset` 默认设为“1”。这样，您便可以在创建命中集的对象名称时，强制 OSD 使用 GMT（格林威治标准时间）时戳。

警告：保留默认值

不要更改 `use_gmt_hitset` 的默认值“1”。如果与此选项相关的错误不是因集群设置造成，切勿手动更改此选项。否则，集群的行为可能变得无法预测。

III 访问集群数据

11 Ceph Object Gateway

本章介绍对象网关相关管理任务的详细信息，例如，检查服务的状态，管理帐户、多站点网关或 LDAP 身份验证。

11.1 对象网关限制和命名限制

下面列出了对象网关的一些重要限制：

11.1.1 桶限制

通过 S3 API 访问对象网关时，桶名必须符合 DNS 且允许使用短划线字符“-”。当通过 Swift API 访问对象网关时，您可使用支持 UTF-8 的字符（斜杠字符“/”除外）的任何组合。桶名最多可包含 255 个字符。桶名必须唯一。

提示：使用符合 DNS 的桶名

虽然通过 Swift API 访问时，可使用任何基于 UTF-8 的桶名，但仍建议您根据 S3 命名限制对桶命名，以免在通过 S3 API 访问同一个桶时发生问题。

11.1.2 存储的对象的限制

每个用户的对象数量上限

默认无限制（大约不超过 2^63）。

每个桶的对象数量上限

默认无限制（大约不超过 2^63）。

要上载/存储的对象的最大大小

单次上载的上限为 5GB。更大的对象可分为多个部分上载。多部分块的最大数量为 10000。

11.1.3 HTTP 报头限制

HTTP 报头和请求限制取决于所使用的 Web 前端。默认的 CivetWeb 限制 HTTP 报头数量最多为 64 个，HTTP 报头大小最大为 16kB。

11.2 部署对象网关

建议通过 DeepSea 基础结构来部署 Ceph Object Gateway，具体做法是在 Salt Master 上的 `policy.cfg` 文件中添加相关的 `role-rgw [...]` 行，并运行必要的 DeepSea 阶段。

- 要在 Ceph 集群部署期间加入对象网关，请参见《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”和《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.5.1 节 “`policy.cfg` 文件”。
- 要在已部署的集群中添加对象网关角色，请参见第 1.2 节 “为节点添加新的角色”。

11.3 操作对象网关服务

通过运行 `systemctl` 命令来操作对象网关服务。您需要拥有 `root` 特权才能操作对象网关服务。请注意，`gateway_host` 是您需要操作其对象网关实例的服务器的主机名。

对象网关服务支持以下子命令：

systemctl status ceph-radosgw@rgw.`gateway_host`
: 列显服务的状态信息。

systemctl start ceph-radosgw@rgw.`gateway_host`
: 如果服务尚未运行，则将它启动。

systemctl restart ceph-radosgw@rgw.`gateway_host`
: 重启动服务。

systemctl stop ceph-radosgw@rgw.`gateway_host`
: 停止正在运行的服务。

systemctl enable ceph-radosgw@rgw.`gateway_host`
: 启用服务，以便在系统启动时自动启动该服务。

systemctl disable ceph-radosgw@rgw.`gateway_host`

禁用服务，以便在系统启动时不自动启动该服务。

11.4 配置参数

在 `ceph.conf` 文件中指定大量选项可能会影响对象网关的行为。下面列出了最重要的选项。有关完整列表，请参见 http://docs.ceph.com/docs/master/radosgw/config-ref/ 。

rgw_thread_pool_size

Civetweb 服务器的线程数。如果需要处理更多请求，请设置更高的值。默认为 100 个线程。

rgw_num_rados_handles

对象网关的 RADOS 集群句柄数（请参见 http://docs.ceph.com/docs/master/rados/api/librados-intro/#step-2-configuring-a-cluster-handle ）。如果 RADOS 句柄数可配置，将可大幅提升所有类型的工作负载的性能。现在，每个对象网关工作线程都可以在其有效期内选取某个 RADOS 句柄。默认值是 1。

rgw_max_chunk_size

将在单个操作中读取的数据块的最大大小。将值增至 4MB (4194304) 可以在处理大型对象时提高性能。默认值为 128kB (131072)。

11.4.1 补充说明

rgw dns name

如果将参数 `rgw dns name` 添加到 `ceph.conf`，请确保已配置 S3 客户端，以定向 `rgw dns name` 所指定端点的请求。

11.5 管理对象网关的访问方式

您可以使用与 S3 或 Swift 兼容的接口来与对象网关通讯。S3 接口与大部分 Amazon S3 RESTful API 都兼容。Swift 接口与大部分 OpenStack Swift API 都兼容。

这两个接口都要求创建特定的用户，并安装相关的客户端软件，以使用该用户的机密密钥来与网关通讯。

11.5.1 访问对象网关

11.5.1.1 S3 接口访问

要访问 S3 接口，需要一个 REST 客户端。`S3cmd` 是一个命令行 S3 客户端。您可以在 OpenSUSE Build Service (https://build.opensuse.org/package/show/Cloud:Tools/s3cmd) 中找到它。该储存库包含既适用于 SUSE Linux Enterprise 发行套件又适用于基于 openSUSE 的发行套件的版本。

如果您想测试自己是否能够访问 S3 接口，也可以编写一个简短的 Python 脚本。该脚本将连接到对象网关，创建新桶，并列出所有桶。`aws_access_key_id` 和 `aws_secret_access_key` 的值取自第 11.5.2.1 节 “添加 S3 和 Swift 用户”中所述 `radosgw_admin` 命令返回的 `access_key` 和 `secret_key` 的值。

1. 安装 `python-boto` 包：

```
sudo zypper in python-boto
```

2. 创建名为 `s3test.py` 的新 Python 脚本，并在其中包含以下内容：

```
import boto
import boto.s3.connection
access_key = '11BS02LGFB6AL6H1ADMW'
secret_key = 'vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY'
conn = boto.connect_s3(
aws_access_key_id = access_key,
aws_secret_access_key = secret_key,
host = '{hostname}',
is_secure=False,
calling_format = boto.s3.connection.OrdinaryCallingFormat(),
)
bucket = conn.create_bucket('my-new-bucket')
for bucket in conn.get_all_buckets():
```

```
  print "{name}\t{created}".format(
  name = bucket.name,
  created = bucket.creation_date,
  )
```

将 `{hostname}` 替换为在其中配置了对象网关服务的主机的主机名，例如 `gateway_host`。

3. 运行脚本：

```
python s3test.py
```

该脚本将输出类似下方所示的信息：

```
my-new-bucket 2015-07-22T15:37:42.000Z
```

11.5.1.2 Swift 接口访问

要通过 Swift 接口访问对象网关，需要使用 `swift` 命令行客户端。该接口的手册页 `man 1 swift` 介绍了有关其命令行选项的详细信息。

SUSE Linux Enterprise 12 SP3 的“Public Cloud”模块中包含了相应的包。在安装该包之前，需要激活该模块并刷新软件储存库：

```
sudo SUSEConnect -p sle-module-public-cloud/12/x86_64
sudo zypper refresh
```

要安装 `swift` 命令，请运行以下命令：

```
sudo zypper in python-swiftclient
```

使用以下语法进行 swift 访问：

```
swift -A http://IP_ADDRESS/auth/1.0 \
-U example_user:swift -K 'swift_secret_key' list
```

请将 `IP_ADDRESS` 替换为网关服务器的 IP 地址，将 `swift_secret_key` 替换为在第 11.5.2.1 节 “添加 S3 和 Swift 用户”中针对 `swift` 用户执行 `radosgw-admin key create` 命令后的输出中的相应值。

例如：

```
swift -A http://gateway.example.com/auth/1.0 -U example_user:swift \
-K 'r5wWIxjOCeEO7DixD1FjTLmNYIViaC6JVhi3013h' list
```

输出为：

```
my-new-bucket
```

11.5.2 管理 S3 和 Swift 帐户

11.5.2.1 添加 S3 和 Swift 用户

需要创建用户、访问钥和机密才能让最终用户与网关交互。用户分两种类型：用户和子用户。与 S3 接口交互时使用用户，子用户是 Swift 接口的用户。每个子用户都与某个用户相关联。

也可以通过 DeepSea 文件 `rgw.sls` 添加用户。有关示例，请参见第 14.3.1 节 “NFS Ganesha 的不同对象网关用户”。

要创建 Swift 用户，请执行以下步骤：

1. 要创建 Swift 用户（在我们的术语中称作子用户），需要先创建关联的用户。

   ```
   sudo radosgw-admin user create --uid=username \
    --display-name="display-name" --email=email
   ```

   例如：

   ```
   sudo radosgw-admin user create \
      --uid=example_user \
      --display-name="Example User" \
      --email=penguin@example.com
   ```

2. 要创建用户的子用户（用于 Swift 接口），必须指定用户 ID（--uid=`username`）、子用户 ID 和该子用户的访问级别。

   ```
   sudo radosgw-admin subuser create --uid=uid \
    --subuser=uid \
   ```

```
 --access=[ read | write | readwrite | full ]
```

例如：

```
sudo radosgw-admin subuser create --uid=example_user \
 --subuser=example_user:swift --access=full
```

3. 为用户生成机密密钥。

```
sudo radosgw-admin key create \
   --gen-secret \
   --subuser=example_user:swift \
   --key-type=swift
```

4. 这两个命令都会输出 JSON 格式的数据，其中显示了用户状态。请注意以下几行，并记住 `secret_key` 值：

```
"swift_keys": [
   { "user": "example_user:swift",
     "secret_key": "r5wWIxjOCeEO7DixD1FjTLmNYIViaC6JVhi3013h"}],
```

通过 S3 接口访问对象网关时，需要运行以下命令来创建 S3 用户：

```
sudo radosgw-admin user create --uid=username \
 --display-name="display-name" --email=email
```

例如：

```
sudo radosgw-admin user create \
   --uid=example_user \
   --display-name="Example User" \
   --email=penguin@example.com
```

该命令还会创建用户的访问钥和机密密钥。检查该命令输出中的 `access_key` 和 `secret_key` 关键字及其值：

```
[...]
 "keys": [
       { "user": "example_user",
         "access_key": "11BSO2LGFB6AL6H1ADMW",
```

```
        "secret_key": "vzCEkuryfn060dfee4fgQPqFrncKEIkh3ZcdOANY"}],
 [...]
```

11.5.2.2 删除 S3 和 Swift 用户

删除 S3 用户与删除 Swift 用户的过程类似。不过，在删除 Swift 用户时，您可能需要同时删除该用户及其子用户。

要删除 S3 或 Swift 用户（包括其所有子用户），请在以下命令中指定 `user rm` 和用户 ID：

```
sudo radosgw-admin user rm --uid=example_user
```

要删除子用户，请指定 `subuser rm` 和子用户 ID。

```
sudo radosgw-admin subuser rm --uid=example_user:swift
```

可使用以下选项：

--purge-data
: 清除与该用户 ID 关联的所有数据。

--purge-keys
: 清除与该用户 ID 关联的所有密钥。

提示：删除子用户

删除某个子用户时，删除的是其对 Swift 接口的访问权限。该用户仍会保留在系统中。

11.5.2.3 更改 S3 和 Swift 用户的访问钥与机密密钥

访问网关时，`access_key` 和 `secret_key` 参数用于标识对象网关用户。更改现有用户密钥的过程与创建新用户密钥的过程相同，旧密钥将被重写。

对于 S3 用户，请运行以下命令：

```
sudo radosgw-admin key create --uid=example_user --key-type=s3 --gen-access-key
 --gen-secret
```

对于 Swift 用户，请运行以下命令：

```
sudo radosgw-admin key create --subuser=example_user:swift --key-type=swift --
gen-secret
```

`--key-type=type`

指定密钥的类型。值为 `swift` 或 `s3`。

`--gen-access-key`

生成随机访问钥（默认针对 S3 用户）。

`--gen-secret`

生成随机机密密钥。

`--secret=key`

指定机密密钥，例如手动生成的密钥。

11.5.2.4 用户配额管理

Ceph Object Gateway 允许您针对用户以及用户拥有的桶设置配额。配额包括一个桶中的最大对象数，以及最大存储大小 (MB)。

在启用用户配额之前，需要先设置该配额的参数：

```
sudo radosgw-admin quota set --quota-scope=user --uid=example_user \
 --max-objects=1024 --max-size=1024
```

`--max-objects`

指定最大对象数。指定负值会禁用检查。

`--max-size`

指定最大字节数。指定负值会禁用检查。

`--quota-scope`

设置配额的范围。选项包括 `bucket` 和 `user`。桶配额将应用到用户拥有的桶。用户配额将应用到用户。

设置用户配额后，可启用该配额：

```
sudo radosgw-admin quota enable --quota-scope=user --uid=example_user
```

要禁用配额，请执行以下命令：

```
sudo radosgw-admin quota disable --quota-scope=user --uid=example_user
```

要列出配额设置，请执行以下命令：

```
sudo radosgw-admin user info --uid=example_user
```

要更新配额统计数字，请执行以下命令：

```
sudo radosgw-admin user stats --uid=example_user --sync-stats
```

11.6 为对象网关启用 HTTPS/SSL

要让默认对象网关角色可使用 SSL 进行安全通讯，您需要拥有 CA 颁发的证书，或创建自我签名证书。为对象网关启用 HTTPS 的配置方法有两种，简单的方法是使用默认设置，高级方法可以微调 HTTPS 相关设置。

11.6.1 创建自我签名证书

提示

如果您已拥有 CA 签名的有效证书，请跳过本节。

默认情况下，DeepSea 预期证书文件位于 Salt Master 的 `/srv/salt/ceph/rgw/cert/rgw.pem` 下。它会将证书分发到具有对象网关角色的 Salt Minion 的 `/etc/ceph/rgw.pem` 下，以便 Ceph 读取。

以下过程说明如何在 Salt Master 节点上生成自我签名的 SSL 证书。

1. 在 `/etc/ssl/openssl.cnf` 文件的 `[v3_req]` 段落，为您想向其宣告对象网关的所有主机名添加 `subjectAltName` 选项：

   ```
   [...]
   [ v3_req ]
   subjectAltName = ${ENV::SAN}
   [...]
   ```

2. 使用 `openssl` 创建密钥和证书。在 `openssl` 前加上 `env SAN=DNS:fqdn` 前缀。输入需要包含在证书中的所有数据。建议您输入 FQDN 作为常用名。对证书签名前，确认“X509v3 Subject Alternative Name:”包含在请求的扩展中，并且生成的证书中设置了“X509v3 Subject Alternative Name:”。

```
root@master # env SAN=DNS:fqdn openssl req -x509 -nodes -days 1095 \
 -newkey rsa:4096 -keyout rgw.key -out /srv/salt/ceph/rgw/cert/rgw.pem
```

11.6.2 简单的 HTTPS 配置

默认情况下，对象网关节点上的 Ceph 会读取 `/etc/ceph/rgw.pem` 证书，并使用端口 443 进行 SSL 安全通讯。如果您不需要更改这些值，请执行以下步骤：

1. 编辑 `/srv/pillar/ceph/stack/global.yml`，添加下行：

```
rgw_configurations: rgw-ssl
rgw_init: default-ssl
```

2. 运行 DeepSea 阶段 2、3、和 4 以应用这些更改：

```
root@master # salt-run state.orch ceph.stage.2
root@master # salt-run state.orch ceph.stage.3
root@master # salt-run state.orch ceph.stage.4
```

11.6.3 高级 HTTPS 配置

如果您需要更改对象网关 SSL 设置的默认值，请执行以下步骤：

1. 将默认对象网关 SSL 配置复制到 `ceph.conf.d` 子目录：

```
root@master # cp /srv/salt/ceph/configuration/files/rgw-ssl.conf \
 /srv/salt/ceph/configuration/files/ceph.conf.d/rgw.conf
```

2. 编辑 `/srv/salt/ceph/configuration/files/ceph.conf.d/rgw.conf`，更改默认选项，例如端口号或 SSL 证书路径，以反映您的设置。

3. 运行 DeepSea 阶段 3 和 4 以应用这些更改：

```
root@master # salt-run state.orch ceph.stage.3
root@master # salt-run state.orch ceph.stage.4
```

提示：绑定到多个端口

Civetweb 服务器可以绑定到多个端口。如果您需要使用 SSL 和非 SSL 两种连接来访问单个对象网关实例，这种做法将非常实用。指定多个端口时，请使用加号“+”分隔端口号。两个端口的配置行如下所示：

```
[client.{{ client }}]
rgw_frontends = civetweb port=80+443s ssl_certificate=/etc/ceph/rgw.pem
```

11.7 同步模块

使用 Jewel 中引入的对象网关多站点功能可以创建多个区域，并在这些区域之间镜像数据和元数据。同步模块构建在多站点框架的基础上，可将数据和元数据转发到不同的外部层。每当发生数据更改（创建桶或用户等元数据操作也视为数据更改）时，可以通过同步模块执行一系列操作。随着 rgw 多站点更改最终在远程站点上保持一致，更改将以异步方式传播。因而很多情况下都适合使用同步模块，例如，将对象存储备份到外部云集群或使用磁带机的自定义备份解决方案、在 Elasticsearch 中为元数据编制索引，等等。

11.7.1 同步区域

同步模块的配置位于区域本地。同步模块会确定区域是要导出数据，还是只能使用已在另一区域中修改的数据。从 Luminous 版本开始，支持的同步插件包括 `elasticsearch`、`rgw` 和 `log`，其中 rgw 是在区域之间同步数据的默认同步插件，log 是记录远程区域中发生的元数据操作的普通同步插件。以下各节内容包含了使用 `elasticsearch` 同步模块的区域示例。其过程与配置任何其他同步插件的过程都类似。

注意：默认同步插件

`rgw` 是默认的同步插件，不需要对此进行显式配置。

11.7.1.1 要求和假设

我们假设已根据第 11.11 节 “多站点对象网关”中所述创建了一个简单的多站点配置，它由 `us-east` 和 `us-west` 这两个区域组成。现在，我们添加第三个区域 `us-east-es`，此区域只处理来自其他站点的元数据。此区域可与 `us-east` 位于同一 Ceph 集群中，也可位于不同的集群中。此区域只使用来自其他区域的元数据，此区域中的对象网关不会直接处理任何最终用户请求。

11.7.1.2 配置同步模块

1. 创建类似于第 11.11 节 “多站点对象网关”中所述区域的第三个区域，例如

```
root # radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-es \
--access-key={system-key} --secret={secret} --endpoints=http://rgw-es:80
```

2. 可通过以下命令配置此区域的同步模块

```
root # radosgw-admin zone modify --rgw-zone={zone-name} --tier-type={tier-
type} \
--tier-config={set of key=value pairs}
```

3. 例如，在 `elasticsearch` 同步模块中运行以下命令

```
root # radosgw-admin zone modify --rgw-zone={zone-name} --tier-
type=elasticsearch \
--tier-config=endpoint=http://localhost:9200,num_shards=10,num_replicas=1
```

   有关支持的各个 tier-config 选项，请参见第 11.7.2 节 “在 Elasticsearch 中存储元数据”。

4. 最后，更新周期

```
root # radosgw-admin period update --commit
```

5. 现在，在区域中启动 radosgw

```
root # systemctl start ceph-radosgw@rgw.`hostname -s`
root # systemctl enable ceph-radosgw@rgw.`hostname -s`
```

11.7.2 在 Elasticsearch 中存储元数据

此同步模块会将来自其他区域的元数据写入 Elasticsearch。从 Luminous 版本开始，我们当前存储在 Elasticsearch 中的是数据字段的 JSON。

```
{
  "_index" : "rgw-gold-ee5863d6",
  "_type" : "object",
  "_id" : "34137443-8592-48d9-8ca7-160255d52ade.34137.1:object1:null",
  "_score" : 1.0,
  "_source" : {
    "bucket" : "testbucket123",
    "name" : "object1",
    "instance" : "null",
    "versioned_epoch" : 0,
    "owner" : {
      "id" : "user1",
      "display_name" : "user1"
    },
    "permissions" : [
      "user1"
    ],
    "meta" : {
      "size" : 712354,
      "mtime" : "2017-05-04T12:54:16.462Z",
      "etag" : "7ac66c0f148de9519b8bd264312c4d64"
    }
  }
}
```

11.7.2.1 Elasticsearch 层类型配置参数

endpoint

指定要访问的 Elasticsearch 服务器端点。

num_shards

（整数）数据同步初始化时将为 Elasticsearch 配置的分片数量。请注意，初始化之后将无法更改此数量。在此处进行任何更改都需要重构建 Elasticsearch 索引，并需要重新初始化数据同步进程。

num_replicas

（整数）数据同步初始化时为 Elasticsearch 配置的副本数量。

explicit_custom_meta

(true | false) 指定是否将为所有用户自定义元数据编制索引，或者用户是否需要（在桶级别）配置应为哪些客户元数据项编制索引。此参数默认为 false

index_buckets_list

（逗号分隔的字符串列表）如果为空，则为所有桶编制索引。否则，只为此处指定的桶编制索引。可以提供桶前缀（例如“foo*”）或桶后缀（例如“*bar”）。

approved_owners_list

（逗号分隔的字符串列表）如果为空，将为所有所有者的桶编制索引（需遵守其他限制）；否则，将只为指定所有者拥有的桶编制索引。也可以提供后缀和前缀。

override_index_path

（字符串）如果非空，则此字符串将用作 Elasticsearch 索引路径。否则，将在同步初始化时确定并生成索引路径。

11.7.2.2 元数据查询

由于 Elasticsearch 集群现在存储对象元数据，因此务必确保 Elasticsearch 端点不会向公众公开，只有集群管理员可访问它们。向最终用户自己公开元数据查询会造成问题，因为我们希望该用户只查询自己的元数据，而不能查询任何其他用户的元数据，这就要求 Elasticsearch 集群像 RGW 所做的那样来对用户进行身份验证，而这就导致了问题发生。

从 Luminous 版本开始，元数据主区域中的 RGW 可处理最终用户请求。这样就无需向公众公开 Elasticsearch 端点，同时也解决了身份验证和授权问题，因为 RGW 本身就能对最终用户请求进行身份验证。出于此目的，RGW 在桶 API 中引入了可处理 Elasticsearch 请求的新查询。所有这些请求必须发送到元数据主区域。

获取 Elasticsearch 查询

```
GET /BUCKET?query={query-expr}
```

请求参数：

- max-keys：要返回的最大项数
- marker：分页标识

```
expression := [(]<arg> <op> <value> [)][<and|or> ...]
```

运算符为下列其中一项：<、<=、==、>=、>

例如：

```
GET /?query=name==foo
```

将返回用户有权读取且名为“foo”的所有带索引键。输出内容将是 XML 格式的键列表，与 S3 的“列出桶”请求的响应类似。

配置自定义元数据字段

定义应该（在指定的桶中）为哪些自定义元数据项编制索引，以及这些键的类型是什么。如果配置了显式自定义元数据索引，则需要此定义，以便 rgw 为指定的自定义元数据值编制索引。如果未配置，在带索引元数据键的类型不是字符串的情况下，也需要此定义。

```
POST /BUCKET?mdsearch
x-amz-meta-search: <key [; type]> [, ...]
```

多个元数据字段必须用逗号加以分隔，可以使用分号“;”强制指定字段的类型。当前允许的类型包括字符串（默认值）、整数和日期。例如，如果您想将自定义对象元数据 x-amz-meta-year、x-amz-meta-date 和 x-amz-meta-title 的索引分别指定为整数、日期和字符串类型，可执行以下命令

```
POST /mybooks?mdsearch
x-amz-meta-search: x-amz-meta-year;int, x-amz-meta-release-date;date, x-
amz-meta-title;string
```

删除自定义元数据配置

删除自定义元数据桶配置。

```
DELETE /BUCKET?mdsearch
```

获取自定义元数据配置

检索自定义元数据桶配置。

```
GET /BUCKET?mdsearch
```

11.8 LDAP 身份验证

除了默认的本地用户身份验证以外，对象网关还能利用 LDAP 服务器服务来对用户进行身份验证。

11.8.1 身份验证机制

对象网关从令牌提取用户的 LDAP 身份凭证。可以基于用户名构造搜索过滤器。对象网关使用配置的服务帐户在目录中搜索匹配的项。如果找到了某个项，对象网关会尝试使用令牌中的密码绑定到所找到的判别名。如果身份凭证有效，绑定将会成功，并且对象网关会授予访问权限。

您可以通过将搜索范围设置为特定的组织单位，或者指定自定义搜索过滤器（例如，要求特定的组成员资格、自定义对象类或属性），来限制允许的用户。

11.8.2 要求

- LDAP 或 Active Directory：对象网关可访问的运行中 LDAP 实例。
- 服务帐户：对象网关要使用且拥有搜索权限的 LDAP 身份凭证。
- 用户帐户：LDAP 目录中的至少一个用户帐户。

重要：LDAP 用户和本地用户不能重叠

不得对本地用户以及要使用 LDAP 进行身份验证的用户使用相同的用户名。对象网关无法区分两者，会将它们视为同一个用户。

提示：健康检查

使用 `ldapsearch` 实用程序可校验服务帐户或 LDAP 连接。例如：

```
ldapsearch -x -D "uid=ceph,ou=system,dc=example,dc=com" -W \
-H ldaps://example.com -b "ou=users,dc=example,dc=com" 'uid=*' dn
```

请务必在 Ceph 配置文件中使用相同的 LDAP 参数，以杜绝可能的问题。

11.8.3 将对象网关配置为使用 LDAP 身份验证

`/etc/ceph/ceph.conf` 配置文件中的以下参数与 LDAP 身份验证相关：

`rgw_ldap_uri`

指定要使用的 LDAP 服务器。请务必使用 `ldaps://fqdn:端口` 参数，以免公开传输明文身份凭证。

`rgw_ldap_binddn`

对象网关使用的服务帐户的判别名 (DN)。

`rgw_ldap_secret`

服务帐户的密码。

rgw_ldap_searchdn

指定在目录信息树中搜索用户的范围，可以是用户的组织单位，或某个更具体的组织单位 (OU)。

`rgw_ldap_dnattr`

在构造的搜索过滤器中用来匹配用户名的属性。根据所用的目录信息树 (DIT)，可能会是 `uid` 或 `cn`。

`rgw_search_filter`

如果未指定，则对象网关会使用 `rgw_ldap_dnattr` 设置自动构造搜索过滤器。使用此参数能非常灵活地缩小所允许用户列表的范围。有关详细信息，请参见第 11.8.4 节 “使用自定义搜索过滤器来限制用户访问权限”。

11.8.4 使用自定义搜索过滤器来限制用户访问权限

可通过两种方式使用 `rgw_search_filter` 参数。

11.8.4.1 用于进一步限制所构造搜索过滤器的部分过滤器

部分过滤器的示例：

```
"objectclass=inetorgperson"
```

对象网关将照常使用令牌中的用户名和 `rgw_ldap_dnattr` 的值生成搜索过滤器。然后，构造的过滤器将与 `rgw_search_filter` 属性中的部分过滤器合并。根据所用的用户名和设置，最终的搜索过滤器可能会变成：

```
"(&(uid=hari)(objectclass=inetorgperson))"
```

在这种情况下，仅当在 LDAP 目录中找到了用户“hari”，该用户具有对象类“inetorgperson”并且确实指定了有效密码时，才向他授予访问权限。

11.8.4.2 完整过滤器

完整过滤器必须包含 `USERNAME` 令牌，在尝试身份验证期间，该令牌将替换为用户名。在这种情况下，不再使用 `rgw_ldap_dnattr` 参数。例如，要将有效用户限制为特定的组，可使用以下过滤器：

```
"(&(uid=USERNAME)(memberOf=cn=ceph-users,ou=groups,dc=mycompany,dc=com))"
```

注意：memberOf 属性

在 LDAP 搜索中使用 `memberOf` 属性需要您实施的特定 LDAP 服务器提供服务器端支持。

11.8.5 生成用于 LDAP 身份验证的访问令牌

`radosgw-token` 实用程序基于 LDAP 用户名和密码生成访问令牌。它会输出 base-64 编码字符串，即实际的访问令牌。请使用偏好的 S3 客户端（请参见第 11.5.1 节 “访问对象网关”），将该令牌指定为访问钥，并使用空机密密钥。

```
root@minion > export RGW_ACCESS_KEY_ID="username"
root@minion > export RGW_SECRET_ACCESS_KEY="password"
```

```
root@minion > radosgw-token --encode --ttype=ldap
```

重要：明文身份凭证

访问令牌是一个 base-64 编码的 JSON 结构，包含明文形式的 LDAP 身份凭证。

注意：Active Directory

对于 Active Directory，请使用 `--ttype=ad` 参数。

11.9 桶索引分片

对象网关在索引池中存储桶索引数据，该池默认为 `.rgw.buckets.index`。如果将太多（成百上千个）对象放入单个桶中，并且不设置每个桶的最大对象数量配额（`rgw bucket default quota max objects`），索引池的性能可能会下降。桶索引分片可在允许每个桶中放入大量对象的同时，防止出现此类性能下降的情况。

11.9.1 桶索引重分片

如果随着桶的增大，其初始配置不再能满足需求，则需要对桶的索引池进行重分片。您可以使用自动联机桶索引重分片（请参见第 11.9.1.1 节 “动态重分片”），也可以手动脱机执行桶索引重分片（请参见第 11.9.1.2 节 “手动重分片”）。

11.9.1.1 动态重分片

从 SUSE Enterprise Storage 5 开始，我们支持联机桶重分片。此功能会检测每个桶的对象数量是否达到某个阈值，如果达到，则会相应地自动增加桶索引使用的分片数量。此进程会减少每个桶索引分片中的条目数。

该检测进程在以下情况和环境中运行：

- 当有新的对象添加到桶中时。
- 在定期扫描所有桶的后台进程中。扫描的目的是为了处理未在更新的现有桶。

需要重分片的桶将会添加到 `reshard_log` 队列，且将安排于稍后进行重分片。重分片线程在后台运行，将逐个执行已安排的重分片。

配置动态重分片

`rgw_dynamic_resharding`

启用或禁用动态桶索引重分片。可用的值为“true”或“false”。默认设为“true”。

`rgw_reshard_num_logs`

重分片日志的分片数。默认设为 16。

`rgw_reshard_bucket_lock_duration`

重分片期间将桶对象锁定的时长。默认设为 120 秒。

`rgw_max_objs_per_shard`

每个桶索引分片的最大对象数。默认设为 100000 个对象。

`rgw_reshard_thread_interval`

两轮重分片线程处理间隔的最长时间。默认设为 600 秒。

重要：多站点配置

多站点环境下不支持动态重分片。从 Ceph 12.2.2 起默认会禁用该功能，但建议您再次检查此设置。

用于管理重分片进程的命令

将桶添加到重分片队列：

```
root@minion > radosgw-admin reshard add \
 --bucket BUCKET_NAME \
 --num-shards NEW_NUMBER_OF_SHARDS
```

列出重分片队列：

```
root@minion > radosgw-admin reshard list
```

处理/安排桶重分片：

```
root@minion > radosgw-admin reshard process
```

显示桶重分片状态：

```
root@minion > radosgw-admin reshard status --bucket BUCKET_NAME
```

取消待处理的桶重分片：

```
root@minion > radosgw-admin reshard cancel --bucket BUCKET_NAME
```

11.9.1.2 手动重分片

第 11.9.1.1 节 “动态重分片”中所述的动态重分片仅适用于简单对象网关配置。对于多站点配置，请使用本节中所述的手动重分片。

要手动对桶索引执行脱机重分片，请使用以下命令：

```
root@minion > radosgw-admin bucket reshard
```

`bucket reshard` 命令执行以下操作：

- 为指定对象创建一组新的桶索引对象。
- 分散这些索引对象的所有对象条目。
- 创建新的桶实例。
- 列出新的桶实例以及桶，以便所有新的索引操作都能够应用到新的桶索引。
- 将旧的和新的桶 ID 打印到标准输出。

过程 11.1： 将桶索引池重分片

1. 确保对桶执行的所有操作都已停止。
2. 备份原始桶索引：

```
root@minion > radosgw-admin bi list \
 --bucket=BUCKET_NAME \
 > BUCKET_NAME.list.backup
```

3. 对桶索引重分片：

```
 root@minion > radosgw-admin reshard \
```

```
 --bucket=BUCKET_NAME \
 --num-shards=NEW_SHARDS_NUMBER
```

提示：旧桶 ID

此命令还会将新的和旧的桶 ID 打印到其输出中。请记下旧桶 ID，清除旧的桶索引对象时需要用到它。

4. 将旧桶索引列表与新桶索引列表进行比较，校验列出的对象是否正确。然后，清除旧的桶索引对象：

```
root@minion > radosgw-admin bi purge
 --bucket=BUCKET_NAME
 --bucket-id=OLD_BUCKET_ID
```

11.9.2 新桶的桶索引分片

有两个选项会影响桶索引分片：

- 对于简单配置，请使用 `rgw_override_bucket_index_max_shards` 选项。
- 对于多站点配置，请使用 `bucket_index_max_shards` 选项。

将选项设为 `0` 将禁用桶索引分片。如果将其设为大于 `0` 的值，则会启用桶索引分片，并设置最大分片数。

下面的公式可帮助您计算建议的分片数：

```
number_of_objects_expected_in_a_bucket / 100000
```

注意，分片的最大数量为 7877。

11.9.2.1 简单配置

1. 打开 Ceph 配置文件，然后添加或修改以下选项：

```
rgw_override_bucket_index_max_shards = 12
```

提示：所有或一个对象网关实例

要为对象网关的所有实例配置桶索引分片，请将 `rgw_override_bucket_index_max_shards` 添加到 `[global]` 段落。

要仅为对象网关的某个特定实例配置桶索引分片，请将 `rgw_override_bucket_index_max_shards` 添加到相关实例段落。

2. 重启动对象网关。有关详细信息，请参见第 11.3 节 “操作对象网关服务”。

11.9.2.2 多站点配置

多站点配置可使用另一个索引池来管理故障转移。要为一个区域组内的区域配置一致的分片数量，请在该区域组的配置中设置 `bucket_index_max_shards` 选项：

1. 将区域组配置导出到 `zonegroup.json` 文件中：

```
root@minion > radosgw-admin zonegroup get > zonegroup.json
```

2. 编辑 `zonegroup.json` 文件，为每个指定的区域设置 `bucket_index_max_shards` 选项。

3. 重设置区域组：

```
root@minion > radosgw-admin zonegroup set < zonegroup.json
```

4. 更新周期：

```
root@minion > radosgw-admin period update --commit
```

11.10 集成 OpenStack Keystone

OpenStack Keystone 是一项用于 OpenStack 产品的身份服务。您可以将对象网关与 Keystone 相集成，以设置接受 Keystone 身份验证令牌的网关。Ceph Object Gateway 端将会对 Keystone 授权可访问网关的用户进行校验，并视需要自动创建用户。对象网关会定期查询 Keystone，以获取已撤消令牌列表。

11.10.1 配置 OpenStack

配置 Ceph Object Gateway 前，需要先配置 OpenStack Keystone 以启用 Swift 服务，并将其指向 Ceph Object Gateway：

1. 设置 Swift 服务。要使用 OpenStack 来验证 Swift 用户，请先创建 Swift 服务：

```
root # openstack service create \
 --name=swift \
 --description="Swift Service" \
 object-store
```

2. 设置端点。创建 Swift 服务后，指向 Ceph Object Gateway。用网关的区域组名或区域名称替换 `REGION_NAME`。

```
root # openstack endpoint create --region REGION_NAME \
 --publicurl   "http://radosgw.example.com:8080/swift/v1" \
 --adminurl    "http://radosgw.example.com:8080/swift/v1" \
 --internalurl "http://radosgw.example.com:8080/swift/v1" \
 swift
```

3. 校验这些设置。创建 Swift 服务并设置端点后，显示端点以确认所有设置正确无误。

```
root # openstack endpoint show object-store
```

11.10.2 配置 Ceph Object Gateway

11.10.2.1 配置 SSL 证书

Ceph Object Gateway 会定期查询 Keystone，以获取已撤消令牌列表。这些请求会被编码并签名。还可配置 Keystone 以提供自我签名令牌，这些令牌同样经过编码和签名。您需要配置网关以便其可以解码并校验这些已签名讯息。因此，需要将 Keystone 用于创建请求的 OpenSSL 证书转换为“nss db”格式：

```
root # mkdir /var/ceph/nss
```

```
root # openssl x509 -in /etc/keystone/ssl/certs/ca.pem \
 -pubkey | certutil -d /var/ceph/nss -A -n ca -t "TCu,Cu,Tuw"
rootopenssl x509 -in /etc/keystone/ssl/certs/signing_cert.pem \
 -pubkey | certutil -A -d /var/ceph/nss -n signing_cert -t "P,P,P"
```

还可以使用自我签名的 SSL 证书终止 OpenStack Keystone，以便让 Ceph Object Gateway 与 Keystone 交互。可在运行 Ceph Object Gateway 的节点上安装 Keystone 的 SSL 证书，也可以将选项 `rgw keystone verify ssl` 的值设为“false”。将 `rgw keystone verify ssl` 设为“false”意味着网关将不会尝试校验证书。

11.10.2.2 配置对象网关的选项

您可以使用以下选项配置 Keystone 集成：

`rgw keystone api version`
: Keystone API 的版本。有效选项为 2 或 3。默认设为 2。

`rgw keystone url`
: Keystone 服务器上的管理 RESTful API 的 URL 和端口号。采用 `SERVER_URL:PORT_NUMBER` 模式。

`rgw keystone admin token`
: 在 Keystone 内部为管理请求配置的令牌或共享密钥。

`rgw keystone accepted roles`
: 处理请求需要具有的角色。默认设为“Member, admin”。

`rgw keystone accepted admin roles`
: 允许用户获取管理特权的角色列表。

`rgw keystone token cache size`
: Keystone 令牌快速缓存中的最大条目数。

`rgw keystone revocation interval`
: 检查已撤消令牌前间隔的秒数。默认设为 15 * 60。

`rgw keystone implicit tenants`
: 在各自的同名租户中创建新用户。默认设为“false”。

`rgw s3 auth use keystone`

如果设为“true”，Ceph Object Gateway 将使用 Keystone 对用户进行身份验证。默认设为“false”。

`nss db path`

NSS 数据库的路径。

还可以配置 Keystone 服务租户、Keystone 的用户和密码（适用于 OpenStack Identity API 2.0 版本），配置方法与配置 OpenStack 服务的方法类似。使用此方法可避免在配置文件中设置共享密钥 `rgw keystone admin token`，生产环境中应禁用该共享机密。服务租户身份凭证应拥有管理员特权，有关详细信息，请参见 OpenStack Keystone 官方文档 (https://docs.openstack.org/keystone/latest/#setting-up-projects-users-and-roles)。相关配置选项如下：

`rgw keystone admin user`

Keystone 管理员用户名。

`rgw keystone admin password`

Keystone 管理员用户密码。

`rgw keystone admin tenant`

Keystone 2.0 版管理员用户租户。

Ceph Object Gateway 用户与 Keystone 租户一一映射。系统会为一个 Keystone 用户指定不同的角色，这些角色可能分布在不止一个租户上。当 Ceph Object Gateway 收到票据时，会查看为该票据指定的租户和用户角色，并根据 `rgw keystone accepted roles` 选项的设置接受或拒绝请求。

提示：映射到 OpenStack 租户

虽然 Swift 租户默认会映射到对象网关用户，但也可通过 `rgw keystone implicit tenants` 选项将其映射到 OpenStack 租户。如此会让容器使用租户名称空间，而不是对象网关默认采用的 S3 之类的全局名称空间。建议在规划阶段就决定好映射方法，以免产生混淆。这是因为以后切换选项只会影响租户下所映射的较新请求，而先前创建的旧桶仍将继续放在全局名称空间中。

对于 OpenStack Identity API 3 版本，您应使用以下选项替换 `rgw keystone admin tenant` 选项：

`rgw keystone admin domain`
: Keystone 管理员用户域。

`rgw keystone admin project`
: Keystone 管理员用户项目。

11.11 多站点对象网关

区域
: 一个或多个对象网关实例的逻辑分组。必须将区域组中的一个区域指定为主区域，负责处理所有桶和用户的创建。

区域组
: 一个区域组由多个区域组成。应设置一个将负责处理系统配置更改的主区域组。

区域组地图
: 用于存放整个系统地图的配置结构，例如，哪个区域组是主区域组、不同区域组之间的关系，以及存储策略等特定配置选项。

领域
: 容纳区域组的容器。使用领域可在集群之间分隔区域组。可以创建多个领域，以便在同一集群中更轻松地运行完全不同的配置。

周期
: 周期存放领域当前状态的配置结构。每个周期都包含一个唯一 ID 和一个版本号。每个领域都有一个关联的当前周期，存放区域组配置的当前状态和存储策略。非主区域发生任何配置更改都会使周期的版本号递增。将主区域更改为其他区域会触发以下更改：

  - 生成具有新周期 ID 和版本号为 1 的新周期。
  - 领域的当前周期会更新，以指向新生成的周期 ID。
  - 领域的版本号将会递增。

您可将每个对象网关配置为参与联合体系结构，在活动区域配置中工作，同时允许写入非主区域。

11.11.1 术语

下面解释了联合体系结构的专用术语：

11.11.2 示例集群设置

本示例重点说明如何创建包含三个不同区域的单个区域组，这三个区域会主动同步其数据。其中两个区域属于同一集群，第三个区域属于其他集群。在对象网关之间镜像数据更改时，不需要同步代理的参与。如此可大大简化配置模式和主动/主动配置。请注意，元数据操作（例如创建新用户）仍需要通过主区域处理。但是，数据操作（例如创建桶和对象）可由任意区域处理。

11.11.3 系统密钥

对象网关需要您在配置区域时创建与 S3 兼容的系统用户，以及他们的访问钥和机密密钥。这样，另一个对象网关实例便可以使用该访问钥和机密密钥远程提取配置。有关创建 S3 用户的详细信息，请参见第 11.5.2.1 节 “添加 S3 和 Swift 用户”。

提示

在创建区域本身之前生成访问钥和机密密钥的做法非常实用，因为这可以让稍后的脚本编写和配置管理工具的使用变得更容易。

对于本示例，我们假设已在环境变量中设置访问钥和机密密钥：

```
# SYSTEM_ACCESS_KEY=1555b35654ad1656d805
# SYSTEM_SECRET_KEY=h7GhxuBLTrlhVUyxSPUKUV8r/2EI4ngqJxD7iBdBYLhwluN30JaT3Q==
```

一般情况下，访问钥包含 20 个字母数字字符，而机密密钥包含 40 个字母数字字符（也可以包含 +/= 字符）。可在命令行中生成这些密钥：

```
# SYSTEM_ACCESS_KEY=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 20 | head
 -n 1)
# SYSTEM_SECRET_KEY=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 40 | head
 -n 1)
```

11.11.4 命名约定

本示例介绍设置主区域的过程。我们假设有一个名为 `us` 的区域组，该区域组横跨美国，将作为我们的主区域组。该区域组将包含以 `区域组-区域` 格式编写的两个区域。这只是我们一贯采用的格式，您可以选择偏好的格式。概括如下：

- 主区域组：美国：`us`
- 主区域：美国东部区域 1：`us-east-1`
- 次要区域：美国东部区域 2：`us-east-2`
- 次要区域：美国西部区域：`us-west`

此配置将属于名为 `gold` 的较大领域。`us-east-1` 和 `us-east-2` 区域属于同一个 Ceph 集群，`us-east-1` 是主区域。`us-west` 在另一个不同的 Ceph 集群中。

11.11.5 默认存储池

为对象网关配置了相应的权限后，它便可自行创建默认存储池。`pg_num` 和 `pgp_num` 值取自 `ceph.conf` 配置文件。默认情况下，与区域相关的存储池遵循 `区域名称.存储池名称` 格式约定。以 `us-east-1` 区域为例，它将创建以下存储池：

```
.rgw.root
us-east-1.rgw.control
us-east-1.rgw.data.root
us-east-1.rgw.gc
us-east-1.rgw.log
us-east-1.rgw.intent-log
us-east-1.rgw.usage
us-east-1.rgw.users.keys
us-east-1.rgw.users.email
```

```
us-east-1.rgw.users.swift
us-east-1.rgw.users.uid
us-east-1.rgw.buckets.index
us-east-1.rgw.buckets.data
us-east-1.rgw.meta
```

也可以在其他区域中创建这些存储池，只需将 `us-east-1` 替换为相应的区域名称即可。

11.11.6 创建领域

配置名为 `gold` 的领域，并将其设为默认领域：

```
cephadm > radosgw-admin realm create --rgw-realm=gold --default
{
  "id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
  "name": "gold",
  "current_period": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "epoch": 1
}
```

请注意，每个领域都有一个 ID，这样，以后便可灵活地执行所需的操作（例如，重命名领域）。每当我们更改主区域中的任何设置时，`current_period` 都会发生变化。如果主区域的配置发生更改，导致当前周期发生更改，`epoch` 将会递增。

11.11.7 删除默认区域组

采用默认设置安装对象网关时会创建名为 `default` 的默认区域组。由于我们不再需要默认区域组，因此将其删除。

```
cephadm > radosgw-admin zonegroup delete --rgw-zonegroup=default
```

11.11.8 创建主区域组

创建名为 `us` 的主区域组。该区域组将管理区域组地图，并将更改传播到系统的其余组件。通过将某个区域组标记为默认区域组，可以明确指定要在后续命令中使用的 rgw-zonegroup 开关。

```
cephadm > radosgw-admin zonegroup create --rgw-zonegroup=us \
--endpoints=http://rgw1:80 --master --default
{
  "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
      "http:\/\/rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3website": [],
  "master_zone": "",
  "zones": [],
  "placement_targets": [],
  "default_placement": "",
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
}
```

或者，可使用以下命令将某个区域组标记为默认区域组：

```
cephadm > radosgw-admin zonegroup default --rgw-zonegroup=us
```

11.11.9 创建主区域

现在，请创建一个默认区域并将其添加到默认区域组。请注意，您在执行元数据操作（例如创建用户）时将会用到此区域：

```
cephadm > radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-east-1 \
--endpoints=http://rgw1:80 --access-key=$SYSTEM_ACCESS_KEY --
secret=$SYSTEM_SECRET_KEY
{
  "id": "83859a9a-9901-4f00-aa6d-285c777e10f0",
  "name": "us-east-1",
  "domain_root": "us-east-1/gc.rgw.data.root",
  "control_pool": "us-east-1/gc.rgw.control",
  "gc_pool": "us-east-1/gc.rgw.gc",
  "log_pool": "us-east-1/gc.rgw.log",
```

```
  "intent_log_pool": "us-east-1/gc.rgw.intent-log",
  "usage_log_pool": "us-east-1/gc.rgw.usage",
  "user_keys_pool": "us-east-1/gc.rgw.users.keys",
  "user_email_pool": "us-east-1/gc.rgw.users.email",
  "user_swift_pool": "us-east-1/gc.rgw.users.swift",
  "user_uid_pool": "us-east-1/gc.rgw.users.uid",
  "system_key": {
      "access_key": "1555b35654ad1656d804",
      "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r\/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
  },
  "placement_pools": [
      {
          "key": "default-placement",
          "val": {
              "index_pool": "us-east-1/gc.rgw.buckets.index",
              "data_pool": "us-east-1/gc.rgw.buckets.data",
              "data_extra_pool": "us-east-1/gc.rgw.buckets.non-ec",
              "index_type": 0
          }
      }
  ],
  "metadata_heap": "us-east-1/gc.rgw.meta",
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
}
```

请注意，`--rgw-zonegroup` 和 `--default` 开关会将该区域添加到某个区域组，并将其设为默认区域。或者，也可以使用以下命令实现相同的目的：

```
cephadm > radosgw-admin zone default --rgw-zone=us-east-1
cephadm > radosgw-admin zonegroup add --rgw-zonegroup=us --rgw-zone=us-east-1
```

11.11.9.1 创建系统用户

要访问区域存储池，需要创建一个系统用户。请注意，在配置次要区域时，也需要这些密钥。

```
cephadm > radosgw-admin user create --uid=zone.user \
--display-name="Zone User" --access-key=$SYSTEM_ACCESS_KEY \
--secret=$SYSTEM_SECRET_KEY --system
```

11.11.9.2 更新周期

由于您更改了主区域配置，因此需要提交这些更改，使其在领域配置结构中生效。最初的周期类似下方所示：

```
cephadm > radosgw-admin period get
{
  "id": "09559832-67a4-4101-8b3f-10dfcd6b2707", "epoch": 1, "predecessor_uuid":
 "", "sync_status": [], "period_map":
  {
    "id": "09559832-67a4-4101-8b3f-10dfcd6b2707", "zonegroups": [],
 "short_zone_ids": []
  }, "master_zonegroup": "", "master_zone": "", "period_config":
  {
     "bucket_quota": {
     "enabled": false, "max_size_kb": -1, "max_objects": -1
     }, "user_quota": {
       "enabled": false, "max_size_kb": -1, "max_objects": -1
     }
  }, "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7", "realm_name": "gold",
 "realm_epoch": 1
}
```

更新周期并提交更改：

```
cephadm > radosgw-admin period update --commit
{
  "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
  "epoch": 1,
  "predecessor_uuid": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "sync_status": [ "[...]"
  ],
  "period_map": {
      "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
      "zonegroups": [
          {
              "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
              "name": "us",
              "api_name": "us",
```

```
            "is_master": "true",
            "endpoints": [
                "http:\/\/rgw1:80"
            ],
            "hostnames": [],
            "hostnames_s3website": [],
            "master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "zones": [
                {
                    "id": "83859a9a-9901-4f00-aa6d-285c777e10f0",
                    "name": "us-east-1",
                    "endpoints": [
                        "http:\/\/rgw1:80"
                    ],
                    "log_meta": "true",
                    "log_data": "false",
                    "bucket_index_max_shards": 0,
                    "read_only": "false"
                }
            ],
            "placement_targets": [
                {
                    "name": "default-placement",
                    "tags": []
                }
            ],
            "default_placement": "default-placement",
            "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
        }
    ],
    "short_zone_ids": [
        {
            "key": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "val": 630926044
        }
    ]
},
"master_zonegroup": "d4018b8d-8c0d-4072-8919-608726fa369e",
```

```
  "master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
  "period_config": {
      "bucket_quota": {
          "enabled": false,
          "max_size_kb": -1,
          "max_objects": -1
      },
      "user_quota": {
          "enabled": false,
          "max_size_kb": -1,
          "max_objects": -1
      }
  },
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
  "realm_name": "gold",
  "realm_epoch": 2
}
```

11.11.9.3 启动对象网关

在启动对象网关之前，需要在配置文件中指定对象网关区域和端口选项。有关对象网关及其配置的详细信息，请参见第 11 章 “Ceph Object Gateway”。对象网关的配置段落应类似下方所示：

```
[client.rgw.us-east-1]
rgw_frontends="civetweb port=80"
rgw_zone=us-east-1
```

启动对象网关：

```
sudo systemctl start ceph-radosgw@rgw.us-east-1
```

11.11.10 创建次要区域

在同一个集群中，创建并配置名为 `us-east-2` 的次要区域。可在托管主区域本身的节点中执行以下所有命令。

要创建次要区域，请使用创建主要区域时所用的相同命令，不过需要去掉 master 标志：

```
cephadm > radosgw-admin zone create --rgw-zonegroup=us --endpoints=http://
rgw2:80 \
--rgw-zone=us-east-2 --access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
{
  "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
  "name": "us-east-2",
  "domain_root": "us-east-2.rgw.data.root",
  "control_pool": "us-east-2.rgw.control",
  "gc_pool": "us-east-2.rgw.gc",
  "log_pool": "us-east-2.rgw.log",
  "intent_log_pool": "us-east-2.rgw.intent-log",
  "usage_log_pool": "us-east-2.rgw.usage",
  "user_keys_pool": "us-east-2.rgw.users.keys",
  "user_email_pool": "us-east-2.rgw.users.email",
  "user_swift_pool": "us-east-2.rgw.users.swift",
  "user_uid_pool": "us-east-2.rgw.users.uid",
  "system_key": {
      "access_key": "1555b35654ad1656d804",
      "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r\/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
  },
  "placement_pools": [
      {
          "key": "default-placement",
          "val": {
              "index_pool": "us-east-2.rgw.buckets.index",
              "data_pool": "us-east-2.rgw.buckets.data",
              "data_extra_pool": "us-east-2.rgw.buckets.non-ec",
              "index_type": 0
          }
      }
  ],
  "metadata_heap": "us-east-2.rgw.meta",
  "realm_id": "815d74c2-80d6-4e63-8cfc-232037f7ff5c"
}
```

11.11.10.1 更新周期

通过执行周期更新并提交更改，通知所有网关有关系统地图中发生的新变化：

```
cephadm > radosgw-admin period update --commit
{
  "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
  "epoch": 2,
  "predecessor_uuid": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "sync_status": [ "[...]"
  ],
  "period_map": {
      "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
      "zonegroups": [
          {
              "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
              "name": "us",
              "api_name": "us",
              "is_master": "true",
              "endpoints": [
                  "http:\/\/rgw1:80"
              ],
              "hostnames": [],
              "hostnames_s3website": [],
              "master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
              "zones": [
                  {
                      "id": "83859a9a-9901-4f00-aa6d-285c777e10f0",
                      "name": "us-east-1",
                      "endpoints": [
                          "http:\/\/rgw1:80"
                      ],
                      "log_meta": "true",
                      "log_data": "false",
                      "bucket_index_max_shards": 0,
                      "read_only": "false"
                  },
                  {
                      "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
```

```
                    "name": "us-east-2",
                    "endpoints": [
                        "http:\/\/rgw2:80"
                    ],
                    "log_meta": "false",
                    "log_data": "true",
                    "bucket_index_max_shards": 0,
                    "read_only": "false"
                }

            ],
            "placement_targets": [
                {
                    "name": "default-placement",
                    "tags": []
                }
            ],
            "default_placement": "default-placement",
            "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
        }
    ],
    "short_zone_ids": [
        {
            "key": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "val": 630926044
        },
        {
            "key": "950c1a43-6836-41a2-a161-64777e07e8b8",
            "val": 4276257543
        }

    ]
},
"master_zonegroup": "d4018b8d-8c0d-4072-8919-608726fa369e",
"master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
"period_config": {
    "bucket_quota": {
        "enabled": false,
```

```
            "max_size_kb": -1,
            "max_objects": -1
        },
        "user_quota": {
            "enabled": false,
            "max_size_kb": -1,
            "max_objects": -1
        }
    },
    "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
    "realm_name": "gold",
    "realm_epoch": 2
}
```

11.11.10.2 启动对象网关

调整次要区域的对象网关配置，并启动对象网关：

```
[client.rgw.us-east-2]
rgw_frontends="civetweb port=80"
rgw_zone=us-east-2
```

```
cephadm > sudo systemctl start ceph-radosgw@rgw.us-east-2
```

11.11.11 将对象网关添加到第二个集群

第二个 Ceph 集群与初始集群属于同一个区域组，不过可以位于不同的地理位置。

11.11.11.1 默认领域和区域组

由于已创建第一个网关的领域，因此可在此处提取该领域并将其设为默认领域：

```
cephadm > radosgw-admin realm pull --url=http://rgw1:80 \
--access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
{
    "id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
```

```
  "name": "gold",
  "current_period": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
  "epoch": 2
}
cephadm > radosgw-admin realm default --rgw-realm=gold
```

通过提取周期，从主区域中获取配置：

```
cephadm > radosgw-admin period pull --url=http://rgw1:80 \
--access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY
```

将已创建的 `us` 区域组设置为默认区域组：

```
cephadm > radosgw-admin zonegroup default --rgw-zonegroup=us
```

11.11.11.2 次要区域配置

使用相同的系统密钥创建名为 `us-west` 的新区域：

```
cephadm > radosgw-admin zone create --rgw-zonegroup=us --rgw-zone=us-west \
--access-key=$SYSTEM_ACCESS_KEY --secret=$SYSTEM_SECRET_KEY \
--endpoints=http://rgw3:80 --default
{
  "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
  "name": "us-west",
  "domain_root": "us-west.rgw.data.root",
  "control_pool": "us-west.rgw.control",
  "gc_pool": "us-west.rgw.gc",
  "log_pool": "us-west.rgw.log",
  "intent_log_pool": "us-west.rgw.intent-log",
  "usage_log_pool": "us-west.rgw.usage",
  "user_keys_pool": "us-west.rgw.users.keys",
  "user_email_pool": "us-west.rgw.users.email",
  "user_swift_pool": "us-west.rgw.users.swift",
  "user_uid_pool": "us-west.rgw.users.uid",
  "system_key": {
      "access_key": "1555b35654ad1656d804",
      "secret_key": "h7GhxuBLTrlhVUyxSPUKUV8r\/2EI4ngqJxD7iBdBYLhwluN30JaT3Q=="
  },
```

```
  "placement_pools": [
      {
          "key": "default-placement",
          "val": {
              "index_pool": "us-west.rgw.buckets.index",
              "data_pool": "us-west.rgw.buckets.data",
              "data_extra_pool": "us-west.rgw.buckets.non-ec",
              "index_type": 0
          }
      }
  ],
  "metadata_heap": "us-west.rgw.meta",
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
}
```

11.11.11.3 更新周期

为了传播区域组地图更改，我们将更新并提交周期：

```
cephadm > radosgw-admin period update --commit --rgw-zone=us-west
{
  "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
  "epoch": 3,
  "predecessor_uuid": "09559832-67a4-4101-8b3f-10dfcd6b2707",
  "sync_status": [
      "", # truncated
  ],
  "period_map": {
      "id": "b5e4d3ec-2a62-4746-b479-4b2bc14b27d1",
      "zonegroups": [
          {
              "id": "d4018b8d-8c0d-4072-8919-608726fa369e",
              "name": "us",
              "api_name": "us",
              "is_master": "true",
              "endpoints": [
                  "http:\/\/rgw1:80"
              ],
```

```
            "hostnames": [],
            "hostnames_s3website": [],
            "master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "zones": [
                {
                    "id": "83859a9a-9901-4f00-aa6d-285c777e10f0",
                    "name": "us-east-1",
                    "endpoints": [
                        "http:\/\/rgw1:80"
                    ],
                    "log_meta": "true",
                    "log_data": "true",
                    "bucket_index_max_shards": 0,
                    "read_only": "false"
                },
                                {
                    "id": "950c1a43-6836-41a2-a161-64777e07e8b8",
                    "name": "us-east-2",
                    "endpoints": [
                        "http:\/\/rgw2:80"
                    ],
                    "log_meta": "false",
                    "log_data": "true",
                    "bucket_index_max_shards": 0,
                    "read_only": "false"
                },
                {
                    "id": "d9522067-cb7b-4129-8751-591e45815b16",
                    "name": "us-west",
                    "endpoints": [
                        "http:\/\/rgw3:80"
                    ],
                    "log_meta": "false",
                    "log_data": "true",
                    "bucket_index_max_shards": 0,
                    "read_only": "false"
                }
            ],
```

```
            "placement_targets": [
                {
                    "name": "default-placement",
                    "tags": []
                }
            ],
            "default_placement": "default-placement",
            "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7"
        }
    ],
    "short_zone_ids": [
        {
            "key": "83859a9a-9901-4f00-aa6d-285c777e10f0",
            "val": 630926044
        },
        {
            "key": "950c1a43-6836-41a2-a161-64777e07e8b8",
            "val": 4276257543
        },
        {
            "key": "d9522067-cb7b-4129-8751-591e45815b16",
            "val": 329470157
        }
    ]
},
"master_zonegroup": "d4018b8d-8c0d-4072-8919-608726fa369e",
"master_zone": "83859a9a-9901-4f00-aa6d-285c777e10f0",
"period_config": {
    "bucket_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    },
    "user_quota": {
        "enabled": false,
        "max_size_kb": -1,
        "max_objects": -1
    }
```

```
  },
  "realm_id": "4a367026-bd8f-40ee-b486-8212482ddcd7",
  "realm_name": "gold",
  "realm_epoch": 2
}
```

请注意，周期的版本号已递增，表示配置发生了更改。

11.11.11.4 启动对象网关

此操作与在第一个区域中启动对象网关类似。唯一的差别在于，对象网关区域配置应反映 `us-west` 区域名称：

```
[client.rgw.us-west]
rgw_frontends="civetweb port=80"
rgw_zone=us-west
```

启动第二个对象网关：

```
sudo systemctl start ceph-radosgw@rgw.us-west
```

11.11.12 故障转移和灾难恢复

如果主区域发生故障，将故障转移到次要区域，以实现灾难恢复。

1. 将次要区域设为主区域和默认区域。例如：

   ```
   root # radosgw-admin zone modify --rgw-zone={zone-name} --master --default
   ```

   默认情况下，Ceph Object Gateway 将以主动/主动配置运行。如果已将集群配置为以主动/被动配置运行，则次要区域是只读区域。删除 --read-only 状态可让区域接收写入操作。例如：

   ```
   root # radosgw-admin zone modify --rgw-zone={zone-name} --master --default
    \
   --read-only=False
   ```

2. 更新周期，使更改生效。

```
root # radosgw-admin period update --commit
```

3. 最后，重启动 Ceph Object Gateway。

```
root # systemctl restart ceph-radosgw@rgw.`hostname -s`
```

如果之前的主区域已恢复，请逆向操作。

1. 在已恢复的区域中，从当前主区域提取周期。

```
root # radosgw-admin period pull --url={url-to-master-zone-gateway} \
--access-key={access-key} --secret={secret}
```

2. 将已恢复的区域设为主区域和默认区域。

```
root # radosgw-admin zone modify --rgw-zone={zone-name} --master --default
```

3. 更新周期，使更改生效。

```
root # radosgw-admin period update --commit
```

4. 然后，在已恢复的区域中重启动 Ceph Object Gateway。

```
root # systemctl restart ceph-radosgw@rgw.`hostname -s`
```

5. 如果次要区域需要采用只读配置，请更新次要区域。

```
root # radosgw-admin zone modify --rgw-zone={zone-name} --read-only
```

6. 更新周期，使更改生效。

```
root # radosgw-admin period update --commit
```

7. 最后，在次要区域中重启动 Ceph Object Gateway。

```
root # systemctl restart ceph-radosgw@rgw.`hostname -s`
```

11.12 使用 HAProxy 在对象网关服务器间实现负载平衡

您可以使用 HAProxy 负载平衡程序将所有请求分布在多个对象网关后端服务器之间。有关配置 HAProxy 的详细信息，请参见https://www.suse.com/documentation/sle-ha-12/book_sleha/data/sec_ha_lb_haproxy.html 。

下面是一种 HAProxy 的简单配置，使用循环复用平衡算法来平衡对象网关节点：

```
root # cat /etc/haproxy/haproxy.cfg
[...]
frontend https_frontend
bind *:443 crt path-to-cert.pem [ciphers: ... ]
default_backend rgw

backend rgw
mode http
balance roundrobin
server rgw_server1 rgw-endpoint1 weight 1 maxconn 100 check
server rgw_server2 rgw-endpoint2 weight 1 maxconn 100 check
[...]
```

12 Ceph iSCSI 网关

本章重点介绍与 iSCSI 网关相关的管理任务。有关部署过程，请参见《部署指南》, 第 10 章 “安装 iSCSI 网关”。

12.1 连接 lrbd 管理的目标

本节介绍如何从运行 Linux、Microsoft Windows 或 VMware 的客户端连接 lrdb 管理的目标。

12.1.1 Linux (`open-iscsi`)

使用 `open-iscsi` 连接 lrbd 支持的 iSCSI 目标需要执行两个步骤。首先，发起程序必须发现网关主机上可用的 iSCSI 目标，然后，必须登录并映射可用的逻辑单元 (LU)。

这两个步骤都需要 `open-iscsi` 守护进程处于运行状态。启动 `open-iscsi` 守护进程的方式取决于您的 Linux 发行套件：

- 在 SUSE Linux Enterprise Server (SLES) 和 Red Hat Enterprise Linux (RHEL) 主机上，运行 `systemctl start iscsid`（如果 `systemctl` 不可用，请运行 `service iscsid start`）。
- 在 Debian 和 Ubuntu 主机上，运行 `systemctl start open-iscsi`（或 `service open-iscsi start`）。

如果发起程序主机运行 SUSE Linux Enterprise Server，请参见 https://www.suse.com/documentation/sles-12/stor_admin/data/sec_iscsi_initiator.html 或 https://www.suse.com/documentation/sles11/stor_admin/data/sec_inst_system_iscsi_initiator.html，了解有关如何连接 iSCSI 目标的详细信息。

对于支持 `open-iscsi` 的其他 Linux 发行套件，请继续发现 `lrbd` 网关上的目标（本示例使用 iscsi1.example.com 作为门户地址；对于多路径访问，请使用 iscsi2.example.com 重复这些步骤）：

```
iscsiadm -m discovery -t sendtargets -p iscsi1.example.com
```

```
192.168.124.104:3260,1 iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol
```

然后登录该门户。如果登录成功完成，则门户中所有基于 RBD 的逻辑单元将立即在系统 SCSI 总线上变为可用：

```
iscsiadm -m node -p iscsi1.example.com --login
Logging in to [iface: default, target: iqn.2003-01.org.linux-
iscsi.iscsi.x86:testvol, portal: 192.168.124.104,3260] (multiple)
Login to [iface: default, target: iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol,
 portal: 192.168.124.104,3260] successful.
```

针对其他门户 IP 地址或主机重复此过程。

如果系统上已安装 `lsscsi` 实用程序，您可以使用它来枚举系统上可用的 SCSI 设备：

```
lsscsi
[8:0:0:0]    disk    SUSE     RBD              4.0   /dev/sde
[9:0:0:0]    disk    SUSE     RBD              4.0   /dev/sdf
```

在多路径配置（其中两个已连接的 iSCSI 设备代表一个相同的 LU）中，您还可以使用 `multipath` 实用程序检查多路径设备状态：

```
multipath -ll
360014050cf9dcfcb2603933ac3298dca dm-9 SUSE,RBD
size=49G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 8:0:0:0 sde 8:64 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
`- 9:0:0:0 sdf 8:80 active ready running
```

现在，可根据需要，将此多路径设备用作任何块设备。例如，可将该设备用作 Linux 逻辑卷管理 (LVM) 的物理卷，或者直接在该设备上创建文件系统。下面的示例说明如何在新连接的多路径 iSCSI 卷上创建 XFS 文件系统：

```
mkfs -t xfs /dev/mapper/360014050cf9dcfcb2603933ac3298dca
log stripe unit (4194304 bytes) is too large (maximum is 256KiB)
log stripe unit adjusted to 32KiB
meta-data=/dev/mapper/360014050cf9dcfcb2603933ac3298dca isize=256    agcount=17,
 agsize=799744 blks
```

```
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0
data     =                       bsize=4096   blocks=12800000, imaxpct=25
         =                       sunit=1024   swidth=1024 blks
naming   =version 2              bsize=4096   ascii-ci=0 ftype=0
log      =internal log           bsize=4096   blocks=6256, version=2
         =                       sectsz=512   sunit=8 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

请注意，由于 XFS 是非集群文件系统，无论何时，您都只能将它挂载到单个 iSCSI 发起程序节点上。

任何时候如果要停止使用与特定目标关联的 iSCSI LU，请运行以下命令：

```
iscsiadm -m node -p iscsi1.example.com --logout
Logging out of session [sid: 18, iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol,
 portal: 192.168.124.104,3260]
Logout of [sid: 18, target: iqn.2003-01.org.linux-iscsi.iscsi.x86:testvol,
 portal: 192.168.124.104,3260] successful.
```

与执行发现和登录时一样，必须针对所有门户 IP 地址或主机名重复注销步骤。

12.1.1.1 多路径配置

多路径配置保留在客户端或发起程序上，并依赖于任何 `lrbd` 配置。在使用块存储之前，请选择一个策略。编辑 `/etc/multipath.conf` 之后，请使用以下命令重启动 `multipathd`

```
sudo systemctl restart multipathd
```

对于包含友好名称的主动/被动配置，请将

```
defaults {
  user_friendly_names yes
}
```

添加到 `/etc/multipath.conf` 。成功连接目标后，请运行

```
multipath -ll
```

```
mpathd (36001405dbb561b2b5e439f0aed2f8e1e) dm-0 SUSE,RBD
size=2.0G features='0' hwhandler='0' wp=rw
|-+- policy='service-time 0' prio=1 status=active
| `- 2:0:0:3 sdl 8:176 active ready running
|-+- policy='service-time 0' prio=1 status=enabled
| `- 3:0:0:3 sdj 8:144 active ready running
`-+- policy='service-time 0' prio=1 status=enabled
  `- 4:0:0:3 sdk 8:160 active ready running
```

注意每个链路的状态。对于主动/主动配置，请将

```
defaults {
  user_friendly_names yes
}

devices {
  device {
    vendor "(LIO-ORG|SUSE)"
    product "RBD"
    path_grouping_policy "multibus"
    path_checker "tur"
    features "0"
    hardware_handler "1 alua"
    prio "alua"
    failback "immediate"
    rr_weight "uniform"
    no_path_retry 12
    rr_min_io 100
  }
}
```

添加到 `/etc/multipath.conf` 。重启动 `multipathd` 并运行

```
multipath -ll
mpathd (36001405dbb561b2b5e439f0aed2f8e1e) dm-3 SUSE,RBD
size=2.0G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 4:0:0:3 sdj 8:144 active ready running
  |- 3:0:0:3 sdk 8:160 active ready running
```

```
`- 2:0:0:3 sdl 8:176 active ready running
```

12.1.2 Microsoft Windows（Microsoft iSCSI 发起程序）

要从 Windows 2012 服务器连接 SUSE Enterprise Storage iSCSI 目标，请执行以下步骤：

1. 打开 Windows 服务器管理器。在仪表盘中，选择 Tools（工具）› iSCSI Initiator（iSCSI 发起程序）。iSCSI Initiator Properties（iSCSI 发起程序属性）对话框随即显示。选择 Discovery（发现）选项卡：

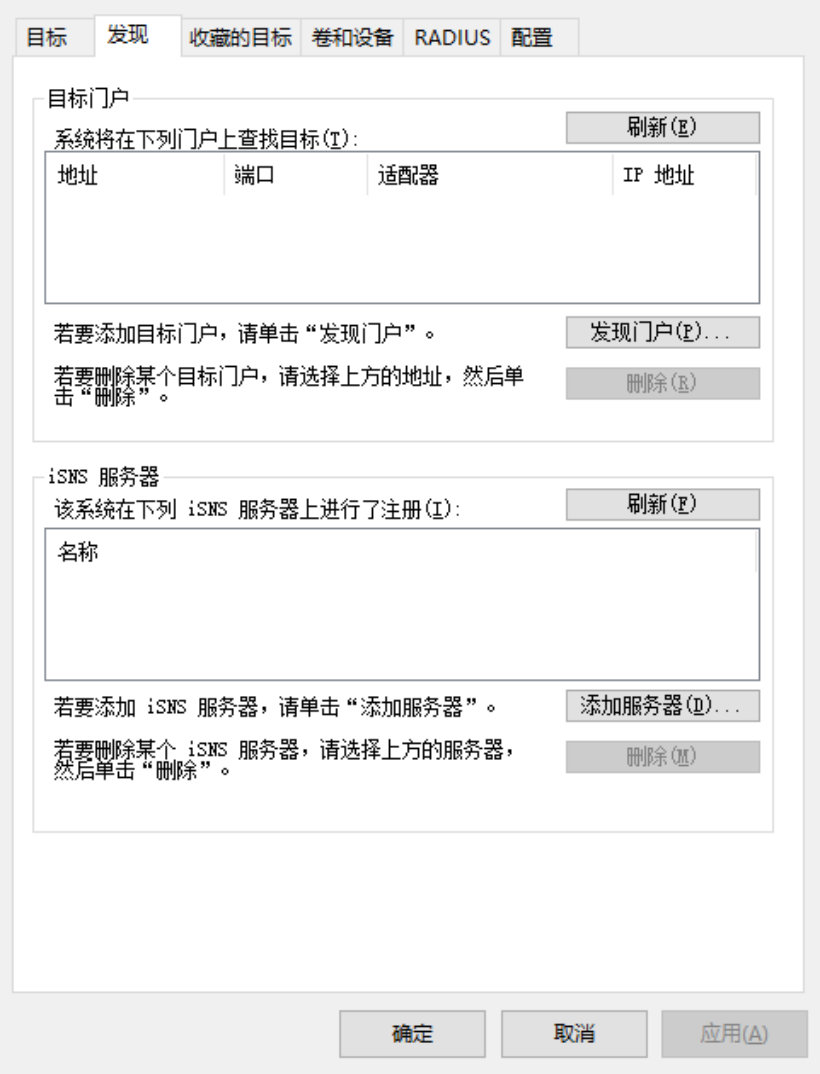


图 12.1： ISCSI 发起程序属性

2. 在 Discover Target Portal（发现目标门户）对话框中的 Target（目标）字段内，输入目标的主机名或 IP 地址，然后点击 OK（确定）：

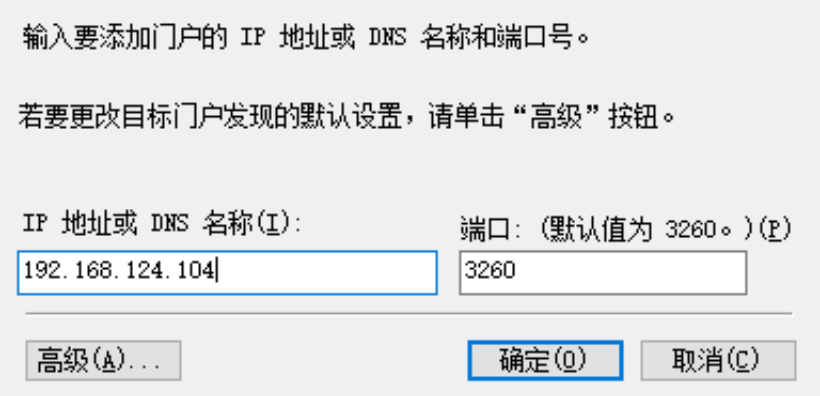


图 12.2： 发现目标门户

3. 针对所有其他网关主机名或 IP 地址重复此过程。完成后，查看 Target Portals（目标门户）列表：

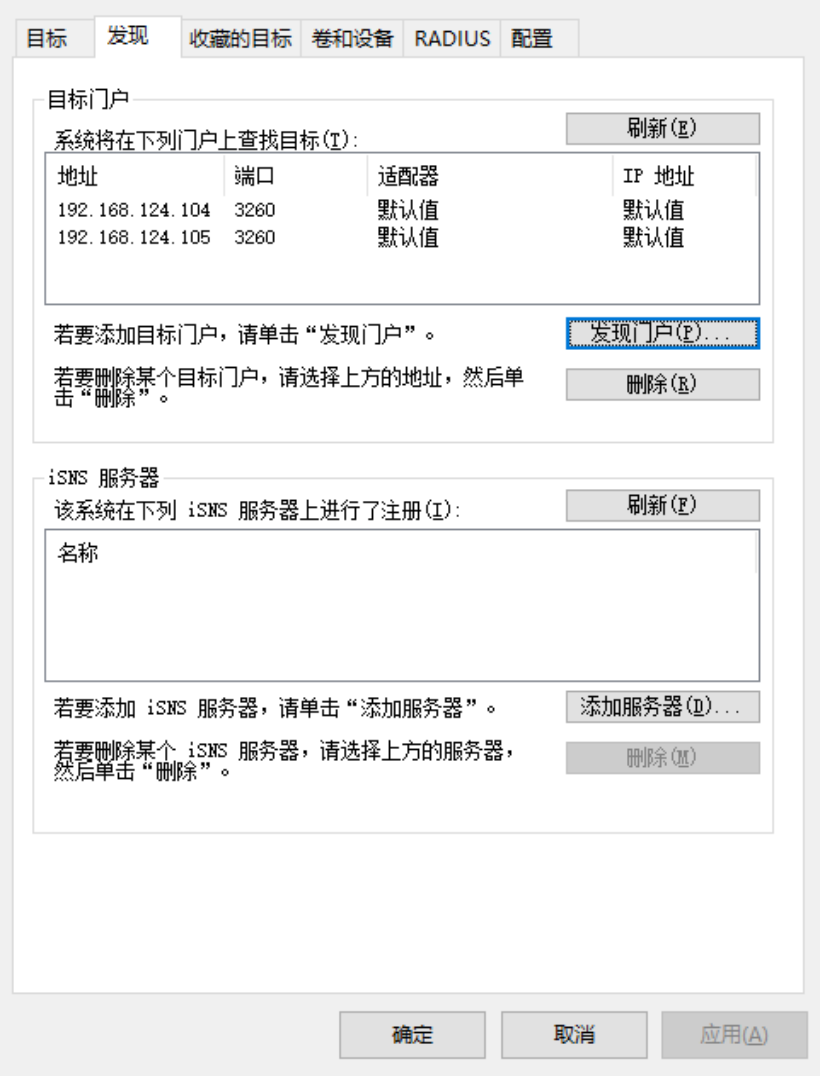


图 12.3： 目标门户

4. 接下来，切换到 Targets（目标）选项卡并查看已发现的目标。

图 12.4： 目标

5. 在 Targets（目标）选项卡中点击 Connect（连接）。Connect To Target（连接目标）对话框随即显示。选中 Enable Multi-path（启用多路径）复选框以启用多路径 I/O (MPIO)，然后点击 OK（确定）：

6. Connect to Target（连接目标）对话框关闭后，选择 Properties（属性）查看目标的属性：

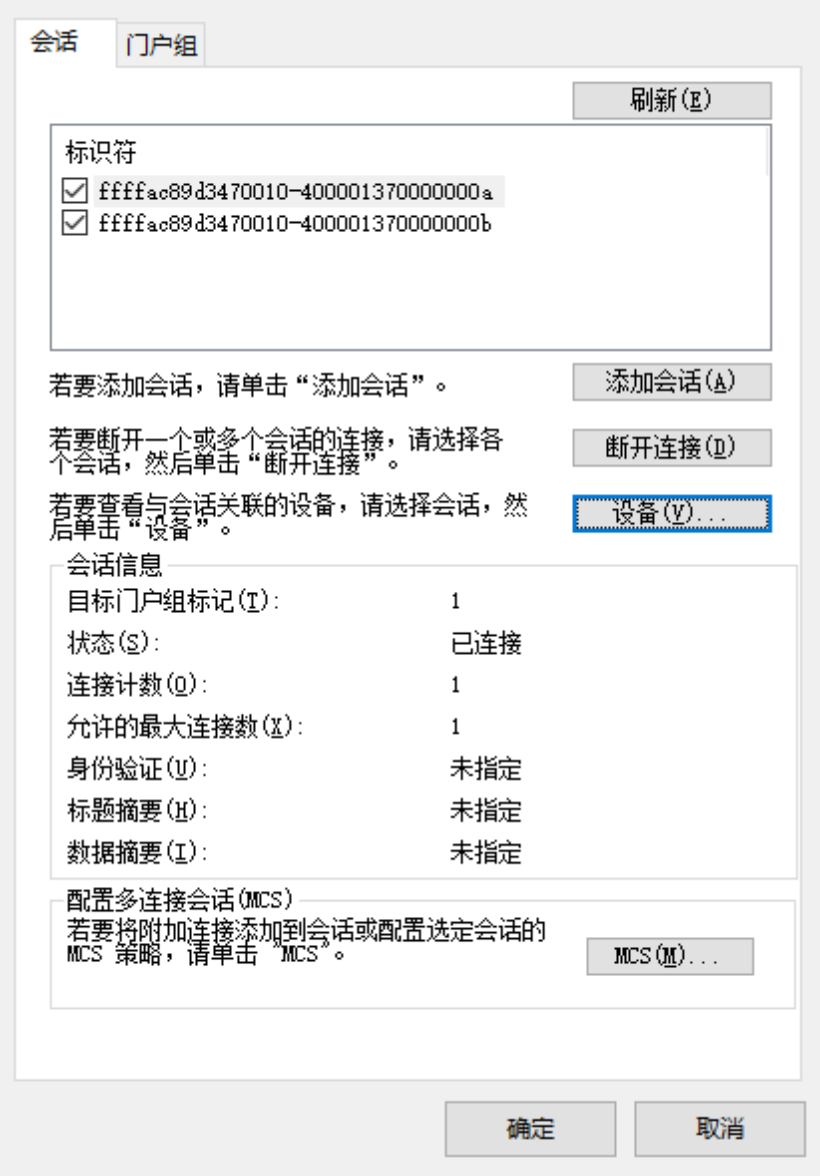


图 12.5： ISCSI 目标属性

7. 选择 Devices（设备），然后点击 MPIO 查看多路径 I/O 配置：

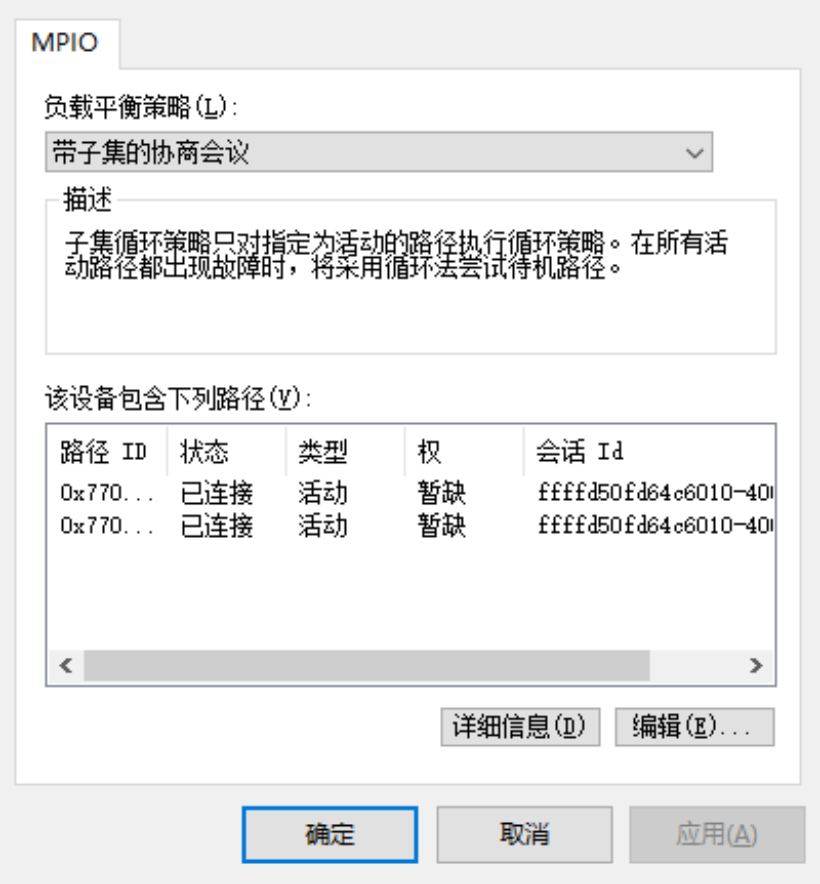


图 12.6： 设备详细信息

默认的负载平衡策略为 Round Robin With Subset（带子集的循环法）。如果您偏向于单纯的故障转移配置，请将策略更改为 Fail Over Only（仅故障转移）。

iSCSI 发起程序的配置到此结束。现在，可以像使用任何其他 SCSI 设备一样使用 iSCSI 卷，并可将其初始化，使其可用作卷和驱动器。点击 OK（确定）关闭 iSCSI Initiator Properties（iSCSI 发起程序属性）对话框，然后继续在 Server Manager（服务器管理器）仪表盘中配置 File and Storage Services（文件和存储服务）角色。

观察新连接的卷。该卷标识为 iSCSI 总线上的 SUSE RBD SCSI 多路径驱动器，并且最初标记为脱机状态，其分区表类型为未知。如果新卷未立即显示，请从 Task（任务）下拉框中选择 Rescan Storage（重新扫描存储），以重新扫描 iSCSI 总线。

1. 右键点击 iSCSI 卷，然后从上下文菜单中选择 New Volume（新建卷）。New Volume Wizard（新建卷向导）随即显示。点击 Next（下一步），突出显示新连接的 iSCSI 卷，然后点击 Next（下一步）开始创建新卷。

图 12.7： 新建卷向导

2. 该设备最初是空的，不包含任何分区表。当出现对话框指出将要使用 GPT 分区表初始化卷时，确认该操作：

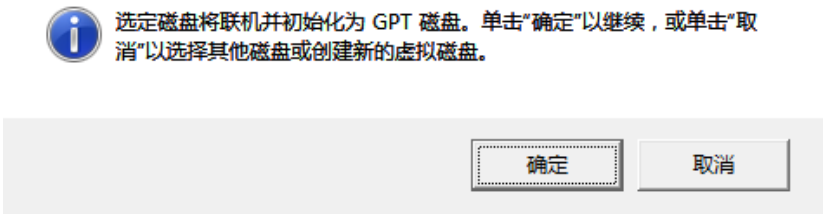


图 12.8： 脱机磁盘提示

3. 选择卷大小。通常，用户会使用设备的全部容量。然后，指定新建卷将在其上变为可用状态的驱动器盘符或目录名称。接下来，选择要在新卷上创建的文件系统。最后，点击 Create（创建）确认所做的选择并完成卷的创建：

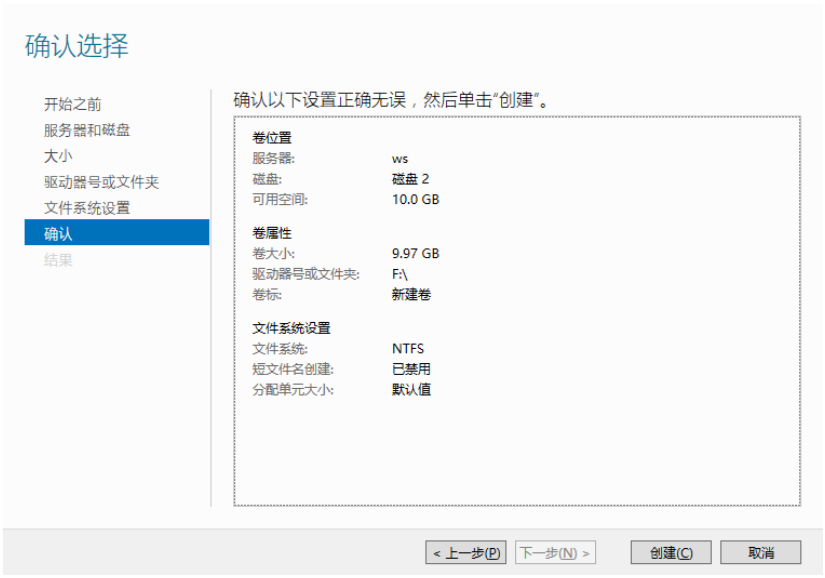


图 12.9： 确认选择的卷设置

完成该过程后，请检查结果，然后点击 Close（关闭）结束驱动器初始化。完成初始化后，便可以像使用新初始化的本地驱动器一样使用该卷（及其 NTFS 文件系统）。

12.1.3 VMware

1. 要连接到 `lrbd` 管理的 iSCSI 卷，需要一个经过配置的 iSCSI 软件适配器。如果 vSphere 配置中未提供此类适配器，请选择 Configuration（配置）› Storage Adapters（存储适配器）› Add（增加）› iSCSI Software initiator（iSCSI 软件发起程序）来创建一个适配器。
2. 如果适用，请通过右键点击该适配器并从上下文菜单中选择 Properties（属性），来选择该适配器的属性：

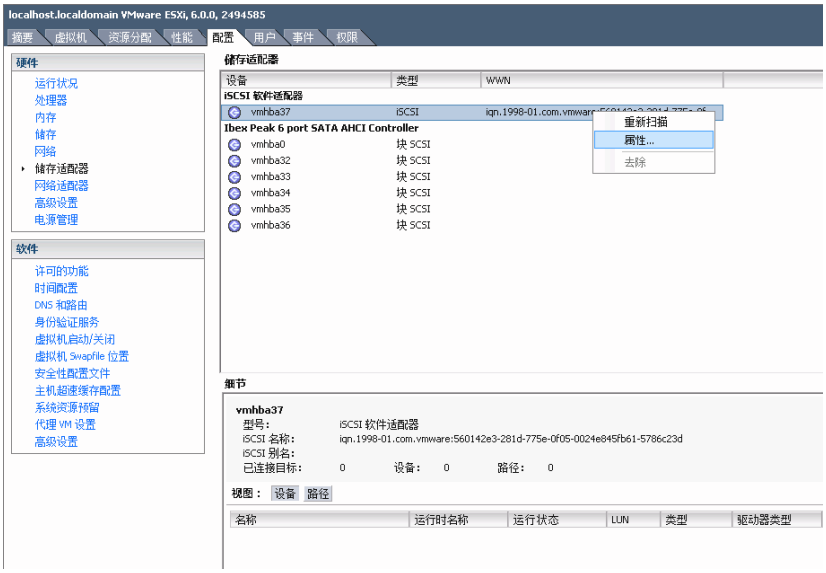


图 12.10： ISCSI 发起程序属性

3. 在 iSCSI Software Initiator（iSCSI 软件发起程序）对话框中，点击 Configure（配置）按钮。然后转到 Dynamic Discovery（动态发现）选项卡并选择 Add（增加）。

4. 输入 `lrbd` iSCSI 网关的 IP 地址或主机名。如果在故障转移配置中运行多个 iSCSI 网关，请针对要运行的所有网关重复此步骤。

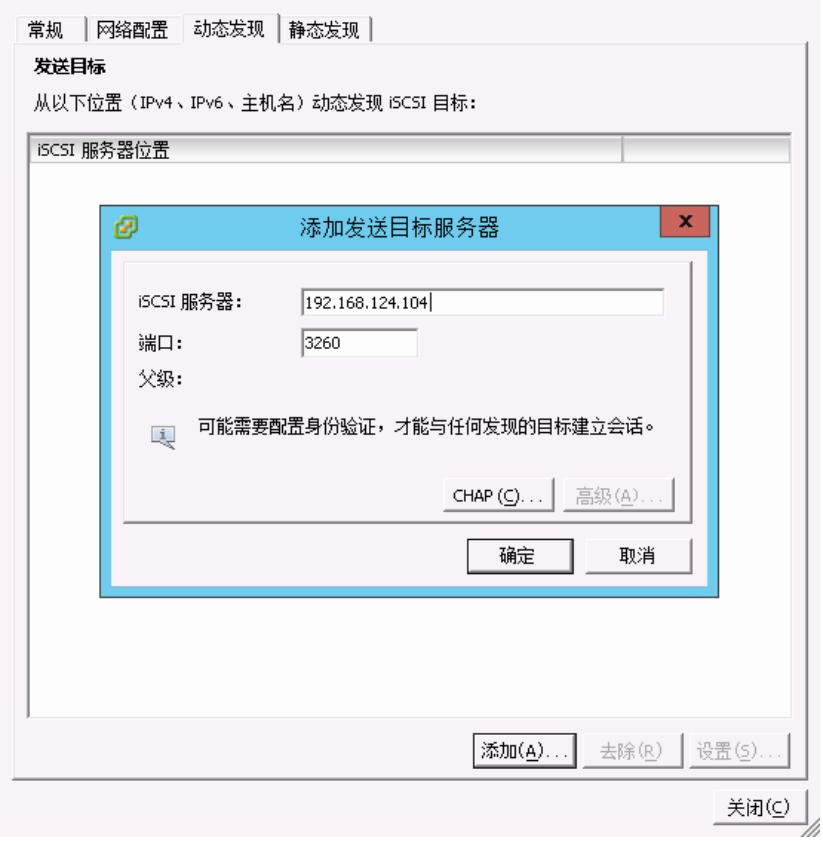


图 12.11： 添加目标服务器

输入所有 iSCSI 网关后，请在对话框中点击 OK（确定），发起对 iSCSI 适配器的重新扫描。

5. 重新扫描完成后，新的 iSCSI 设备会显示在 Details（详细信息）窗格中的 Storage Adapters（存储适配器）列表下。对于多路径设备，现在可以右键点击该适配器，然后从上下文菜单中选择 Manage Paths（管理路径）：

图 12.12： 管理多路径设备

您现在应该会看到，所有路径的 Status（状态）下面都带有绿灯。其中一个路径应该已标记为 Active (I/O)（主动 (I/O)），其他所有路径只是标记为 Active（主动）：

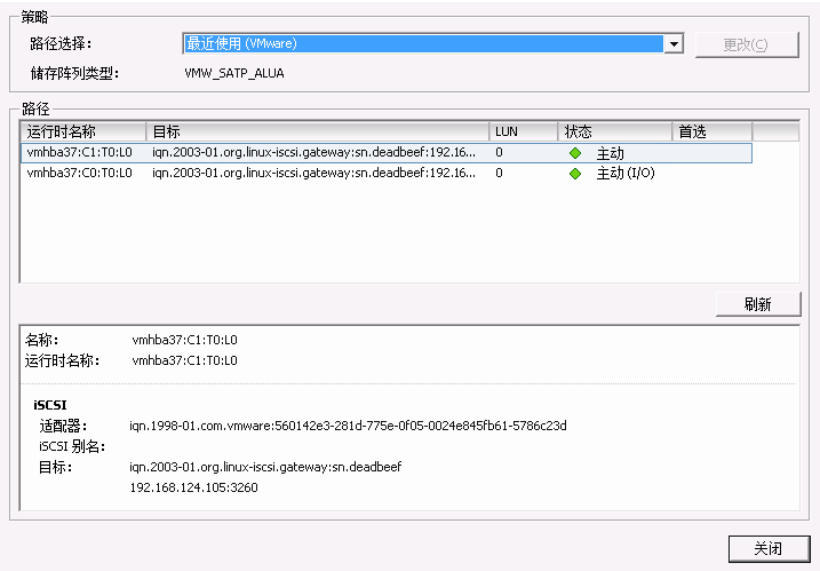


图 12.13： 多路径的路径列表

6. 现在，您可以从 Storage Adapters（存储适配器）切换到标为 Storage（存储）的项目。在窗格右上角选择 Add Storage...（添加存储...）打开 Add Storage（添加存储）对话框。然后选择 Disk/LUN（磁盘/LUN）并点击 Next（下一步）。新添加的 iSCSI 设备会显示在 选择磁盘/LUN（Select Disk/LUN）列表中。选择该设备，然后点击 Next（下一步）继续：

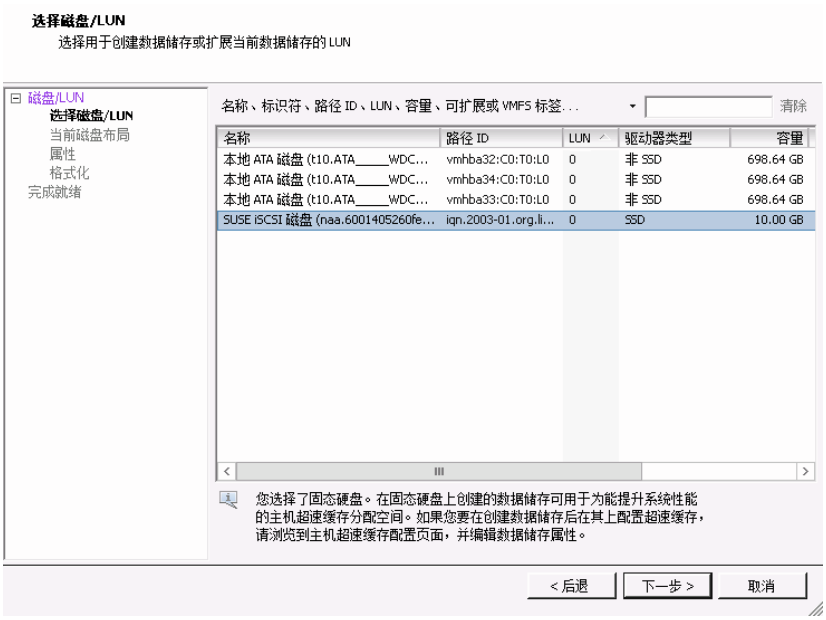


图 12.14： “ADD STORAGE”（添加存储）对话框

点击 Next（下一步）接受默认的磁盘布局。

7. 在 Properties（属性）窗格中，为新数据存储指定名称，然后点击 Next（下一步）。接受将卷的整个空间用于数据存储的默认设置，或者选择 Custom Space Setting（自定义空间设置）以创建较小的数据存储：

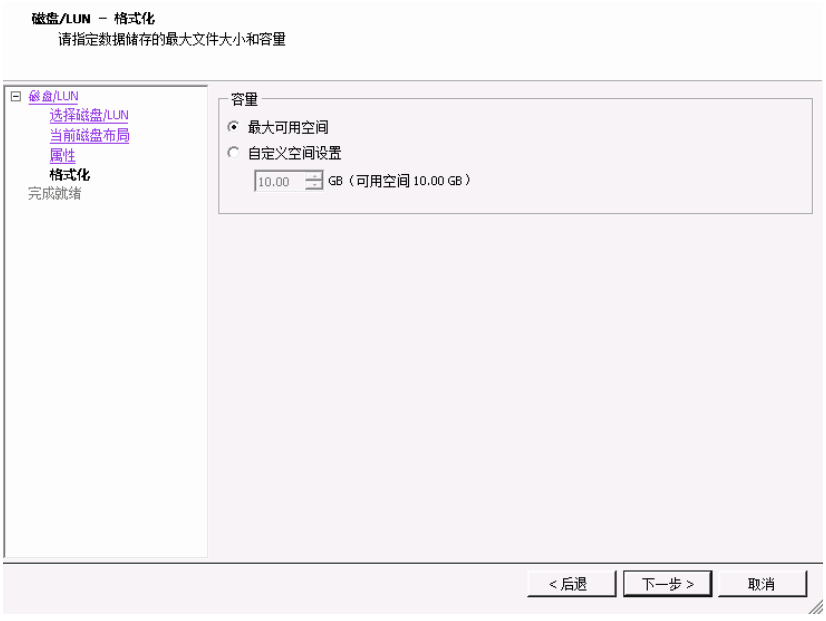


图 12.15： 自定义空间设置

点击 Finish（完成）以完成数据存储的创建。

新数据存储现在即会显示在数据存储列表中，您可以选择它来检索详细信息。现在，您可以像使用任何其他 vSphere 数据存储一样使用基于 `lrbd` 的 iSCSI 卷。

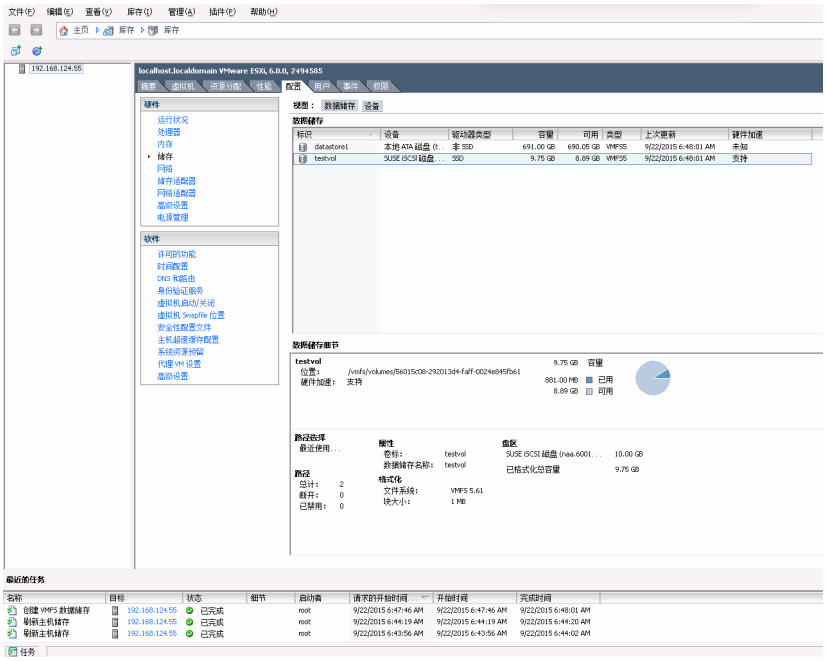


图 12.16： ISCSI 数据存储概述

12.2 结论

`lrbd` 是 SUSE Enterprise Storage 的一个关键组件，使用该组件可以通过支持 iSCSI 协议的任何服务器或客户端访问高度可用的分布式块存储。在一个或多个 iSCSI 网关主机上使用 `lrbd`，可将 Ceph RBD 映像用作与 iSCSI 目标关联的逻辑单元 (LU)，并可根据需要以负载平衡且高度可用的方式来访问该逻辑单元。

由于 `lrbd` 的所有配置都存储在 Ceph RADOS 对象存储中，`lrbd` 网关主机先天就不具有持久性状态，因而可以任意对其进行更换或者增减。因此，SUSE Enterprise Storage 可让 SUSE 客户在市售硬件和完全开源的平台上运行真正的分布式、高度可用、有弹性且可自我修复的企业存储技术。

13 集群文件系统

本章介绍在设置集群并导出 CephFS 后通常应执行的管理任务。如需有关设置 CephFS 的详细信息，请参见《部署指南》，第 11 章 “安装 CephFS”。

13.1 挂载 CephFS

创建文件系统后，如果 MDS 可用，您便可以从客户端主机挂载文件系统。

13.1.1 客户端准备

如果客户端主机运行的是 SUSE Linux Enterprise 12 SP2 或 SP3，您可以跳过本节，因为系统无需额外配置即可挂载 CephFS。

如果客户端主机运行的是 SUSE Linux Enterprise 12 SP1，您需要应用所有最新的增补程序，之后才能挂载 CephFS。

无论是哪一种情况，SUSE Linux Enterprise 中都包含了挂载 CephFS 需要的所有项目。不需要 SUSE Enterprise Storage 产品。

为了支持完整的 `mount` 语法，在尝试挂载 CephFS 之前，应该先安装 `ceph-common` 包（随附于 SUSE Linux Enterprise 中）。

13.1.2 创建机密文件

Ceph 集群默认是在启用身份验证的情况下运行的。应该创建一个文件用于存储您的机密密钥（而不是密钥环本身）。要获取特定用户的机密密钥，然后创建该文件，请执行以下操作：

过程 13.1： 创建机密密钥

1. 在密钥环文件中查看特定用户的密钥：

```
cat /etc/ceph/ceph.client.admin.keyring
```

2. 复制要使用所挂载 Ceph FS 文件系统的用户的密钥。密钥通常类似下方所示：

```
AQCj2YpRiAe6CxAA7/ETt7Hcl9IyxyYciVs47w==
```

3. 为用户 admin 创建一个文件名包含用户名的文件，例如 `/etc/ceph/admin.secret`。
4. 将密钥值粘贴到上一步中创建的文件。
5. 设置对该文件的适当访问权限。该用户应是唯一有权读取该文件的用户，其他人不能有任何访问权限。

13.1.3 挂载 CephFS

可以使用 `mount` 命令挂载 CephFS。需要指定监视器的主机名或 IP 地址。由于 SUSE Enterprise Storage 中默认会启用 `cephx` 身份验证，因此，您还需要指定一个用户名及其相关的机密：

```
sudo mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \
 -o name=admin,secret=AQATSKdNGBnwLhAAnNDKnH65FmVKpXZJVasUeQ==
```

由于上一条命令会保留在外壳历史中，因此更安全的做法是从文件读取机密：

```
sudo mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \
 -o name=admin,secretfile=/etc/ceph/admin.secret
```

请注意，机密文件应该只包含实际的密钥环机密。因此，在本示例中，该文件只包含下行：

```
AQATSKdNGBnwLhAAnNDKnH65FmVKpXZJVasUeQ==
```

提示：指定多个监视器

最好是在 `mount` 命令行中指定多个监视器并以逗号分隔，以防在挂载时某个监视器恰好停机。每个监视器的地址采用 `主机[:端口]` 格式。如果未指定端口，默认会使用端口 6789。

在本地主机上创建挂载点：

```
sudo mkdir /mnt/cephfs
```

挂载 CephFS：

```
sudo mount -t ceph ceph_mon1:6789:/ /mnt/cephfs \
 -o name=admin,secretfile=/etc/ceph/admin.secret
```

如果要挂载文件系统的某个子集，可以指定子目录 `subdir`：

```
sudo mount -t ceph ceph_mon1:6789:/subdir /mnt/cephfs \
 -o name=admin,secretfile=/etc/ceph/admin.secret
```

可在 `mount` 命令中指定多个监视器主机：

```
sudo mount -t ceph ceph_mon1,ceph_mon2,ceph_mon3:6789:/ /mnt/cephfs \
 -o name=admin,secretfile=/etc/ceph/admin.secret
```

重要：对根目录的读取访问权限

如果使用了实施路径限制的客户端，则 MDS 功能需要包含对根目录的读取访问权限。例如，密钥环可能如下所示：

```
client.bar
 key: supersecretkey
 caps: [mds] allow rw path=/barjail, allow r path=/
 caps: [mon] allow r
 caps: [osd] allow rwx
```

`allow r path=/` 部分表示路径受限的客户端能够查看根卷，但无法写入根卷。在要求完全隔离的用例中，这可能会造成问题。

13.2 卸载 CephFS

要卸载 CephFS，请使用 `umount` 命令：

```
sudo umount /mnt/cephfs
```

13.3 /etc/fstab 中的 CephFS

要在客户端启动时自动挂载 CephFS，请在其文件系统表 `/etc/fstab` 中插入相应的行：

```
mon1:6790,mon2:/subdir /mnt/cephfs ceph name=admin,secretfile=/etc/ceph/
secret.key,noatime,_netdev 0 2
```

13.4 多个活动 MDS 守护进程（主动/主动 MDS）

默认情况下，CephFS 是针对单个活动 MDS 守护进程配置的。要调整大规模系统的元数据性能，可以启用多个活动 MDS 守护进程，以便互相分担元数据工作负载。

13.4.1 何时使用主动/主动 MDS

如果按默认设置使用单个 MDS 时元数据性能出现瓶颈，可考虑使用多个活动 MDS 守护进程。

增加守护进程并不会提高所有工作负载类型的性能。例如，增加 MDS 守护进程的数量不会让单个客户端上运行的单个应用受益，除非该应用在同时执行大量元数据操作。

通常能够因大量活动 MDS 守护进程受益的工作负载是使用许多客户端的工作负载，也许是在许多独立目录中工作的工作负载。

13.4.2 增加 MDS 活动集群的大小

每个 CephFS 文件系统都有一项 `max_mds` 设置，用于控制将要创建的级别数。仅当某个备用守护进程可供新的级别使用时，文件系统中的实际级别数才会增加。例如，如果只有一个 MDS 守护进程在运行，并且 `max_mds` 设置为 2，将不会创建另一个级别。

在下面的示例中，我们将 `max_mds` 选项设置为 2，以便在保留默认级别的情况下再创建一个新级别。要查看更改，请在设置 `max_mds` 之前和之后运行 `ceph status`，然后观察包含 `fsmap` 的行：

```
root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-1/1/1 up  {0=node2=up:active}, 1 up:standby
    [...]
root@master # ceph mds set max_mds 2
```

```
root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-2/2/2 up  {0=node2=up:active,1=node1=up:active}
    [...]
```

新建的级别 (1) 会经历“正在创建”状态，然后进入“活动”状态。

重要：待机守护进程

即使使用多个活动 MDS 守护进程，当任何在运行活动守护进程的服务器发生故障时，高可用性系统也仍会要求待机守护进程接管工作。

因此，高可用性系统的 `max_mds` 合理最大值比系统中的 MDS 服务器总数小 1。要在发生多次服务器故障时保持可用性，可增加系统中待机守护进程的数量，使之与不会导致失去可用性的服务器故障数一致。

13.4.3 减小级别数

所有级别（包括要删除的级别）首先必须是活动的。这意味着，至少需要有 `max_mds` 个 MDS 守护进程可用。

首先，将 `max_mds` 设为一个较小的数字。例如，我们重新使用单个活动 MDS：

```
root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-2/2/2 up  {0=node2=up:active,1=node1=up:active}
    [...]
root@master # ceph mds set max_mds 1
root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-1/1/1 up  {0=node2=up:active}, 1 up:standby
    [...]
```

请注意，我们仍有两个活动 MDS。即使减小 `max_mds`，级别也仍会存在，因为 `max_mds` 只会限制新级别的创建。

接下来，使用 `ceph mds deactivate 级别` 命令删除不需要的级别：

```
root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-2/2/1 up  {0=node2=up:active,1=node1=up:active}
root@master # ceph mds deactivate 1
telling mds.1:1 192.168.58.101:6805/2799214375 to deactivate

root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-2/2/1 up  {0=node2=up:active,1=node1=up:stopping}

root@master # ceph status
  [...]
  services:
    [...]
    mds: cephfs-1/1/1 up  {0=node2=up:active}, 1 up:standby
```

已停用的级别首先会进入“正在停止”状态并保持一段时间，期间它会将所分担的元数据负载转移给其余活动守护进程。此阶段可能需要数秒到数分钟时间。如果 MDS 看上去停滞在“正在停止”状态，则应该调查原因，确定是否存在可能的错误。

如果 MDS 守护进程在“正在停止”状态下崩溃或终止，待机守护进程会接管工作，级别将恢复为“活动”状态。当此守护进程重新运行后，您可以尝试再次将它停用。

守护进程结束“正在停止”状态后，将再次启动，并重新变为待机守护进程。

13.4.4 手动将目录树关联到级别

在多个活动元数据服务器配置中，将会运行一个平衡器，用于在集群中均衡分配元数据负载。这种模式通常足以满足大多数用户的需求，但有时，用户需要使用元数据到特定级别的显式映射来覆盖动态平衡器。这样，管理员或用户便可以在整个集群上均衡地分配应用负载，或限制用户的元数据请求对整个集群的影响。

针对此目的提供的机制称为“导出关联”。它是目录的扩展属性。此扩展属性名为 `ceph.dir.pin`。用户可以使用标准命令设置此属性：

```
setfattr -n ceph.dir.pin -v 2 /path/to/dir
```

扩展属性的值 (`-v`) 是要将目录子树指定到的级别。默认值 -1 表示不关联该目录。

目录导出关联继承自设置了导出关联的最近的父级。因此，对某个目录设置导出关联会影响该目录的所有子级。但是，可以通过设置子目录导出关联来覆盖父级的关联。例如：

```
mkdir -p a/b                        # "a" and "a/b" start with no export pin set.
setfattr -n ceph.dir.pin -v 1 a/  # "a" and "b" are now pinned to rank 1.
setfattr -n ceph.dir.pin -v 0 a/b # "a/b" is now pinned to rank 0
                                    # and "a/" and the rest of its children
                                    # are still pinned to rank 1.
```

13.5 管理故障转移

如果 MDS 守护进程停止与监视器通讯，监视器会等待 `mds_beacon_grace` 秒（默认为 15 秒），然后将守护进程标记为 laggy。可以配置一个或多个“待机”守护进程，用于在 MDS 守护进程故障转移期间接管工作。

13.5.1 配置待机守护进程

有多项配置设置可控制守护进程处于待机状态时的行为。可以在运行 MDS 守护进程的主机上的 `ceph.conf` 中指定这些设置。守护进程在启动时会加载这些设置，然后将其发送到监视器。

默认情况下，如果不使用其中的任何设置，则不具备级别的所有 MDS 守护进程将用作任一级别的“待机”守护进程。

将待机守护进程与特定名称或级别相关联的设置不保证该守护进程只用于该级别。具体而言，当有多个待机守护进程可用时，将使用关联的待机守护进程。如果某个级别发生故障，而此时有某个待机守护进程可用，则即使该守护进程与其他某个级别或指定守护进程相关联，也会使用该守护进程。

mds_standby_replay

如果设置为 true，则待机守护进程将持续读取某个已启动级别的元数据日记。这就为此级别提供了一个热元数据快速缓存，当为该级别提供服务的守护进程发生故障时，此日记可加快故障转移过程的速度。

一个已启动的级别只能指定一个待机重放守护进程。如果将两个守护进程都设置为待机重放，则其中任意一个会赢得控制权，另一个将成为正常的非重放待机守护进程。

当某个守护进程进入待机重放状态时，它只会用作所跟随级别的待机守护进程。如果另一个级别发生故障，此待机重放守护进程不会作为替代者，即使没有其他待机守护进程可用也是如此。

mds_standby_for_name

如果指定此设置，则仅当最后一个包含故障级别的守护进程与此名称匹配时，待机守护进程才会接管该故障级别。

mds_standby_for_rank

如果指定此设置，待机守护进程只会接管指定的级别。如果另外的级别发生故障，将不会使用此守护进程来替代此级别。

与 `mds_standby_for_fscid` 结合使用时，可以指定在使用多个文件系统时，具体针对哪个文件系统的级别。

mds_standby_for_fscid

如果设置了 `mds_standby_for_rank`，则 mds_standby_for_fscid 只是一个用于指出所指文件系统级别的限定符。

如果未设置 `mds_standby_for_rank`，则设置 FSCID 会导致此守护进程以指定 FSCID 中的任何级别为目标。如果希望只在特定的文件系统中将某个守护进程用于任何级别，可使用此设置。

mon_force_standby_active

在监视器主机上使用此设置。其默认值为 true。

如果值为 false，则 `standby_replay 配置为 true` 的守护进程只有在其已配置要跟随的级别/名称发生故障时，才会变为活动守护进程。另一方面，如果此设置为 true，则可将其他某个级别指定给 `standby_replay 配置为 true` 的守护进程。

13.5.2 示例

下面显示了多个示例 `ceph.conf` 配置。可将包含所有守护进程的配置的 `ceph.conf` 复制到所有服务器，或者在每台服务器上创建一个不同的文件，并在其中包含该服务器的守护进程配置。

13.5.2.1 简单对

“a”和“b”两个 MDS 守护进程充当一对。其中，当前未指定级别的守护进程将是另一个守护进程的待机重放跟随者。

```
[mds.a]
mds standby replay = true
mds standby for rank = 0

[mds.b]
mds standby replay = true
mds standby for rank = 0
```

14 NFS Ganesha：通过 NFS 导出 Ceph 数据

NFS Ganesha 是一台 NFS 服务器（请参见与 NFS 共享文件系统 (https://www.suse.com/documentation/sles-12/book_sle_admin/data/cha_nfs.html)↗），它在用户地址空间中运行，而不是作为操作系统内核的一部分运行。借助 NFS Ganesha，您可以插入自己的存储机制（例如 Ceph），并从任何 NFS 客户端访问它。

系统按用户将 S3 桶导出到 NFS，例如，通过路径 `GANESHA_NODE:/用户名/桶名` 导出。

默认通过路径 `GANESHA_NODE:/cephfs` 导出 CephFS。

14.1 安装

有关安装说明，请参阅《部署指南》, 第 12 章 “安装 NFS Ganesha”。

14.2 配置

有关可在配置文件中使用的所有参数的列表，请参见：

- `man ganesha-config`
- `man ganesha-ceph-config`，用于 CephFS 文件系统抽象层 (FSAL) 选项。
- `man ganesha-rgw-config`，用于对象网关 FSAL 选项。

本节包含的信息可帮助您配置 NFS Ganesha 服务器，以导出可通过对象网关和 CephFS 访问的集群数据。

NFS Ganesha 配置通过 `/etc/ganesha/ganesha.conf` 控制。注意，对此文件所做的更改在执行 DeepSea 阶段 4 时会被覆盖。要永久更改这些设置，请编辑位于 Salt Master 上的文件 `/srv/salt/ceph/ganesha/files/ganesha.conf.j2`。

14.2.1 Export 段落

本节介绍如何配置 `ganesha.conf` 中的 `EXPORT` 段落。

```
EXPORT
{
  Export_Id = 1;
  Path = "/";
  Pseudo = "/";
  Access_Type = RW;
  Squash = No_Root_Squash;
  [...]
  FSAL {
    Name = CEPH;
  }
}
```

14.2.1.1 Export 主段落

Export_Id

每个导出项都需要有唯一的“Export_Id”（强制）。

Path

相关 CephFS 存储池中的导出项路径（强制）。允许从 CephFS 中导出子目录。

Pseudo

目标 NFS 导出项路径（对于 NFSv4 为强制）。它定义在哪个 NFS 导出项路径下可获得导出的数据。

示例：使用值 `/cephfs/` 并执行

```
root # mount GANESHA_IP:/cephfs/ /mnt/
```

之后，CephFS 数据可在客户端上的目录 `/mnt/cephfs/` 中获得。

Access_Type

“RO”表示只读访问权限，默认值是“None”。

Squash

NFS 匿名访问选项。

FSAL

导出“文件系统抽象层”。请参见第 14.2.1.2 节 “FSAL 子段落”。

14.2.1.2 FSAL 子段落

```
EXPORT
{
  [...]
  FSAL {
    Name = CEPH;
  }
}
```

Name

定义 NFS Ganesha 使用的后端。允许的值为 `CEPH`（表示 CephFS）或 `RGW`（表示对象网关）。根据您的选择，必须在 `policy.cfg` 中定义 `role-mds` 或 `role-rgw`。

14.2.2 RGW 段落

```
RGW {
  ceph_conf = "/etc/ceph/ceph.conf";
  name = "name";
  cluster = "ceph";
}
```

ceph_conf

指向 `ceph.conf` 文件。与 DeepSea 一起部署时，不需要更改此值。

name

NFS Ganesha 使用的 Ceph 客户端用户名。

cluster

Ceph 集群的名称。SUSE Enterprise Storage 5 目前只支持一个集群名称，默认为 `ceph`。

14.2.3 更改默认 NFS Ganesha 端口

NFS Ganesha 默认使用端口 2049 提供 NFS 支持，使用 875 提供 rquota 支持。要更改默认端口号，请在 `NFS_CORE_PARAM` 段落中使用 `NFS_Port` 和 `RQUOTA_Port` 选项，例如：

```
NFS_CORE_PARAM
{
 NFS_Port = 2060;
 RQUOTA_Port = 876;
}
```

14.3 自定义 NFS Ganesha 角色

可为集群节点定义自定义 NFS Ganesha 角色。然后可在 `policy.cfg` 中将这些角色指定给节点。角色允许：

- 分别使用不同的 NFS Ganesha 节点来访问对象网关和 CephFS。
- 将不同的对象网关用户指定给 NFS Ganesha 节点。

拥有不同的对象网关用户可让 NFS Ganesha 节点访问不同的 S3 桶。S3 桶可用于进行访问控制。注意：不要将 S3 桶与 CRUSH 地图中使用的 Ceph 桶混淆。

14.3.1 NFS Ganesha 的不同对象网关用户

下面针对 Salt Master 的示例过程展示如何创建两个具有不同对象网关用户的 NFS Ganesha 角色。在此示例中，使用了角色 `gold` 和 `silver`，DeepSea 已经提供了它们的示例配置文件。

1. 使用您选择的编辑器打开 `/srv/pillar/ceph/stack/global.yml` 文件。如果该文件不存在，请予以创建。
2. 该文件需要包含以下几行：

```
rgw_configurations:
  - rgw
  - silver
  - gold
ganesha_configurations:
  - silver
  - gold
```

稍后可以在 policy.cfg 中指定这些角色。

3. 创建 /srv/salt/ceph/rgw/users/users.d/gold.yml 文件并添加以下内容：

```
- { uid: "gold1", name: "gold1", email: "gold1@demo.nil" }
```

创建 /srv/salt/ceph/rgw/users/users.d/silver.yml 文件并添加以下内容：

```
- { uid: "silver1", name: "silver1", email: "silver1@demo.nil" }
```

4. 现在，需要为每个角色创建 ganesha.conf 的模板。使用 DeepSea 的原始模板是较佳的做法。创建两个副本：

```
root # cd /srv/salt/ceph/ganesha/files/
root # cp ganesha.conf.j2 silver.conf.j2
root # cp ganesha.conf.j2 gold.conf.j2
```

5. 新的角色需要密钥环来访问集群。要提供访问权限，请复制 ganesha.j2：

```
root # cp ganesha.j2 silver.j2
root # cp ganesha.j2 gold.j2
```

6. 复制对象网关的密钥环：

```
root # cd /srv/salt/ceph/rgw/files/
root # cp rgw.j2 silver.j2
root # cp rgw.j2 gold.j2
```

7. 对象网关还需要不同角色的配置：

```
root # cd /srv/salt/ceph/configuration/files/
root # cp ceph.conf.rgw silver.conf
root # cp ceph.conf.rgw gold.conf
```

8. 在 /srv/pillar/ceph/proposals/policy.cfg 中将新建的角色指定给集群节点：

```
role-silver/cluster/NODE1.sls
role-gold/cluster/NODE2.sls
```

将 `NODE1` 和 `NODE2` 分别替换为要将角色指定给的节点的名称。

9. 执行 DeepSea 阶段 0 到 4。

14.3.2 分隔 CephFS 和对象网关 FSAL

下面针对 Salt Master 的示例过程展示如何创建使用 CephFS 和对象网关的 2 个不同的新角色：

1. 使用您选择的编辑器打开文件 `/srv/pillar/ceph/rgw.sls`。如果该文件不存在，请予以创建。

2. 该文件需要包含以下几行：

```
rgw_configurations:
  ganesha_cfs:
    users:
      - { uid: "demo", name: "Demo", email: "demo@demo.nil" }
  ganesha_rgw:
    users:
      - { uid: "demo", name: "Demo", email: "demo@demo.nil" }

ganesha_configurations:
  - ganesha_cfs
  - ganesha_rgw
```

   稍后可以在 `policy.cfg` 中指定这些角色。

3. 现在，需要为每个角色创建 `ganesha.conf` 的模板。使用 DeepSea 的原始模板是较佳的做法。创建两个副本：

```
root # cd /srv/salt/ceph/ganesha/files/
root # cp ganesha.conf.j2 ganesha_rgw.conf.j2
root # cp ganesha.conf.j2 ganesha_cfs.conf.j2
```

4. 编辑 `ganesha_rgw.conf.j2`，删除以下段落：

```
{% if salt.saltutil.runner('select.minions', cluster='ceph', roles='mds') !
= [] %}
```

```
        [...]
{% endif %}
```

5. 编辑 `ganesha_cfs.conf.j2`，删除以下段落：

```
{% if salt.saltutil.runner('select.minions', cluster='ceph', roles=role) !=
 [] %}
        [...]
{% endif %}
```

6. 新的角色需要密钥环来访问集群。要提供访问权限，请复制 `ganesha.j2`：

```
root # cp ganesha.j2 ganesha_rgw.j2
root # cp ganesha.j2 ganesha_cfs.j2
```

   可从 `ganesha_rgw.j2` 中删除 `caps mds = "allow *"` 这一行。

7. 复制对象网关的密钥环：

```
root # cp /srv/salt/ceph/rgw/files/rgw.j2 \
/srv/salt/ceph/rgw/files/ganesha_rgw.j2
```

8. 对象网关需要您对新角色进行配置：

```
root # cp /srv/salt/ceph/configuration/files/ceph.conf.rgw \
/srv/salt/ceph/configuration/files/ceph.conf.ganesha_rgw
```

9. 在 `/srv/pillar/ceph/proposals/policy.cfg` 中将新建的角色指定给集群节点：

```
role-ganesha_rgw/cluster/NODE1.sls
role-ganesha_cfs/cluster/NODE1.sls
```

   将 `NODE1` 和 `NODE2` 分别替换为要将角色指定给的节点的名称。

10. 执行 DeepSea 阶段 0 到 4。

14.4 启动或重启动 NFS Ganesha

要启用并启动 NFS Ganesha 服务，请运行以下命令：

```
root # systemctl enable nfs-ganesha
root # systemctl start nfs-ganesha
```

要重启动 NFS Ganesha，请运行以下命令：

```
root # systemctl restart nfs-ganesha
```

启动或重启动 NFS Ganesha 时，NFS v4 会有 90 秒的超时宽限期。在宽限期内，会主动拒绝来自客户端的新请求。因此，当 NFS 处于宽限状态时，客户端可能会发生请求处理速度变慢的情况。

14.5 设置日志级别

通过编辑文件 `/etc/sysconfig/nfs-ganesha`，可更改默认调试级别 `NIV_EVENT`。将 `NIV_EVENT` 替换为 `NIV_DEBUG` 或 `NIV_FULL_DEBUG`。提高日志详细程度可能会在日志文件中产生大量数据。

```
OPTIONS="-L /var/log/ganesha/ganesha.log -f /etc/ganesha/ganesha.conf -N
 NIV_EVENT"
```

更改日志级别时，需要重启动服务。

14.6 校验导出的 NFS 共享

使用 NFS v3 时，可以在 NFS Ganesha 服务器节点上校验是否导出了 NFS 共享：

```
root # showmount -e
/ (everything)
```

14.7 挂载导出的 NFS 共享

要在客户端主机上挂载导出的 NFS 共享（根据第 14.2 节 “配置”中的配置），请运行以下命令：

```
root # mount -t nfs -o rw,noatime,sync \
```

```
nfs_ganesha_server_hostname:/ /path/to/local/mountpoint
```

14.8 其他资源

https://github.com/nfs-ganesha/nfs-ganesha/wiki/Docs 中提供了原始 NFS Ganesha 文档。

IV 使用 GUI 工具管理集群

15 openATTIC

提示：Calamari 已删除

Calamari 曾是用于管理和监视 Ceph 集群的首选 Web UI 应用。但自 SUSE Enterprise Storage 5 开始，删除了 Calamari，以更先进的 openATTIC 代替。

openATTIC 是一个中央存储管理系统，支持 Ceph 存储集群。借助 openATTIC，您可以从中央管理界面控制所有事宜。您不再需要熟悉 Ceph 存储工具的内部工作方式。集群管理任务可以使用 openATTIC 直观的 Web 界面或通过其 REST API 执行。

15.1 openATTIC 部署和配置

本节介绍部署和配置 openATTIC 及其支持的功能的步骤，以便您可以通过易于使用的 Web 界面来管理 Ceph 集群。

15.1.1 启用使用 SSL 安全访问 openATTIC 的功能

系统默认使用不安全的 HTTP 协议来访问 openATTIC Web 应用程式。要启用安全访问 openATTIC 的功能，您需要手动配置 Apache Web 服务器：

1. 如果您没有经知名证书颁发机构 (CA) 签名的 SSL 证书，请创建一个自我签名的 SSL 证书，并将其文件复制到 Web 服务器预期会检索该证书的目录下，例如：

   ```
   root # openssl req -newkey rsa:2048 -new -nodes -x509 -days 3650 \
    -keyout key.pem -out cert.pem
   root # cp cert.pem /etc/ssl/certs/servercert.pem
   root # cp key.pem /etc/ssl/certs/serverkey.pem
   ```

   请参见 https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_apache2_ssl.html 了解有关创建 SSL 证书的详细信息。

2. 为 `/etc/sysconfig/apache2` 配置文件中的 `APACHE_SERVER_FLAGS` 选项添加“SSL”。您可以手动执行此操作，也可以运行以下命令来执行：

```
root # a2enmod ssl
root # a2enflag SSL
```

3. 为新 Apache 虚拟主机创建 `/etc/apache2/vhosts.d/vhost-ssl.conf`，在其中包含以下内容：

```
<IfDefine SSL>
<IfDefine !NOSSL>
<VirtualHost *:80>
 ServerName OA_HOST_NAME
 Redirect "/" "https://OA_HOST_NAME/"
</VirtualHost>
<VirtualHost _default_:443>
 ServerName OA_HOST_NAME
 DocumentRoot "/srv/www/htdocs"
 ErrorLog /var/log/apache2/error_log
 TransferLog /var/log/apache2/access_log
 SSLEngine on
 SSLCertificateFile /etc/ssl/certs/servercert.pem
 SSLCertificateKeyFile /etc/ssl/certs/serverkey.pem
 CustomLog /var/log/apache2/ssl_request_log ssl_combined
</VirtualHost>
</IfDefine>
</IfDefine>
```

4. 重启动 Web 服务器，以重新加载新虚拟主机定义以及证书文件：

```
root # systemctl restart apache2.service
```

15.1.2 部署 openATTIC

自 SUSE Enterprise Storage 5 开始，已采用 DeepSea 角色的形式来部署 openATTIC。有关一般过程，请参见第 1 章 “Salt 集群管理”。

15.1.3 openATTIC 初始设置

默认情况下，`oaconfig` 会创建管理用户帐户 `openattic`，其用户名与密码相同。为安全起见，强烈建议立即更改此密码：

```
root@minion > oaconfig changepassword openattic
Changing password for user 'openattic'
Password: <enter password>
Password (again): <re-enter password>
Password changed successfully for user 'openattic'
```

15.1.4 openATTIC 中的 DeepSea 集成

一些 openATTIC 功能（例如 iSCSI 网关和对象网关管理）会使用 DeepSea REST API。默认情况下，会启用并配置该 API。如果您出于调试目的需要覆盖其默认设置，请编辑 `/etc/sysconfig/openattic`，添加或更改以下数行：

```
SALT_API_HOST="salt_api_host"
SALT_API_PORT=8001
SALT_API_USERNAME="example_user"
SALT_API_PASSWORD="password"
```

重要：`oaconfig restart`

在更改 `/etc/sysconfig/openattic` 文件之后，记得运行 `oaconfig restart`。

重要：文件语法

在 Python 和 Bash 中会使用 `/etc/sysconfig/openattic`。因此，该文件需要采用 Bash 可以理解的格式，并且在“等号”前后不能有空格。

15.1.5 对象网关管理

默认情况下，openATTIC 中的对象网关管理功能处于启用状态。如果您需要覆盖 DeepSea 中发现的对象网关 API 默认值，请在 `/etc/sysconfig/openattic` 中包括以下选项及相关值。例如：

```
RGW_API_HOST="rgw_api_host"
RGW_API_PORT=80
RGW_API_SCHEME="http"
RGW_API_ACCESS_KEY="VFEG733GBY0DJCIV6NK0"
RGW_API_SECRET_KEY="lJzPbZYZTv8FzmJS5eiiZPHxlT2LMGOMW8ZAeOAq"
```

注意：对象网关的默认资源

如果您的对象网关管理资源未配置为使用默认值“admin”（即“http://rgw_host:80/admin”中所用的值），则您还需要相应地设置 `RGW_API_ADMIN_RESOURCE` 选项。

要获取对象网关身份凭证，请使用 `radosgw-admin` 命令：

```
root@minion > radosgw-admin user info --uid=admin
```

15.1.6 iSCSI 网关管理

默认情况下，openATTIC 中的 iSCSI 网关管理功能处于启用状态。如果您需要覆盖默认 Salt API 主机名，请按第 15.1.4 节 “openATTIC 中的 DeepSea 集成”中所述更改 `SALT_API_HOST` 的值。

15.2 openATTIC Web 用户界面

您可以使用 Web 用户界面来管理 openATTIC。打开 Web 浏览器并导航到 http://`SERVER_HOST`/openattic。要登录，请使用默认用户名 openattic 及相应的密码。

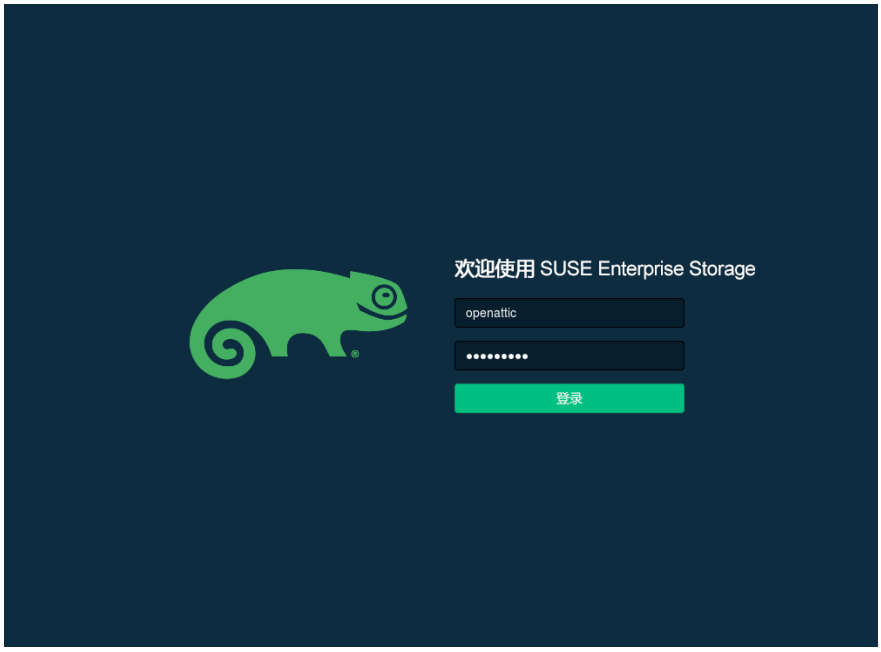


图 15.1： OPENATTIC 登录屏幕

openATTIC 用户界面分为顶部菜单窗格和内容窗格两部分。

顶部窗格的右侧包含当前用户设置的链接、Logout（注销）链接，及现有 Background tasks（后台任务）和系统 Notifications（通知）列表的链接。顶部窗格的其余部分包含 openATTIC 主菜单。

内容窗格因所激活的项目菜单而异。默认会显示一个仪表盘，其中包含许多控件，用于告知您集群的状态。

图 15.2： OPENATTIC 仪表盘

15.3 仪表盘

各仪表盘控件显示与正在运行的 Ceph 集群相关的特定状态信息。点击某个控件的标题之后，该控件会扩展至整个内容窗格，可能会显示更多详细信息。下面列出了几个控件：

Status（状态）控件指出集群是否在正常工作。如果检测到问题，可点击控件内的副标题来查看详细的错误讯息。

Monitors in Quorum（仲裁中的监视器）、Pools（存储池）、OSDs In（加入的 OSD）、OSDs Out（移出的 OSD）、OSDs Up（启动的 OSD）、OSDs Down（停机的 OSD）和Average PGs per OSD（每个 OSD 的平均 PG 数）控件仅显示相关数字。

图 15.3： 基本控件

下列控件显示总存储容量和可用存储容量：Cluster Capacity（集群容量）、Available Capacity（可用容量）、Used Capacity（已用容量）和 Capacity（容量）。

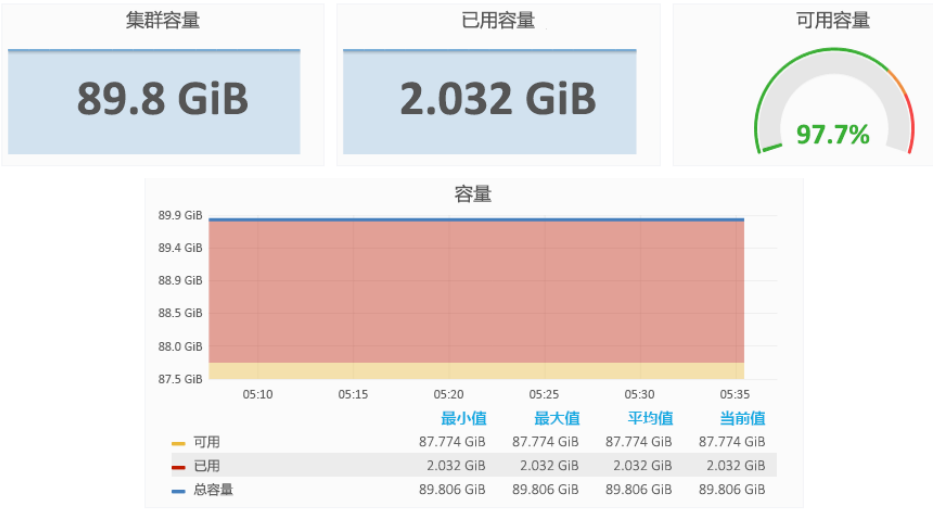


图 15.4： 容量控件

下列控件显示 OSD 和监视器节点延迟：Average OSD Apply Latency（平均 OSD 应用延迟）、Average OSD Commit Latency（平均 OSD 提交延迟）和 Average Monitor Latency（平均监视器延迟）：

图 15.5： 延迟控件

Throughput（吞吐量）控件即时显示每秒的读取和写入统计数字。

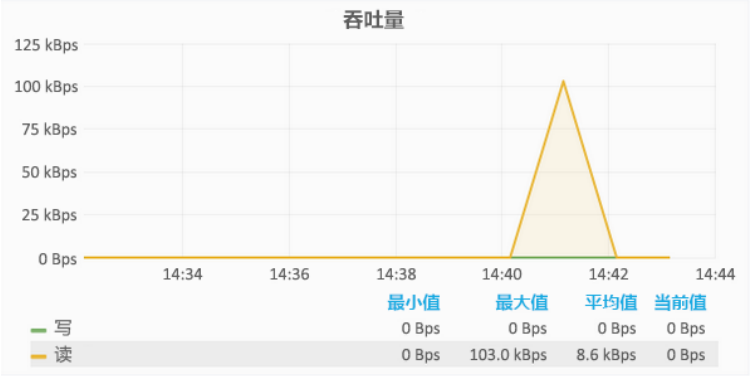


图 15.6： 吞吐量

提示：鼠标悬停时显示更多详细信息

如果您将鼠标指针移到显示的任何图表上方，会在弹出窗口中看到与所指日期和时间相关的更多详细信息。

如果在图表区域中点击，然后沿时间轴向左或向右拖动鼠标指针，时间轴上的间隔将放大到您通过移动鼠标标记的间隔。要缩回原来的比例，请双击图表。

15.4 Ceph 相关任务

openATTIC的 主菜单列出 Ceph 相关任务。目前有以下相关任务：OSD、RBD、存储池、节点、iSCSI、NFS、CRUSH 地图和对象网关。

15.4.1 常用 Web UI 功能

在 openATTIC 中，经常会使用列表 — 例如，存储池列表、OSD 节点列表或 RBD 设备列表。下列常用控件可帮助您管理或调整这些列表：

点击 可刷新项目列表。

点击 可显示或隐藏表格的相应列。

点击 10 并选择要在一页上显示多少行。

在 中点击并键入要搜索的字符串以过滤行。

如果列表跨多页显示，可使用 « ‹ 1 页，共 1 页 › » 来更改当前显示的页。

15.4.2 列出 OSD 节点

要列出所有可用 OSD 节点，请点击主菜单中的 OSD。

列表会显示每个 OSD 的名称、主机名、状态、权重和存储后端。

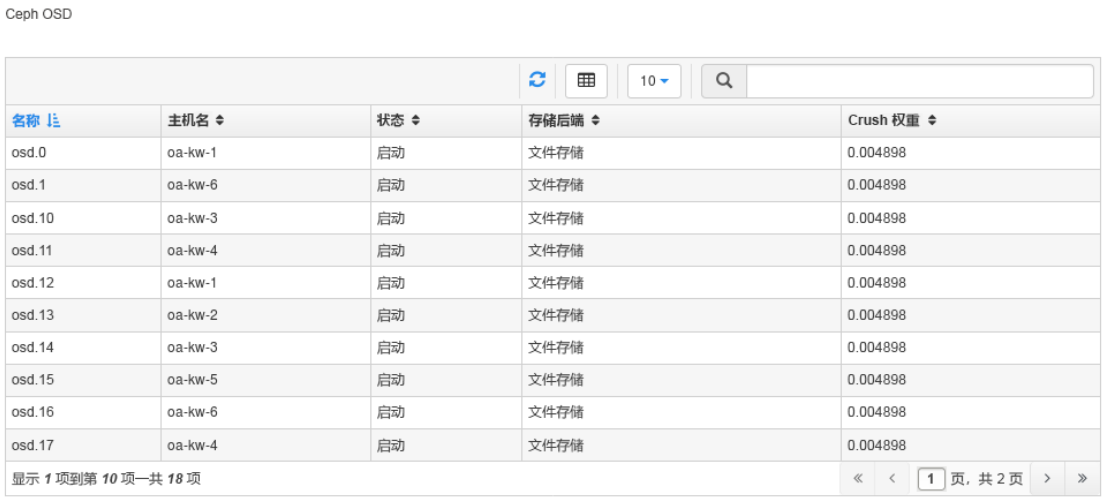


图 15.7： OSD 节点列表

15.4.3 管理 RADOS 块设备 (RBD)

要列出所有可用的 RADOS 块设备，请点击主菜单中的 RBD。

列表会显示每个设备的名称、相关存储池名称、设备大小和已占用百分比（如果在 RADOS 块设备创建期间启用了“fast-diff”）。

图 15.8： RBD 列表

15.4.3.1 状态信息

要查看有关某台设备的更详细信息，请选中其最左侧列中的复选框：

图 15.9： RBD 详细信息

15.4.3.2 统计数字

点击 RADOS 块设备的 Statistics（统计数字）选项卡可查看已传送数据的统计数字。您可以放大和缩小时间范围，只需用鼠标高亮显示该时间范围，或先点击选项卡左上角的日期，然后再选择相应时间范围。

15.4.3.3 RADOS 块设备快照

要创建 RADOS 块设备快照，请点击该块设备的 Snapshots（快照）选项卡，然后从左上方的下拉框中选择 Create（创建）。

选择快照后，您可以对其执行重命名、保护、克隆或删除操作。您可以一次选择多个快照将它们删除。Rollback（回滚）可根据当前快照恢复设备的状态。

图 15.10： RBD 快照

15.4.3.4 删除 RBD

要删除某个设备或一组设备，请选中其最左侧列中的复选框，然后点击 RBD 表格左上方的 Delete（删除）：

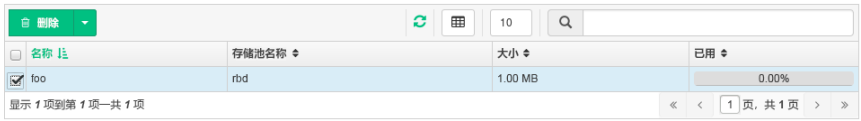


图 15.11： 删除 RBD

15.4.3.5 添加 RBD

要添加新设备，请点击 RBD 表格左上方的 Add（增加），然后在 Create RBD（创建 RBD）屏幕上执行以下操作：

图 15.12： 添加新的 RBD

1. 输入新设备的名称。有关命名限制，请参见《部署指南》，第 2 章 “硬件要求和建议”，第 2.8 节 “命名限制”。
2. 选择将会存储新存储池的集群。
3. 选择将在其中创建新 RBD 设备的存储池。
4. 指定新设备的大小。如果您点击上面的 use max（最大可用）链接，则会填充最大存储池大小。
5. 要细调设备参数，请点击 Expert settings（专家设置），然后激活或停用显示的选项。
6. 点击 Create（创建）以确认。

15.4.4 管理存储池

提示：存储池的更多信息

有关 Ceph 存储池的更多一般信息，请参见第 7 章 “管理存储池”。有关纠删码池特定的信息，请参见第 9 章 “纠删码池”。

要列出所有可用的存储池，请点击主菜单中的 Pools（存储池）。

列表会显示每个存储池的名称、ID、已用空间百分比、归置组数量、副本容量、类型（“副本”或“纠删”）、纠删码配置和 CRUSH 规则组。

图 15.13： 存储池列表

要查看有关某个存储池的更详细信息，请选中其最左侧列中的复选框：

图 15.14： 存储池详细信息

15.4.4.1 删除存储池

要删除某个或一组存储池，请选中其最左侧列中的复选框，然后点击存储池表格左上方的 Delete（删除）：

图 15.15： 删除存储池

15.4.4.2 添加存储池

要添加新存储池，请点击存储池表格左上方的 Add（增加），然后在 Create Ceph pool（创建 Ceph 存储池）屏幕上执行以下操作：

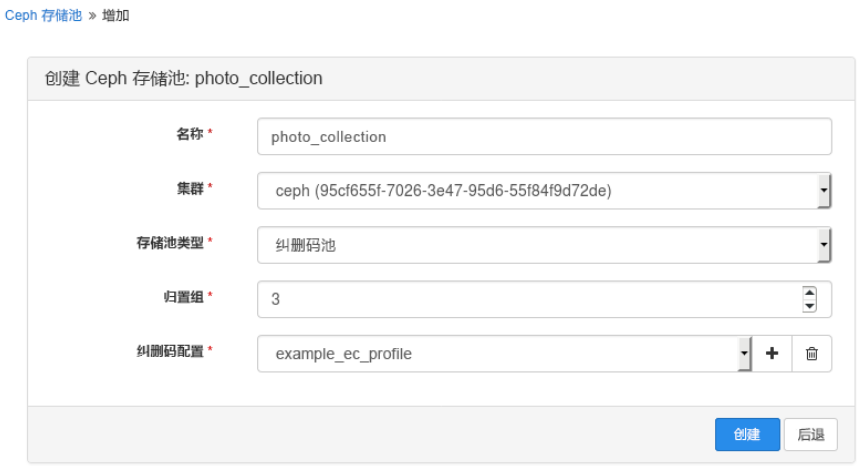


图 15.16： 添加新存储池

1. 输入新存储池的名称。有关命名限制，请参见《部署指南》，第 2 章 “硬件要求和建议”，第 2.8 节 “命名限制”。
2. 选择将会存储新存储池的集群。
3. 选择存储池类型。存储池可以是副本池或是纠删码池。
4. a. 对于副本池，请指定副本容量和归置组数。

   b. 对于纠删码池，请指定归置组数和纠删码配置。您可以点击加号“+”，并指定配置名称、数据块和编码块以及规则组故障域，来添加自定义配置。
5. 点击 Create（创建）以确认。

15.4.5 列出节点

点击主菜单中的 Nodes（节点），可查看集群上可用的节点列表。

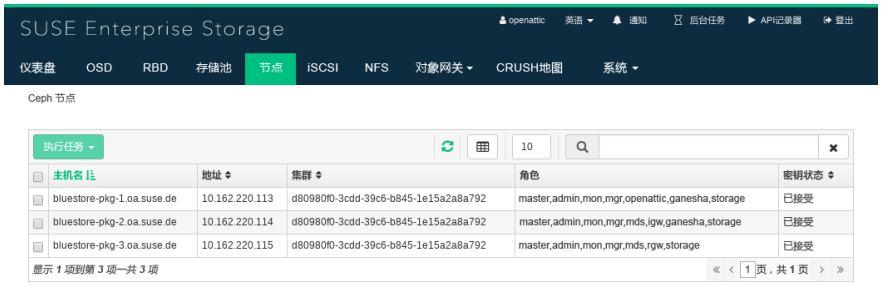


图 15.17： 节点列表

每个节点用其主机名、公共 IP 地址、它所属集群的 ID、节点角色（例如“admin”、“storage”或“master”）和密钥接受状态表示。

15.4.6 管理 NFS Ganesha

提示：NFS Ganesha 的更多信息

有关 NFS Ganesha 的更多一般信息，请参见第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”。

要列出所有可用的 NFS 导出项，请点击主菜单中的 NFS。

列表会显示每个导出项的目录、主机名、状态、存储后端类型以及访问类型。

图 15.18： NFS 导出项列表

要查看有关某个 NFS 导出项的更详细信息，请选中其最左侧列中的复选框：

图 15.19： NFS 导出项详细信息

提示：NFS 挂载命令

在导出项详细视图的底部，有一个挂载命令，可用来轻松挂载来自客户端计算机的相关 NFS 导出项。

15.4.6.1 添加 NFS 导出项

要添加新的 NFS 导出项，请点击导出项表格左上方的 Add（增加）并输入所需的信息。

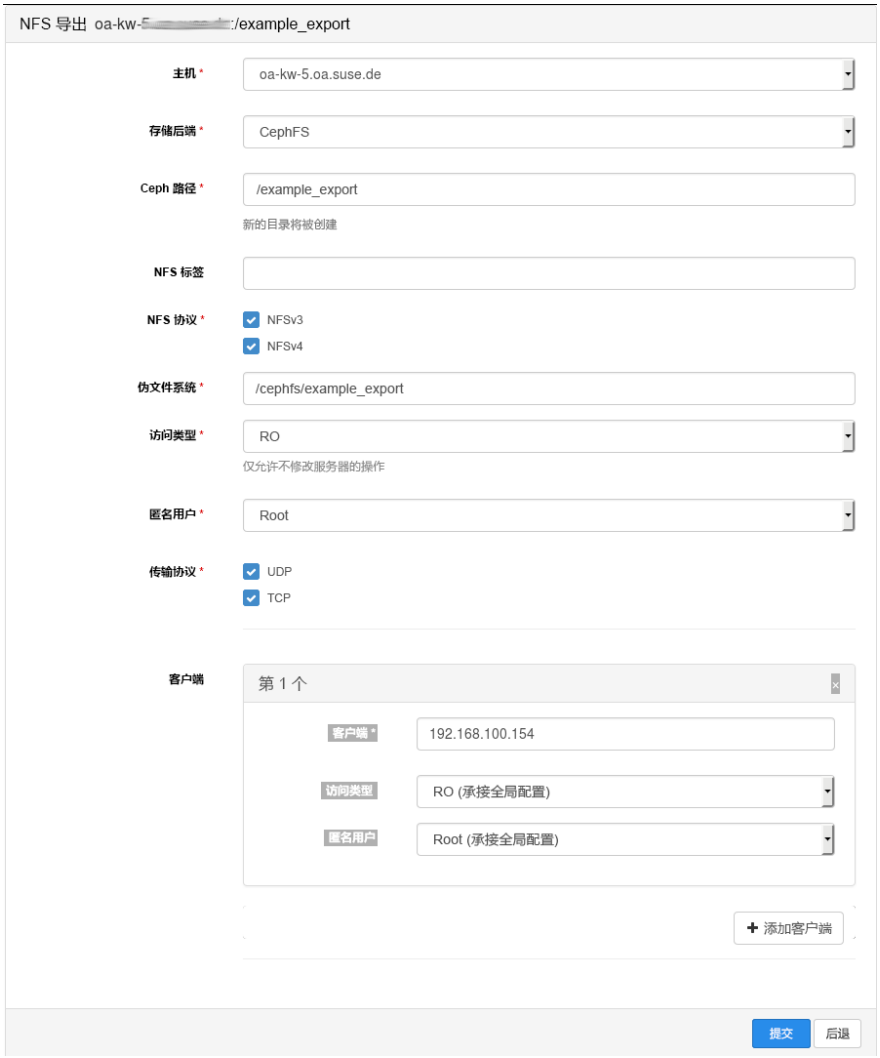


图 15.20： 添加新的 NFS 导出项

1. 选择 NFS 导出项的服务器主机。
2. 选择存储后端 — CephFS 或对象网关。
3. 输入 NFS 导出项的目录路径。如果该目录在服务器上不存在，系统将创建该目录。
4. 指定 NFS 相关的其他选项，例如支持的 NFS 协议版本、访问类型、匿名访问或传输协议。
5. 如果您需要设置限制，仅允许特定的客户端访问，请点击 Add clients（添加客户端）并添加它们的 IP 地址以及访问类型和匿名访问选项。
6. 点击 Submit（提交）以确认。

15.4.6.2 克隆和删除 NFS 导出项

要删除某个或一组导出项，请选中其最左侧列中的复选框，然后点击导出表格左上方的 Delete（删除）。

同样，您可以选择 Clone（克隆）以克隆选中的网关。

15.4.6.3 编辑 NFS 导出项

要编辑现有导出项，可点击导出项表格中该导出项的名称，也可以选中相应复选框，然后点击网关表格左上方的 Edit（编辑）。

然后，您便可以调整 NFS 导出项的所有详细信息。

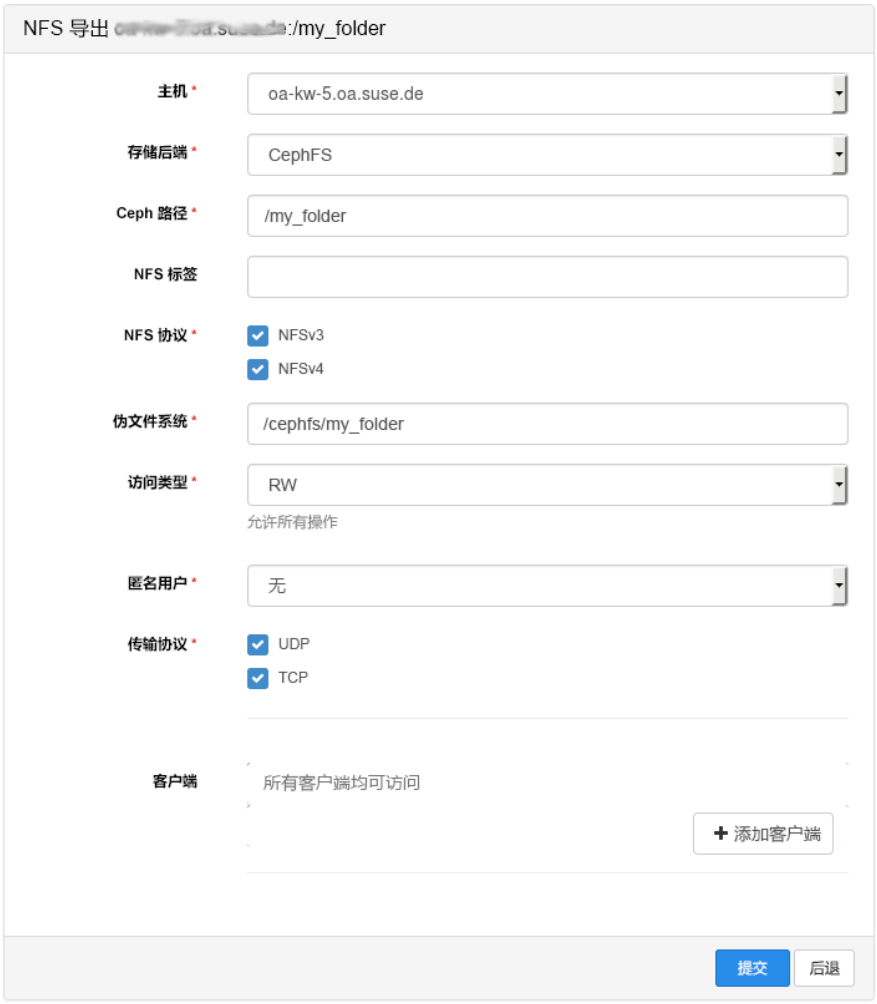


图 15.21： 编辑 NFS 导出项

15.4.7 管理 iSCSI 网关

提示：iSCSI 网关的更多信息

有关 iSCSI 网关的更多一般信息，请参见《部署指南》，第 10 章 “安装 iSCSI 网关”和第 12 章 “Ceph iSCSI 网关”。

要列出所有可用网关，请点击主菜单中的 iSCSI。

列表会显示每个网关的目标、状态、相关门户和 RBD 映像。

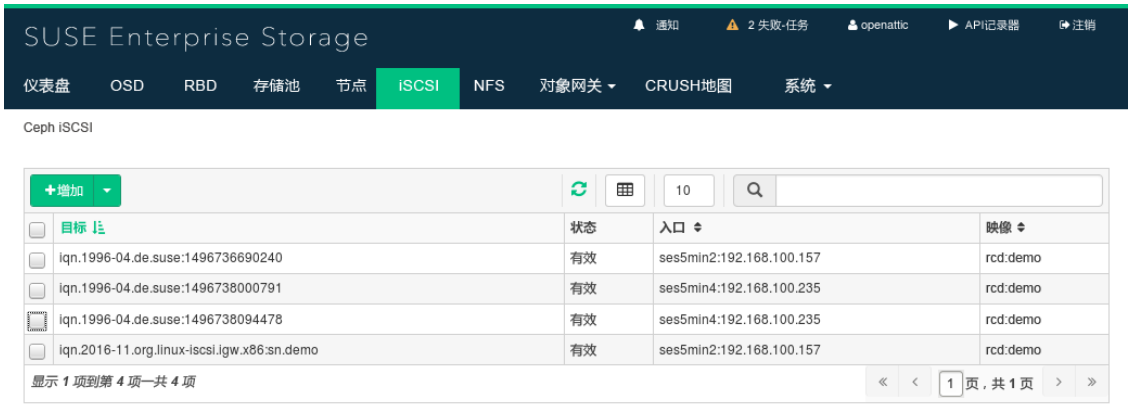


图 15.22： ISCSI 网关列表

要查看有关某个网关的更详细信息，请选中其最左侧列中的复选框：

图 15.23： 网关详细信息

15.4.7.1 添加 iSCSI 网关

要添加新的 iSCSI 网关，请点击网关表格左上方的 Add（增加）并输入所需的信息。

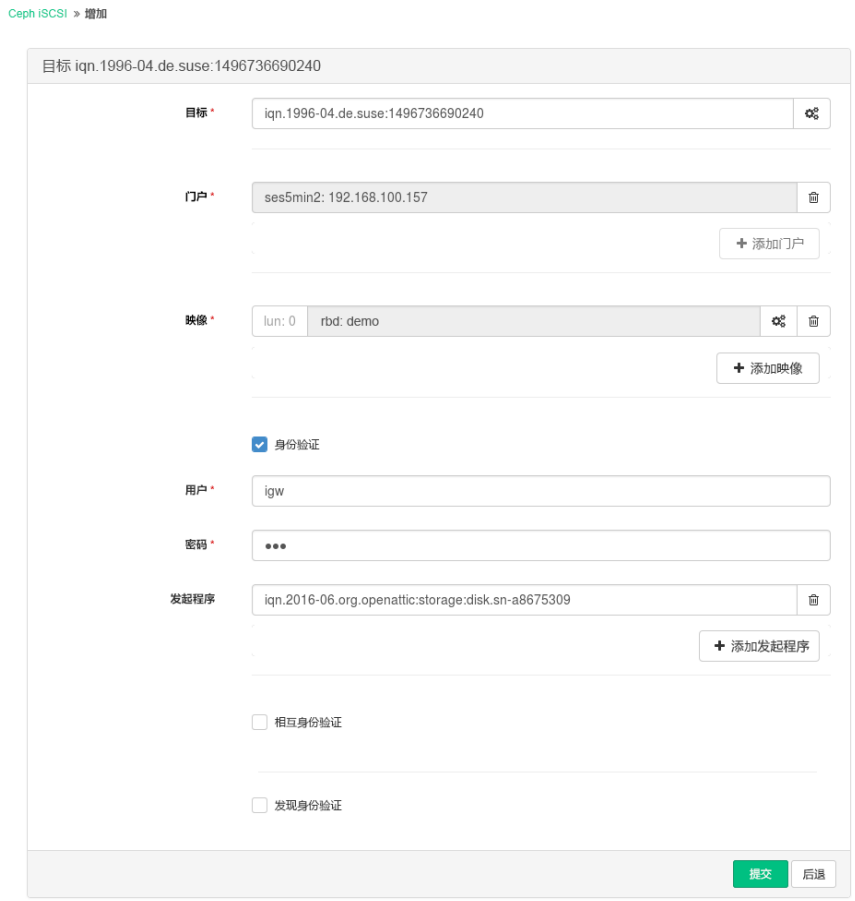


图 15.24： 添加新的 ISCSI 网关

1. 输入新网关的目标地址。
2. 点击 Add portal（添加门户）并从列表中选择一个或多个 iSCSI 门户。
3. 点击 Add image（添加映像）并为网关选择一个或多个 RBD 映像。
4. 如果您需要使用身份验证才能访问网关，请选中 Authentication（身份验证）复选框并输入身份凭证。选中 Mutual authentication（相互身份验证）和 Discovery authentication（发现身份验证）之后，您可看到更多高级身份验证选项。
5. 点击 Submit（提交）以确认。

15.4.7.2 编辑 iSCSI 网关

要编辑某个现有 iSCSI 网关，可点击网关表格中该网关的名称，也可以选中相应复选框，然后点击网关表格左上方的 Edit（编辑）。

然后，您便可以修改 iSCSI 目标、添加或删除门户，以及添加或删除相关 RBD 映像。您还可以调整网关的身份验证信息。

15.4.7.3 克隆和删除 iSCSI 网关

要删除某个或一组网关，请选中其最左侧列中的复选框，然后点击网关表格左上方的 Delete（删除）。

同样，您可以选择 Clone（克隆）以克隆选中的网关。

15.4.7.4 启动和停止 iSCSI 网关

要启动所有网关，请选择网关表格左上方的 Start all（全部启动）。要停止所有网关，请选择 Stop all（全部停止）。

15.4.8 查看集群 CRUSH 地图

点击主菜单中的 CRUSH Map（CRUSH 地图）可查看集群 CRUSH 地图。

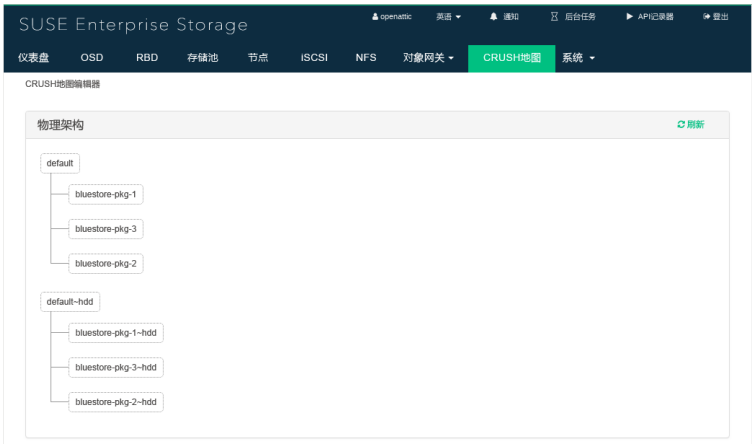


图 15.25： CRUSH 地图

在 Physical setup（物理架构）窗格中，您可以看到如 CRUSH 地图中所描述的集群结构。

您可以在 Replication rules（复制规则）窗格的 Content（内容）下拉框中选择其中一个规则组，来查看相应的规则组。

图 15.26： 复制规则

15.4.9 管理对象网关用户和桶

提示：对象网关的更多信息

有关对象网关的更多一般信息，请参见第 11 章 “Ceph Object Gateway”。

要列出对象网关用户，请选择主菜单中的Object Gateway（对象网关）› Users（用户）。

列表会显示每个用户的 ID、显示名称、电子邮件地址、用户是否处于暂停状态以及用户的最大桶数。

图 15.27： 对象网关用户列表

15.4.9.1 添加新的对象网关用户

要添加新的对象网关用户，请点击用户表格左上方的 Add（增加）并输入相关信息。

提示：更多信息

第 11.5.2 节 “管理 S3 和 Swift 帐户”中提供了有关对象网关用户帐户的详细信息。

图 15.28： 添加新的对象网关用户

1. 输入用户名、全名和（可选）电子邮件地址以及用户的最大桶数。
2. 如果用户最初应处于暂停状态，请选中 Suspended（暂停）复选框。
3. 指定 S3 身份验证的访问钥和机密密钥。如果您希望 openATTIC 为您生成密钥，请选中 Generate key（生成钥）。
4. 在 User quota（用户配额）部分，设置当前用户的配额限制。

选中 Enabled（启用）以激活用户配额限制。您可以指定集群中用户可使用磁盘空间的 Maximum size（最大容量），也可以选中 Unlimited size（容量不限）不设置大小限制。同样，可以指定用户可在集群存储上存储的 Maximum objects（最大对象数目），也可以选中 Unlimited objects（对象数目不限），允许用户存储任意数量的对象。

图 15.29： 用户配额

5. 在 Bucket Quota（桶配额）部分，设置当前用户的桶配额限制。

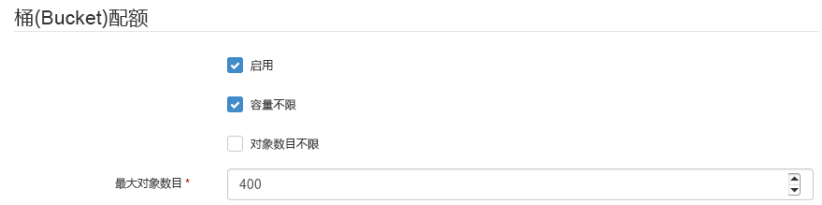


图 15.30： 桶配额

6. 点击 Submit（提交）以确认。

15.4.9.2 删除对象网关用户

要删除一个或多个对象网关用户，请选中其最左侧列中的复选框，然后选择用户表格左上方的 Delete（删除）。

15.4.9.3 编辑对象网关用户

要编辑对象网关用户的用户信息，请选中其最左侧列中的复选框，然后选择用户表格左上方的 Edit（编辑），或者点击用户的 ID。您可以更改在第 15.4.9.1 节 “添加新的对象网关用户”中添加用户时输入的信息，以及下列其他信息：

子用户

添加、删除或编辑当前所编辑用户的子用户。

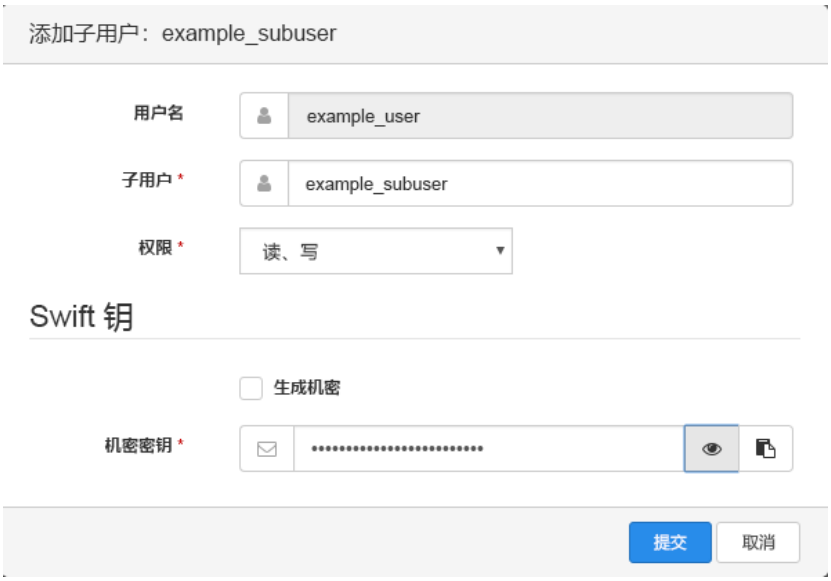


图 15.31： 添加子用户

密钥

添加、删除或查看当前所编辑用户的访问钥和机密密钥。

您可为当前编辑的用户添加 S3 钥，或查看其子用户的 Swift 钥。

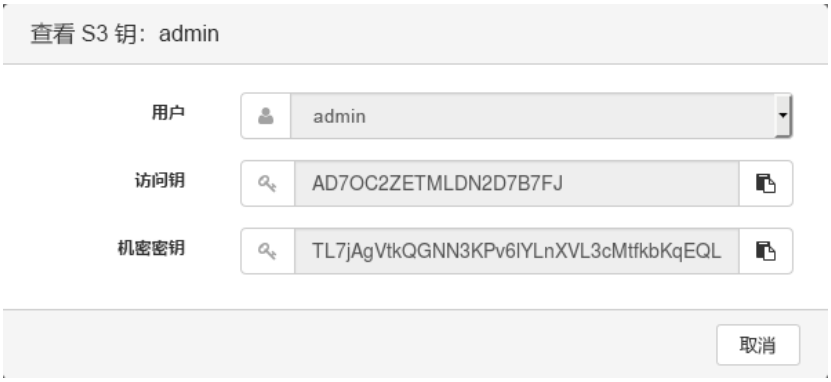


图 15.32： 查看 S3 钥

使能

添加或删除用户的使能。使能适用于 buckets、zone、users、metadata 和 usage。每个使能值可以是“read”、“write”或“*”（表示读取和写入特权）其中之一。

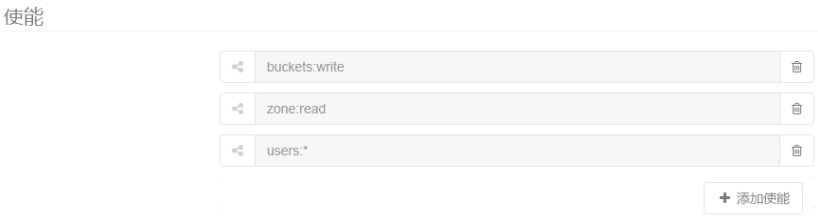


图 15.33： 使能

15.4.9.4 列出对象网关用户的桶

提示

桶是一种用于存储数据对象的机制。一个用户帐户可具有许多个桶，但是桶名称必须唯一。“桶”这个术语通常在 Amazon S3 API 中使用，而在 OpenStack Swift API 环境中使用的是“容器”。

点击 Object Gateway（对象网关）› Buckets（桶）可列出所有可用的对象网关桶。

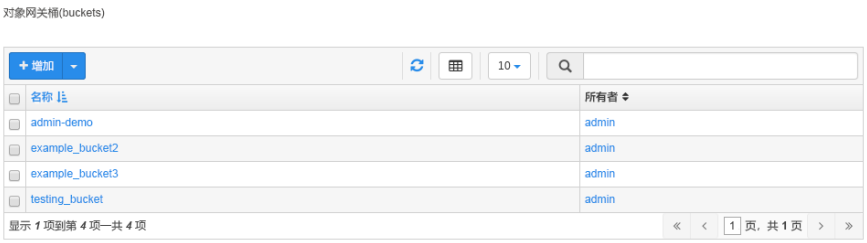


图 15.34： 对象网关桶

15.4.9.5 为对象网关用户添加桶

要添加新的桶，请点击桶表格左上方的 Add（增加），然后输入新的桶名称及相关的对象网关用户。点击 Submit（提交）以确认。

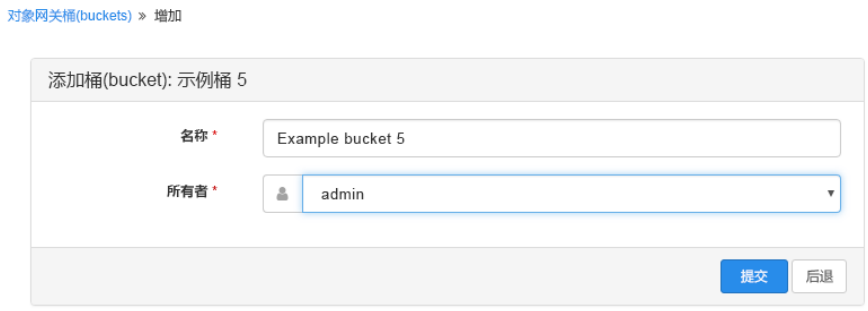


图 15.35： 添加新的桶

15.4.9.6 查看桶详细信息

要查看有关某个对象网关桶的详细信息，请选中桶表格最左侧列中相应的复选框。

图 15.36： 桶详细信息

15.4.9.7 编辑桶

要编辑某个桶，请选中最左侧列中该桶的复选框，然后选择桶表格左上方的 Edit（编辑），或者点击它的名称。

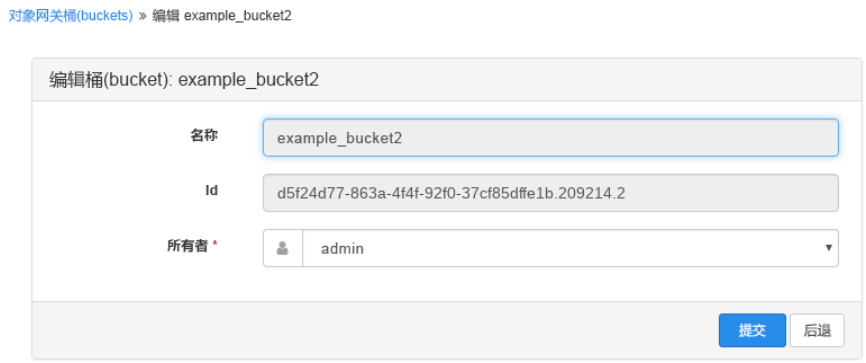


图 15.37： 编辑对象网关桶

在编辑屏幕上，您可以更改桶所属的用户。

15.4.9.8 删除桶

要删除一个或多个对象网关桶，请激活桶表格最左侧列中相应的复选框，然后选择桶表格左上方的 Delete（删除）。

图 15.38： 删除桶

要确认删除，请在 Delete buckets（删除桶）弹出窗口中键入“yes”，然后点击 删除。

警告：请谨慎删除

目前，当您删除对象网关桶时，系统不会校验桶实际上是否正被使用，例如由 NFS Ganesha 通过 S3 存储后端使用。

V 与虚拟化工具集成

16 将 libvirt 与 Ceph 搭配使用

libvirt 库在超级管理程序接口与使用这些接口的软件应用之间建立了一个虚拟机抽象层。使用 libvirt，开发人员和系统管理员可将工作重心放在通用管理框架、通用 API、通用外壳接口 (virsh) 以及诸多不同的超级管理程序（包括 QEMU/KVM、Xen、LXC 或 VirtualBox）上。

Ceph 块设备支持 QEMU/KVM。您可以通过与 libvirt 连接的软件来使用 Ceph 块设备。云解决方案使用 libvirt 来与 QEMU/KVM 交互，而 QEMU/KVM 通过 librbd 来与 Ceph 块设备交互。

要创建使用 Ceph 块设备的 VM，请按以下各节中所述的过程操作。在示例中，我们分别使用了 libvirt-pool、client.libvirt 和 new-libvirt-image 作为存储池名称、用户名和映像名称。您可以根据个人喜好使用任何值，但在执行后续过程中的命令时，请务必替换这些值。

16.1 配置 Ceph

要将 Ceph 配置为与 libvirt 搭配使用，请执行以下步骤：

1. 创建存储池。下面的示例使用存储池名称 libvirt-pool 和 128 个归置组。

   ```
   ceph osd pool create libvirt-pool 128 128
   ```

   校验该存储池是否存在。

   ```
   ceph osd lspools
   ```

2. 创建 Ceph 用户。下面的示例使用 Ceph 用户名 client.libvirt 并引用 libvirt-pool。

   ```
   ceph auth get-or-create client.libvirt mon 'allow r' osd \
    'allow class-read object_prefix rbd_children, allow rwx pool=libvirt-pool'
   ```

   校验该名称是否存在。

   ```
   ceph auth list
   ```

注意

`libvirt` 将使用 ID `libvirt`，而不是 Ceph 名称 `client.libvirt` 来访问 Ceph。有关 ID 与名称之间的差别的详细说明，请参见 http://docs.ceph.com/docs/master/rados/operations/user-management/#user 。

3. 使用 QEMU 在 RBD 池中创建映像。下面的示例使用映像名称 `new-libvirt-image` 并引用 `libvirt-pool`。

提示：密钥环文件位置

确保在 /etc/ceph/ceph.conf 中指定“libvirt”用户密钥环路径，例如：

```
keyring = /etc/ceph/client.libvirt.keyring
```

如果该密钥环不存在，请使用以下命令创建它：

```
root # ceph auth get client.libvirt > /etc/ceph/
client.libvirt.keyring
```

```
qemu-img create -f raw rbd:libvirt-pool/new-libvirt-image:id=libvirt 2G
```

校验该映像是否存在。

```
rbd -p libvirt-pool ls
```

16.2 准备 VM 管理器

虽然您可以单独使用 `libvirt`，而不借助 VM 管理器，但您可能会发现，使用 `virt-manager` 创建第一个域会更简单。

1. 安装虚拟机管理器。

```
sudo zypper in virt-manager
```

2. 准备/下载要运行虚拟化的系统的 OS 映像。
3. 起动虚拟机管理器。

```
virt-manager
```

16.3 创建 VM

要使用 `virt-manager` 创建 VM，请执行以下步骤：

1. 从列表中选择连接，右键点击该连接，然后选择新建。
2. 通过提供现有存储的路径来导入现有的磁盘映像。指定 OS 类型和内存设置，并给虚拟机命名，例如 `libvirt-virtual-machine`。
3. 完成配置并启动 VM。
4. 使用 `sudo virsh list` 校验新建的域是否存在。如果需要，请指定连接字符串，例如

```
virsh -c qemu+ssh://root@vm_host_hostname/system list
Id    Name                           State
----------------------------------------------------
[...]
 9     libvirt-virtual-machine       running
```

5. 在将 VM 配置为与 Ceph 搭配使用前，登录 VM 并将其停止。

16.4 配置 VM

本章重点介绍如何使用 `virsh` 配置 VM，以与 Ceph 集成。`virsh` 命令通常需要 root 特权 (`sudo`)，它不会返回相应的结果，也不会告知您需要 root 特权。有关 `virsh` 命令的参考，请参见 Virsh 命令参考 (http://www.libvirt.org/virshcmdref.html)。

1. 使用 `virsh edit vm-domain-name` 打开配置文件。

```
sudo virsh edit libvirt-virtual-machine
```

2. <devices> 下面应有一个 <disk> 项。

```
<devices>
    <emulator>/usr/bin/qemu-system-x86_64</emulator>
    <disk type='file' device='disk'>
      <driver name='qemu' type='raw'/>
      <source file='/path/to/image/recent-linux.img'/>
      <target dev='vda' bus='virtio'/>
      <address type='drive' controller='0' bus='0' unit='0'/>
    </disk>
```

将 `/path/to/image/recent-linux.img` 替换为 OS 映像的路径。

重要

请使用 `sudo virsh edit`，不要使用文本编辑器。如果使用文本编辑器编辑 `/etc/libvirt/qemu` 下的配置文件，`libvirt` 可能无法识别更改。如果 `/etc/libvirt/qemu` 下的 XML 文件内容与 `sudo virsh dumpxml vm-domain-name` 返回的结果有差异，则表示 VM 可能没有正常工作。

3. 将之前创建的 Ceph RBD 映像添加为 <disk> 项。

```
<disk type='network' device='disk'>
        <source protocol='rbd' name='libvirt-pool/new-libvirt-image'>
                <host name='monitor-host' port='6789'/>
        </source>
        <target dev='vda' bus='virtio'/>
</disk>
```

将 `monitor-host` 替换为主机的名称，并根据需要替换存储池名称和/或映像名称。可为 Ceph monitor 添加多个 <host> 项。`dev` 属性是逻辑设备名称，将显示在 VM 的 `/dev` 目录下。可选的 bus 属性表示要模拟的磁盘设备类型。有效的设置都特定于驱动程序（例如 ide、scsi、virtio、xen、usb 或 sata）。有关 <disk> 元素及其子元素和属性的详细信息，请参见磁盘 (http://www.libvirt.org/formatdomain.html#elementsDisks)。

4. 保存文件。

5. 如果 Ceph 集群已启用身份验证（默认会启用），则您必须生成机密。打开所选的编辑器，并创建包含以下内容的 `secret.xml` 文件：

```
<secret ephemeral='no' private='no'>
        <usage type='ceph'>
                <name>client.libvirt secret</name>
        </usage>
</secret>
```

6. 定义机密。

```
sudo virsh secret-define --file secret.xml
<uuid of secret is output here>
```

7. 获取 `client.libvirt` 密钥，并将密钥字符串保存到某个文件中。

```
ceph auth get-key client.libvirt | sudo tee client.libvirt.key
```

8. 设置机密的 UUID。

```
sudo virsh secret-set-value --secret uuid of secret \
--base64 $(cat client.libvirt.key) && rm client.libvirt.key secret.xml
```

此外，必须通过将以下 `<auth>` 项添加到前面输入的 `<disk>` 元素（请将 uuid 值替换为上述命令行示例的结果），来手动设置机密。

```
sudo virsh edit libvirt-virtual-machine
```

然后，在域配置文件中添加 `<auth></auth>` 元素：

```
...
</source>
<auth username='libvirt'>
        <secret type='ceph' uuid='9ec59067-fdbc-a6c0-03ff-df165c0587b8'/>
</auth>
<target ...
```

注意

示例 ID 为 `libvirt`，而不是在第 16.1 节 “配置 Ceph”的步骤 2 中生成的 Ceph 名称 `client.libvirt`。请务必使用所生成的 Ceph 名称的 ID 组成部分。如果出于某种原因需要重新生成机密，则在再次执行 `sudo virsh secret-set-value` 之前，需要执行 `sudo virsh secret-undefine uuid`。

16.5 总结

配置要与 Ceph 搭配使用的 VM 之后，便可启动该 VM。要校验 VM 与 Ceph 是否可相互通讯，可执行以下过程。

1. 检查 Ceph 是否在运行：

```
ceph health
```

2. 检查 VM 是否在运行：

```
sudo virsh list
```

3. 检查 VM 是否在与 Ceph 通讯。将 `vm-domain-name` 替换为 VM 域的名称：

```
sudo virsh qemu-monitor-command --hmp vm-domain-name 'info block'
```

4. 检查 `/dev` 或 `/proc/partitions` 下是否存在 `&target dev='hdb' bus='ide'/>` 中的设备：

```
ls /dev
cat /proc/partitions
```

17 Ceph 用作 QEMU KVM 实例的后端

最常见的 Ceph 用例涉及到向虚拟机提供块设备映像。例如，在理想的配置中，用户可以创建包含 OS 和所有相关软件的“黄金”映像。然后，用户可以创建该映像的快照。最后，用户可以克隆该快照（通常要克隆多次，有关详细信息，请参见第 8.3 节 “块设备快照”）。能够创建快照的写入时复制克隆，就意味着 Ceph 能够快速向虚拟机供应块设备映像，因为客户端不需要在每次运转新的虚拟机时都下载整个映像。

Ceph 块设备可与 QEMU 虚拟机集成。有关 QEMU KVM 的详细信息，请参见 https://www.suse.com/documentation/sles-12/book_virt/data/part_virt_qemu.html 。

17.1 安装

要使用 Ceph 块设备，需在 QEMU 上安装相应的驱动程序。请检查是否已安装 `qemu-block-rbd` 包，并根据需要予以安装：

```
sudo zypper install qemu-block-rbd
```

17.2 用法

使用 QEMU 命令行时，您需要指定存储池名称和映像名称。您也可以指定快照名称。

```
qemu-img command options \
rbd:pool-name/image-name@snapshot-name:option1=value1:option2=value2...
```

例如，可按如下所示指定 `id` 和 `conf` 选项：

```
qemu-img command options \
rbd:pool_name/image_name:id=glance:conf=/etc/ceph/ceph.conf
```

17.3 使用 QEMU 创建映像

可以通过 QEMU 创建块设备映像。必须指定 `rbd`、存储池名称，以及要创建的映像的名称。此外，必须指定映像的大小。

```
qemu-img create -f raw rbd:pool-name/image-name size
```

例如：

```
qemu-img create -f raw rbd:pool1/image1 10G
Formatting 'rbd:pool1/image1', fmt=raw size=10737418240 nocow=off cluster_size=0
```

重要

事实上，`raw` 数据格式是可对 RBD 使用的唯一合理格式选项。从技术上讲，您也可以使用 QEMU 支持的其他格式（例如 `qcow2`），但这会增加额外的开销，如果启用了快速缓存，还会在实时迁移虚拟机时让卷变得不安全。

17.4 使用 QEMU 调整映像大小

可以通过 QEMU 调整块设备映像的大小。必须指定 `rbd`、存储池名称，以及要调整大小的映像的名称。此外，必须指定映像的大小。

```
qemu-img resize rbd:pool-name/image-name size
```

例如：

```
qemu-img resize rbd:pool1/image1 9G
Image resized.
```

17.5 使用 QEMU 检索映像信息

可以通过 QEMU 检索块设备映像的信息。必须指定 `rbd`、存储池名称和映像名称。

```
qemu-img info rbd:pool-name/image-name
```

例如：

```
qemu-img info rbd:pool1/image1
```

```
image: rbd:pool1/image1
file format: raw
virtual size: 9.0G (9663676416 bytes)
disk size: unavailable
cluster_size: 4194304
```

17.6 使用 RBD 运行 QEMU

QEMU 可以通过 `librbd` 直接将映像作为虚拟块设备来访问。这可以避免额外的环境切换，并可利用 RBD 快速缓存的优势。

您可以使用 `qemu-img` 将现有的虚拟机映像转换成 Ceph 块设备映像。例如，如果您有一个 qcow2 映像，则可以运行：

```
qemu-img convert -f qcow2 -O raw sles12.qcow2 rbd:pool1/sles12
```

要运行从该映像引导的虚拟机，可以运行：

```
qemu -m 1024 -drive format=raw,file=rbd:pool1/sles12
```

RBD 快速缓存 (http://ceph.com/docs/master/rbd/rbd-config-ref/#cache-settings) 可大幅提高性能。QEMU 的快速缓存选项可控制 `librbd` 快速缓存：

```
qemu -m 1024 -drive format=rbd,file=rbd:pool1/sles12,cache=writeback
```

17.7 启用丢弃功能/TRIM

Ceph 块设备支持丢弃操作。这意味着，guest 可以发送 TRIM 请求，让 Ceph 块设备回收未使用的空间。可以通过结合 discard 选项挂载 `XFS`，在 guest 中启用此功能。

要让 guest 可使用此功能，必须为块设备显式启用此功能。为此，您必须指定与驱动器关联的 `discard_granularity`：

```
qemu -m 1024 -drive format=raw,file=rbd:pool1/sles12,id=drive1,if=none \
-device driver=ide-hd,drive=drive1,discard_granularity=512
```

注意

上面的示例使用 IDE 驱动程序。Virtio 驱动程序不支持丢弃功能。

如果使用 `libvirt`，请使用 `virsh edit` 编辑 libvirt 域的配置文件，以包含 `xmlns:qemu` 值。然后，将 `qemu:commandline block` 添加为该域的子级。下面的示例说明如何将包含 `qemu id=` 的两个设备设置为不同的 `discard_granularity` 值。

```
<domain type='kvm' xmlns:qemu='http://libvirt.org/schemas/domain/qemu/1.0'>
 <qemu:commandline>
  <qemu:arg value='-set'/>
  <qemu:arg value='block.scsi0-0-0.discard_granularity=4096'/>
  <qemu:arg value='-set'/>
  <qemu:arg value='block.scsi0-0-1.discard_granularity=65536'/>
 </qemu:commandline>
</domain>
```

17.8 QEMU 快速缓存选项

QEMU 的快速缓存选项与以下 Ceph RBD 快速缓存设置对应。

写回：

```
rbd_cache = true
```

直写：

```
rbd_cache = true
rbd_cache_max_dirty = 0
```

无：

```
rbd_cache = false
```

QEMU 的快速缓存设置会覆盖 Ceph 的默认设置（未在 Ceph 配置文件中显式指定的设置）。如果在 Ceph 配置文件中显式指定了 RBD 快速缓存 (http://ceph.com/docs/master/rbd/rbd-config-ref/#cache-settings) 设置，则您的 Ceph 设置会覆盖 QEMU 快速缓存设置。如果在 QEMU 命令行中指定了快速缓存设置，则 QEMU 命令行设置会覆盖 Ceph 配置文件设置。

VI FAQ、提示和故障诊断

18 技巧与提示

本章提供可帮助您增强 Ceph 集群性能的信息，以及有关如何设置集群的提示。

18.1 调整整理 (Scrub)

默认情况下，Ceph 每天会执行浅层整理 (light scrub)（有关详细信息，请参见第 6.5 节 “整理 (Scrub)”），每周会执行深层整理 (deep scrub)。浅层整理 (light scrub) 会检查对象大小及校验和，以确保归置组存储的是相同的对象数据。深层整理 (deep scrub) 会检查对象的内容及其副本，以确保实际内容相同。在整理 (scrub) 期间检查数据完整性会增加集群上的 I/O 负载。

默认设置允许 Ceph OSD 在不合适的时间（如负载较重时）启动整理 (scrub)。当整理 (scrub) 操作与客户操作发生冲突时，可能会出现延迟和性能不佳情况。Ceph 提供了数个整理 (scrub) 设置，可将整理 (scrub) 限制在低负载或非高峰时段执行。

如果集群在日间负载高而在夜间负载低，请考虑将整理 (scrub) 限制在夜间执行，例如在晚上 11 点到早上 6 点期间执行。

```
[osd]
osd_scrub_begin_hour = 23
osd_scrub_end_hour = 6
```

如果使用时间限制无法有效决定整理 (scrub) 时间表，请考虑使用 `osd_scrub_load_threshold` 选项。其默认值为 0.5，但也可针对低负载情况进行相应调整：

```
[osd]
osd_scrub_load_threshold = 0.25
```

18.2 在不重新平衡的情况下停止 OSD

进行定期维护时，您可能需要停止 OSD。如果您不希望 CRUSH 自动重新平衡集群，以免出现大量数据传输，请先将集群设为 `noout`：

```
root@minion > ceph osd set noout
```

当集群设为 noout 时，您便可开始在需要执行维护工作的故障域中停止 OSD：

```
root@minion > systemctl stop ceph-osd@OSD_NUMBER.service
```

有关详细信息，请参见第 3.1.2 节 “启动、停止和重启动个别服务”。

完成维护工作后，再次启动 OSD：

```
root@minion > systemctl start ceph-osd@OSD_NUMBER.service
```

OSD 服务启动后，取消集群的 noout 设置：

```
root@minion > ceph osd unset noout
```

18.3 节点时间同步

Ceph 要求特定节点之间的时间保持精确的同步。您应该使用自己的 NTP 服务器设置节点。尽管您可以将所有 ntpd 实例指向远程公共时间服务器，但不建议对 Ceph 采用这种做法。如果采用这种配置，集群中的每个节点都会借助自己的 NTP 守护进程通过因特网来持续与三到四台时间服务器通讯，而这些服务器全部都相距很远。此解决方案在很大程度上带来了延迟方面的变数，使得难以甚至无法将时钟偏差保持在 0.05 秒以下（Ceph monitor 要求这种精度）。

因此，应该使用一台计算机作为整个集群的 NTP 服务器。这样，NTP 服务器 ntpd 实例可以指向远程（公共）NTP 服务器，或者可以有自己的时间源。然后，所有节点上的 ntpd 实例将指向这台本地服务器。此类解决方案有多种优势，例如，避免不必要的网络流量和时钟偏差，减轻公共 NTP 服务器上的负载。有关如何设置 NTP 服务器的详细信息，请参见《SUSE Linux Enterprise Server 管理指南》 (https://www.suse.com/documentation/sled11/book_sle_admin/data/cha_netz_xntp.html)。

要更改集群上的时间，请执行以下操作：

重要：设置时间

您可能会遇到需要将时间往回调的情况，例如，当时间从夏令时改成标准时间时就需要如此。不建议将时间回调的幅度超过集群的关闭时长。将时间往前调不会造成任何问题。

过程 18.1： 集群上的时间同步

1. 停止正在访问 Ceph 集群的所有客户端，尤其是使用 iSCSI 的客户端。

2. 关闭 Ceph 集群。在每个节点上，运行：

```
systemctl stop ceph.target
```

注意

如果您使用了 Ceph 和 SUSE OpenStack Cloud，则还需停止 SUSE OpenStack Cloud。

3. 校验 NTP 服务器的设置是否正确，即所有 ntpd 守护进程是否可从本地网络中的一个或多个源获取时间。
4. 在 NTP 服务器上设置正确的时间。
5. 确认 NTP 正在运行且在正常工作，然后在所有节点上运行：

```
status ntpd.service
```

或者

```
ntpq -p
```

6. 启动所有监视节点，并校验是否不存在时钟偏差：

```
systemctl start target
```

7. 启动所有 OSD 节点。
8. 启动其他 Ceph 服务。
9. 启动 SUSE OpenStack Cloud（如果有）。

18.4 检查不均衡的数据写入

如果数据均衡写入 OSD，则认为集群是平衡的。集群中的每个 OSD 都分配了权重。权重是一个相对数字，告知 Ceph 应写入相关 OSD 的数据量。权重越高，要写入的数据就越多。如果 OSD 的权重为零，则不会向其写入任何数据。如果某个 OSD 的权重相对于其他 OSD 而言较高，则大部分数据将会写入这个 OSD，致使集群变得不平衡。

不平衡集群的性能较差；如果某个权重较高的 OSD 突然崩溃，则大量的数据就需要转移到其他 OSD，这也会导致集群速度变慢。

为避免此问题，应该定期检查 OSD 中的数据写入量。如果写入量介于给定规则组所指定 OSD 组容量的 30% 到 50% 之间，则您需要重新设置 OSD 的权重。检查各个磁盘，找出其中哪些磁盘的填满速度比其他磁盘更快（或者一般情况下速度更慢），并降低其权重。对于数据写入量不足的 OSD 可以采用相同的思路：可以提高其权重，让 Ceph 将更多的数据写入其中。在下面的示例中，您将确定 ID 为 13 的 OSD 的权重，并将权重从 3 重新设置为 3.05：

```
$ ceph osd tree | grep osd.13
 13  3                   osd.13  up  1

 $ ceph osd crush reweight osd.13 3.05
 reweighted item id 13 name 'osd.13' to 3.05 in crush map

 $ ceph osd tree | grep osd.13
 13  3.05                osd.13  up  1
```

提示：按使用率重新设置 OSD 的权重

`ceph osd reweight-by-utilization` *阈值* 命令可自动完成降低严重过度使用的 OSD 的权重的过程。默认情况下，此命令将对达到平均使用率的 120% 的 OSD 降低权重，但是，如果您指定了阈值，则命令会改用该百分比。

18.5 /var/lib/ceph 的 Btrfs 子卷

SUSE Linux Enterprise 默认安装在 Btrfs 分区上。应该从 Btrfs 快照和回滚操作中排除目录 `/var/lib/ceph`，当 MON 在节点上运行时更应如此。DeepSea 提供了 `fs` 运行程序，可为此路径设置子卷。

18.5.1 全新安装的要求

如果您是首次安装集群，则必须满足以下要求才能使用 DeepSea 运行程序：

- Salt 和 DeepSea 已根据本文档正确安装且正常运行。
- 已调用 `salt-run state.orch ceph.stage.0` 将所有 Salt 和 DeepSea 模块同步到 Minion。
- Ceph 尚未安装，因此 ceph.stage.3 尚未运行，且 `/var/lib/ceph` 尚不存在。

18.5.2 现有安装的要求

如果已安装集群，则必须满足以下要求才能使用 DeepSea 运行程序：

- 已将节点升级到 SUSE Enterprise Storage，并且集群受 DeepSea 的控制。
- Ceph 集群已启动且正常运行。
- 升级过程已将 Salt 和 DeepSea 模块同步到所有 Minion 节点。

18.5.3 自动安装

- 在 Salt Master 上运行：

```
root@master # salt-run state.orch ceph.migrate.subvolume
```

对于不存在 `/var/lib/ceph` 目录的节点，此命令一次将在一个节点上执行以下操作：

- 创建 `/var/lib/ceph`，作为 `@/var/lib/ceph` Btrfs 子卷。
- 挂载新子卷并相应地更新 `/etc/fstab`。
- 对 `/var/lib/ceph` 禁用写入时复制。

对于已安装 Ceph 的节点，此命令一次将在一个节点上执行以下操作：

- 终止正在运行的 Ceph 进程。
- 卸载节点上的 OSD。
- 创建 `@/var/lib/ceph` Btrfs 子卷，并迁移现有的 `/var/lib/ceph` 数据。

- 挂载新子卷并相应地更新 `/etc/fstab`。
- 对 `/var/lib/ceph/*` 禁用写入时复制，并忽略 `/var/lib/ceph/osd/*`。
- 重新挂载 OSD。
- 重启动 Ceph 守护进程。

18.5.4 手动安装

此过程使用新的 `fs` 运行程序。

1. 在所有节点上检查 `/var/lib/ceph` 的状态，并列显有关如何继续操作的建议：

   ```
   root@master # salt-run fs.inspect_var
   ```

   此操作会返回以下命令之一：

   ```
   salt-run fs.create_var
   salt-run fs.migrate_var
   salt-run fs.correct_var_attrs
   ```

2. 运行上一步中返回的命令。
   如果某个节点上出错，针对其他节点的执行进程也会停止，并且运行程序会尝试还原已执行的更改。请查阅出现问题的 Minion 上的日志文件，以确定问题所在。解决问题后，可以重新运行运行程序。

命令 `salt-run fs.help` 提供 `fs` 模块的所有运行程序和模块命令列表。

18.6 增加文件描述符

对于 OSD 守护进程而言，读/写操作对保持 Ceph 集群平衡至关重要。这些守护进程通常需要同时打开许多文件进行读取和写入。在 OS 级别，同时打开的文件的最大数目称为“文件描述符的最大数目”。

为防止 OSD 用尽文件描述符，您可以覆盖 OS 默认值，并在 `/etc/ceph/ceph.conf` 中指定该数字，例如：

```
max_open_files = 131072
```

更改 `max_open_files` 之后，需在相关的 Ceph 节点上重启动 OSD 服务。

18.7 如何对包含 OSD 日记的 OSD 使用现有分区

重要

本节介绍一个仅供存储专家和开发人员研究的高级主题。所述的方法主要用于解决使用非标准 OSD 日记大小的情况。如果 OSD 分区小于 10GB，则其初始权重将舍入为 0，因而不会在其上放置任何数据，所以您应该提高其权重。我们不会处理日记过满的情况。

如果您要使用现有的磁盘分区作为 OSD 节点，则需要将 OSD 日记和数据分区列入 GPT 分区表中。

需要将正确的分区类型设置为 OSD 分区，使 `udev` 能够正确识别这些分区；并将分区的所有权设置为 `ceph:ceph`。

例如，要设置日记分区 `/dev/vdb1` 和数据分区 `/dev/vdb2` 的分区类型，请运行以下命令：

```
sudo sgdisk --typecode=1:45b0969e-9b03-4f30-b4c6-b4b80ceff106 /dev/vdb
sudo sgdisk --typecode=2:4fbd7e29-9d25-41b8-afd0-062c0ceff05d /dev/vdb
```

提示

Ceph 分区表类型列于 `/usr/lib/udev/rules.d/95-ceph-osd.rules` 中：

```
cat /usr/lib/udev/rules.d/95-ceph-osd.rules
# OSD_UUID
ACTION=="add", SUBSYSTEM=="block", \
  ENV{DEVTYPE}=="partition", \
  ENV{ID_PART_ENTRY_TYPE}=="4fbd7e29-9d25-41b8-afd0-062c0ceff05d", \
  OWNER:="ceph", GROUP:="ceph", MODE:="660", \
  RUN+="/usr/sbin/ceph-disk --log-stdout -v trigger /dev/$name"
ACTION=="change", SUBSYSTEM=="block", \
  ENV{ID_PART_ENTRY_TYPE}=="4fbd7e29-9d25-41b8-afd0-062c0ceff05d", \
```

```
  OWNER="ceph", GROUP="ceph", MODE="660"

# JOURNAL_UUID
ACTION=="add", SUBSYSTEM=="block", \
  ENV{DEVTYPE}=="partition", \
  ENV{ID_PART_ENTRY_TYPE}=="45b0969e-9b03-4f30-b4c6-b4b80ceff106", \
  OWNER:="ceph", GROUP:="ceph", MODE:="660", \
  RUN+="/usr/sbin/ceph-disk --log-stdout -v trigger /dev/$name"
ACTION=="change", SUBSYSTEM=="block", \
  ENV{ID_PART_ENTRY_TYPE}=="45b0969e-9b03-4f30-b4c6-b4b80ceff106", \
  OWNER="ceph", GROUP="ceph", MODE="660"
[...]
```

18.8 与虚拟化软件集成

18.8.1 在 Ceph 集群中存储 KVM 磁盘

您可以为 KVM 驱动的虚拟机创建磁盘映像，将该映像存储在 Ceph 存储池中，选择性地将现有映像的内容转换到该映像，然后使用 `qemu-kvm` 运行虚拟机，以利用集群中存储的磁盘映像。有关详细信息，请参见第 17 章 “Ceph 用作 QEMU KVM 实例的后端”。

18.8.2 在 Ceph 集群中存储 `libvirt` 磁盘

类似于 KVM（请参见第 18.8.1 节 “在 Ceph 集群中存储 KVM 磁盘”），您可以使用 Ceph 来存储 `libvirt` 驱动的虚拟机。这样做的好处是可以运行任何支持 `libvirt` 的虚拟化解决方案，例如 KVM、Xen 或 LXC。有关详细信息，参见第 16 章 “将 libvirt 与 Ceph 搭配使用”。

18.8.3 在 Ceph 集群中存储 Xen 磁盘

使用 Ceph 存储 Xen 磁盘的方法之一是按第 16 章 “将 libvirt 与 Ceph 搭配使用”中所述利用 `libvirt`。

另一种方法是让 Xen 直接与 `rbd` 块设备驱动程序通讯：

1. 如果尚未为 Xen 准备磁盘映像，请新建一个：

```
rbd create myimage --size 8000 --pool mypool
```

2. 列出存储池 `mypool` 中的映像，并检查您的新映像是否在该存储池中：

```
rbd list mypool
```

3. 通过将 `myimage` 映像映射到 `rbd` 内核模块来创建一个新的块设备：

```
sudo rbd map --pool mypool myimage
```

提示：用户名和身份验证

要指定用户名，请使用 `--id 用户名`。此外，如果您使用了 `cephx` 身份验证，则还必须指定机密。该机密可能来自密钥环，或某个包含机密的文件：

```
sudo rbd map --pool rbd myimage --id admin --keyring /path/to/keyring
```

或者

```
sudo rbd map --pool rbd myimage --id admin --keyfile /path/to/file
```

4. 列出所有映射的设备：

```
rbd showmapped
 id pool   image   snap device
 0  mypool myimage -    /dev/rbd0
```

5. 现在，可以将 Xen 配置为使用此设备作为运行虚拟机所用的磁盘。例如，可将下行添加到 `xl` 样式的域配置文件：

```
disk = [ '/dev/rbd0,,sda', '/dev/cdrom,,sdc,cdrom' ]
```

18.9 Ceph 的防火墙设置

警告：使用防火墙时 DeepSea 阶段失败

当防火墙处于活动状态（甚至只是配置了防火墙）时，DeepSea 部署阶段会失败。要正确通过该阶段，需要运行以下命令关闭防火墙

```
root@master # systemctl stop SuSEfirewall2.service
```

或在 `/srv/pillar/ceph/stack/global.yml` 中将 `FAIL_ON_WARNING` 选项设为“False”：

```
FAIL_ON_WARNING: False
```

建议使用 SUSE 防火墙保护网络集群通讯。可以通过选择 YaST › 安全性和用户 › 防火墙 › 允许的服务来编辑防火墙的配置。

下面列出了 Ceph 相关服务以及这些服务通常使用的端口号：

Ceph Monitor
: 启用 Ceph MON 服务或端口 6789 (TCP)。

Ceph OSD 或元数据服务器
: 启用 Ceph OSD/MDS 服务或端口 6800-7300 (TCP)。

iSCSI 网关
: 打开端口 3260 (TCP)。

对象网关
: 打开对象网关通讯所用的端口。此端口在 `/etc/ceph.conf` 内以 `rgw frontends =` 开头的行中设置。HTTP 的默认端口为 80，HTTPS (TCP) 的默认端口为 443。

NFS Ganesha
: 默认情况下，NFS Ganesha 使用端口 2049（NFS 服务、TCP）和 875（rquota 支持、TCP）。有关更改默认 NFS Ganesha 端口的详细信息，请参见第 14.2.3 节 “更改默认 NFS Ganesha 端口”。

基于 Apache 的服务，例如 openATTIC、SMT 或 SUSE Manager

打开用于 HTTP 的端口 80，用于 HTTPS (TCP) 的端口 443。

SSH

打开端口 22 (TCP)。

NTP

打开端口 123 (UDP)。

Salt

打开端口 4505 和 4506 (TCP)。

Grafana

打开端口 3000 (TCP)。

Prometheus

打开端口 9100 (TCP)。

18.10 测试网络性能

为方便测试网络性能，DeepSea `net` 运行程序提供了以下命令。

- 向所有节点发出简单 ping：

```
root@master # salt-run net.ping
Succeeded: 9 addresses from 9 minions average rtt 1.35 ms
```

- 向所有节点发出大规模 ping：

```
root@master # salt-run net.jumbo_ping
Succeeded: 9 addresses from 9 minions average rtt 2.13 ms
```

- 带宽测试：

```
root@master # salt-run net.iperf
Fastest 2 hosts:
    |_
      - 192.168.58.106
```

```
      - 2981 Mbits/sec
    |_
      - 192.168.58.107
      - 2967 Mbits/sec
Slowest 2 hosts:
    |_
      - 192.168.58.102
      - 2857 Mbits/sec
    |_
      - 192.168.58.103
      - 2842 Mbits/sec
```

18.11 更换存储磁盘

如果您需要更换 Ceph 集群中的存储磁盘，可在集群具有完全运作能力时更换。更换操作会导致数据传输量暂时性增加。

如果整个磁盘都有故障，Ceph 至少需要重新写入与故障磁盘容量相同的数据量。如果正常取下磁盘然后重新装上，以免更换过程中出现冗余损失，重新写入的数据量将增大到两倍。如果新磁盘与更换掉的磁盘大小不同，将导致重新分发某些额外数据，甚至超出所有 OSD 的用量。

19 常见问题 (FAQ)

19.1 归置组数量对集群的性能有何影响？

当集群空间的 70% 至 80% 已满时，便有必要在其中添加更多的 OSD。增加 OSD 的数量时，可以考虑同时增加归置组的数量。

警告

更改归置组 (PG) 的数量会导致在集群中传输大量的数据。

为最近调整大小的集群计算最佳值是一项复杂的任务。

如果 PG 数量较大，将会创建一些较小的数据块。在发生 OSD 故障后，这样可以加速恢复过程，但同时会对监视器节点施加大量的负荷，因为这些节点负责计算数据位置。

另一方面，如果 PG 数量较小，则发生 OSD 故障后，恢复系统所需的时间和数据传输量就会增加，但不会对监视器节点施加如此多的负荷，因为需要由这些节点计算位置的数据块更少（但更大）。

有关集群最佳 PG 数量的详细信息，请参见在线计算器 (http://ceph.com/pgcalc/)。

19.2 是否可以在同一集群上使用 SSD 和普通硬盘？

一般而言，固态硬盘 (SSD) 的速度比普通硬盘要快。如果对同一写入操作混用这两种磁盘，SSD 磁盘的数据写入速度将会因普通硬盘的性能限制而减慢。因此，对于遵循相同规则的数据写入操作，切勿混用 SSD 和普通硬盘（有关数据存储规则的详细信息，请参见第 6.3 节 “规则组”）。

以下两种情况通常适合在同一集群上使用 SSD 和普通硬盘：

1. 针对遵循不同规则的数据写入操作使用各自的磁盘类型。然后，您需要针对 SSD 磁盘和普通硬盘分别使用不同的规则。
2. 针对特定目的使用各自的磁盘类型。例如，将 SSD 磁盘用于日记，将普通硬盘用于存储数据。

19.3 在 SSD 上使用日记存在哪些利弊？

为 OSD 日记使用 SSD 有助于提高性能，因为在仅包含普通硬盘的 OSD 中，日记通常会成为瓶颈。SSD 往往用于共享多个 OSD 的日记。

下面列出了为 OSD 日记使用 SSD 的潜在弊端：

- SSD 磁盘比普通硬盘更昂贵。但是，由于一个 OSD 日记最多只需要 6GB 磁盘空间，因此价格因素并不那么重要。
- SSD 磁盘会占用存储插槽，而大容量普通硬盘可以利用这些插槽来扩展集群容量。
- 与普通硬盘相比，SSD 磁盘的写入周期更少，但最新的技术有望解决该问题。
- 如果在同一块 SSD 磁盘上共享更多的日记，则在 SSD 磁盘发生故障后，将会面临丢失所有相关 OSD 的风险。在这种情况下，将需要移动大量数据来重新平衡集群。
- 热插拔磁盘变得更复杂，因为有故障的 OSD 与日记磁盘之间不存在一对一的数据映射关系。

19.4 磁盘出现故障时会发生什么情况？

当某个存储集群数据的磁盘出现硬件问题而无法正常工作时，会发生以下情况：

- 相关的 OSD 将会崩溃，并自动从集群中删除。
- 有故障磁盘的数据会从其他 OSD 中的存储相同数据的其他副本复制到集群中的另一个 OSD。
- 然后，您应该从集群的 CRUSH 地图中删除该磁盘，并从主机硬件中移除其实体。

19.5 日记磁盘出现故障时会发生什么情况？

可将 Ceph 配置为在独立于 OSD 的设备上存储日记或预写式日志。如果专用于日记的磁盘发生故障，相关的 OSD 也会发生故障（请参见第 19.4 节 “磁盘出现故障时会发生什么情况？”）。

警告：在一个磁盘上托管多个日记

要大幅提升性能，可以使用高速磁盘（例如 SSD）来存储多个 OSD 的日记分区。不建议在一个磁盘上托管 4 个以上的 OSD 的日记，因为一旦日记磁盘发生故障，您就会面临所有相关 OSD 磁盘存储的数据都将丢失的风险。

20 查错

本章描述您在操作 Ceph 集群时可能会遇到的多种问题。

20.1 报告软件问题

如果您在运行 SUSE Enterprise Storage 时遇到了与某些组件（例如 Ceph 或对象网关）相关的问题，请将问题报告给 SUSE 技术支持。建议使用 `supportconfig` 实用程序来报告问题。

提示

由于 `supportconfig` 是模块化软件，因此请确保已安装 `supportutils-plugin-ses` 包。

```
rpm -q supportutils-plugin-ses
```

如果 Ceph 服务器上缺少此包，可使用以下命令安装

```
zypper ref && zypper in supportutils-plugin-ses
```

尽管您可以在命令行中使用 `supportconfig`，但我们建议使用相关的 YaST 模块。https://www.suse.com/documentation/sles-12/singlehtml/book_sle_admin/book_sle_admin.html#sec.admsupport.supportconfig 上提供了有关 `supportconfig` 的详细信息。

20.2 使用 `rados` 发送大型对象失败并显示“OSD 已满”

`rados` 是用于管理 RADOS 对象存储的命令行实用程序。有关更多信息，请参见 `man 8 rados`。

如果您使用 `rados` 实用程序将大型对象发送到 Ceph 集群，例如

```
rados -p mypool put myobject /file/to/send
```

该对象可能会填满所有相关的 OSD 空间，并导致集群性能出现严重问题。

20.3 XFS 文件系统损坏

在极少见的情况下（例如出现内核错误，或硬件损坏/配置不当），OSD 用来存储数据的底层文件系统 (XFS) 可能会受损或无法挂载。

如果您确定硬件没有问题并且系统配置正确，请报告 SUSE Linux Enterprise Server 内核的 XFS 子系统出现了错误，并将特定的 OSD 标记为停机：

```
ceph osd down OSD identification
```

警告：不要格式化或修改受损的设备

尽管使用 `xfs_repair` 来修复文件系统问题看似合理，但它会修改文件系统，因此请不要使用该命令。OSD 可以启动，但它的运行可能会受到影响。

现在，请运行以下命令擦除底层磁盘，并重新创建 OSD：

```
ceph-disk prepare --zap $OSD_DISK_DEVICE $OSD_JOURNAL_DEVICE"
```

例如：

```
ceph-disk prepare --zap /dev/sdb /dev/sdd2
```

20.4 “每个 OSD 的 PG 数过多”状态讯息

如果在运行 `ceph status` 之后收到 `每个 OSD 的 PG 数过多` 讯息，则表示超出了 `mon_pg_warn_max_per_osd` 值（默认值为 300）。系统会将此值与每个 OSD 的 PG 数比率进行比较。这意味着集群设置并不是最佳的。

创建存储池后，便不能减少 PG 数。您可以放心地删除尚不包含任何数据的存储池，然后重新创建具有较少 PG 的存储池。如果存储池中已包含数据，则唯一的解决方法是将 OSD 添加到集群，使每个 OSD 的 PG 比率变低。

20.5 “nn pg 停滞在非活动状态”状态讯息

如果在运行 `ceph status` 之后收到 停滞在非活动状态 状态讯息，则表示 Ceph 不知道要将存储的数据复制到何处，因此无法遵循复制规则。此问题可能在完成初始 Ceph 设置后立即发生，并且系统可自动修复。在其他情况下出现此问题可能需要进行手动交互，例如激活已中止的 OSD，或者将新的 OSD 添加到集群。在极少见的情况下，降低复制级别可能有所帮助。

如果位置组一直处于停滞状态，则您需要检查 `ceph osd tree` 的输出。输出采用的应该是树型结构，类似于第 20.7 节 “OSD 停机”中的示例。

如果 `ceph osd tree` 的输出的结构相对扁平，如以下示例中所示

```
ceph osd tree
ID WEIGHT TYPE NAME    UP/DOWN REWEIGHT PRIMARY-AFFINITY
-1      0 root default
 0      0 osd.0             up  1.00000          1.00000
 1      0 osd.1             up  1.00000          1.00000
 2      0 osd.2             up  1.00000          1.00000
```

您应该检查相关的 CRUSH 地图是否包含树型结构。如果 CRUSH 地图也是扁平的，或者不包含上面示例中所示的主机，则可能表示集群中的主机名解析未正常工作。

如果层次结构不正确（例如，根包含主机，但 OSD 位于顶层，并且本身未指定到主机），则您需要将 OSD 移到层次结构中的正确位置。可以使用 `ceph osd crush move` 和/或 `ceph osd crush set` 命令实现此目的。有关更多详细信息，请参见第 6.4 节 “CRUSH 地图操作”。

20.6 OSD 权重为 0

当 OSD 启动时，系统会给它指定一个权重。权重越高，集群向该 OSD 写入数据的几率就越大。该权重将在集群 CRUSH 地图中指定，或者通过 OSD 的启动脚本计算得出。

在某些情况下，计算出的 OSD 权重值可能会向下舍入到零。这表示不会安排该 OSD 存储数据，因此不会向其写入数据。发生此情况的原因通常是相应的磁盘太小（小于 15GB），应该更换为更大的磁盘。

20.7 OSD 停机

OSD 守护进程的状态要么是正在运行，要么是已停止/停机。导致 OSD 停机的原因一般有以下三种：

- 硬盘故障。
- OSD 已崩溃。
- 服务器已崩溃。

可运行以下命令来查看 OSD 的详细状态

```
ceph osd tree
# id  weight  type name up/down reweight
 -1    0.02998  root default
 -2    0.009995   host doc-ceph1
 0     0.009995      osd.0 up  1
 -3    0.009995   host doc-ceph2
 1     0.009995      osd.1 up  1
 -4    0.009995   host doc-ceph3
 2     0.009995      osd.2 down  1
```

示例列表显示 `osd.2` 已停机。然后，可以检查是否已挂载 OSD 所在的磁盘：

```
lsblk -f
 [...]
 vdb
 ├─vdb1               /var/lib/ceph/osd/ceph-2
 └─vdb2
```

可以通过检查 OSD 的日志文件 `/var/log/ceph/ceph-osd.2.log` 来跟踪其停机原因。找到并解决 OSD 未运行的原因之后，请使用以下命令将它启动

```
sudo systemctl start ceph-osd@2.service
```

请记得将 `2` 替换为已停止的 OSD 的实际编号。

20.8 查找运行缓慢的 OSD

优化集群性能时，识别集群中运行缓慢的存储/OSD 非常重要。原因在于，如果将数据写入（最）缓慢的磁盘，则会拖慢整个写操作，因为它始终要等待在所有相关磁盘上的操作全部完成。

找到存储瓶颈并非无足轻重。需要检查每一个 OSD 才能找出使写入过程减慢的 OSD。要针对单个 OSD 执行基准测试，请运行：

```
ceph tell osd.OSD_ID_NUMBER bench
```

例如：

```
root # ceph tell osd.0 bench
 { "bytes_written": 1073741824,
   "blocksize": 4194304,
   "bytes_per_sec": "19377779.000000"}
```

然后，需要在每个 OSD 上运行此命令，并与 `bytes_per_sec` 值相比较，以找出（最）缓慢的 OSD。

20.9 解决时钟偏差警告

所有集群节点中的时间信息都必须同步。如果某个节点的时间未完全同步，在检查集群状态时，您可能会收到时钟偏差警告。

可使用 NTP 来管理时间同步（请参见 http://en.wikipedia.org/wiki/Network_Time_Protocol ）。设置每个节点，使其时间与一台或多台 NTP 服务器同步，最好是与同组的 NTP 服务器同步。如果节点上仍然出现时间偏差，请执行以下步骤予以修复：

```
systemctl stop ntpd.service
systemctl stop ceph-mon.target
systemctl start ntpd.service
systemctl start ceph-mon.target
```

然后，可以查询 NTP 同级，并使用 `sudo ntpq -p` 检查时间偏移。

Ceph monitor 的时钟需要同步，彼此之间的偏差必须控制在 0.05 秒以内。有关详细信息，请参考第 18.3 节 “节点时间同步”。

20.10 网络问题导致集群性能不佳

导致集群性能变差的原因有很多，其中之一可能是网络问题。在这种情况下，您可能会发现集群即将达到仲裁数、OSD 和监视器节点脱机、数据传输耗费很长时间，或者尝试了很多次重新连接。

要检查集群性能下降是否由网络问题导致，请检查 `/var/log/ceph` 目录中的 Ceph 日志文件。

要解决集群上的网络问题，请重点关注以下几点：

- 基本网络诊断。尝试使用 DeepSea 诊断工具运行程序 `net.ping` 在集群节点之间执行 ping 命令，确定单个接口是否可以连接到特定的接口，并了解平均响应时间。此命令还会报告比平均值要慢得多的任何特定响应时间。例如：

  ```
  root@master # salt-run net.ping
    Succeeded: 8 addresses from 7 minions average rtt 0.15 ms
  ```

  尝试在启用极大帧的情况下验证所有接口：

  ```
  root@master # salt-run net.jumbo_ping
    Succeeded: 8 addresses from 7 minions average rtt 0.26 ms
  ```

- 网络性能基准测试。尝试使用 DeepSea 的网络性能运行程序 `net.iperf` 来测试节点间的网络带宽。在某个给定的集群节点上，有许多 `iperf` 进程（具体视 CPU 核心数而定）作为服务器启动。其余的集群节点将作为客户端来生成网络流量。该运行程序会报告单个节点上所有 `iperf` 进程的累积带宽。此值应该能反映所有集群节点上可达到的最大网络吞吐量。例如：

  ```
  root@master # salt-run net.iperf cluster=ceph output=full
  192.168.128.1:
      8644.0 Mbits/sec
  192.168.128.2:
      10360.0 Mbits/sec
  ```

```
192.168.128.3:
    9336.0 Mbits/sec
192.168.128.4:
    9588.56 Mbits/sec
192.168.128.5:
    10187.0 Mbits/sec
192.168.128.6:
    10465.0 Mbits/sec
```

- 检查集群节点上的防火墙设置。确保这些设置不会阻止 Ceph 运转所需的端口/协议。有关防火墙设置的详细信息，请参见第 18.9 节 “Ceph 的防火墙设置”。
- 检查网卡、电缆或交换机等网络硬件是否正常运行。

提示：独立网络

为确保在集群节点之间进行快速安全的网络通讯，请设置一个专供集群 OSD 和监视器节点使用的独立网络。

20.11 /var 空间不足

默认情况下，Salt Master 会在其作业快速缓存中保存每个 Minion 针对每个作业返回的内容。以后便可使用快速缓存来查找之前的作业的结果。快速缓存目录默认为 `/var/cache/salt/master/jobs/`。

每个 Minion 针对每个作业返回的内容都保存在一个文件中。久而久之，此目录会变得非常大，具体大小取决于发布的作业数量和 `/etc/salt/master` 文件中 `keep_jobs` 选项的值。`keep_jobs` 用于设置应将有关过去的 Minion 作业的信息保留多少小时（默认值为 24）。

```
keep_jobs: 24
```

重要：请勿将 keep_jobs 设置为 0

如果将 `keep_jobs` 设置为“0”，则作业快速缓存清除程序永不运行，可能会导致分区变满。

要禁用作业快速缓存，请将 `job_cache` 设置为“False”：

```
job_cache: False
```

提示：恢复因作业快速缓存而变满的分区

当由于 `keep_jobs` 设置不当而导致包含作业快速缓存文件的分区变满时，请执行以下步骤释放磁盘空间并改进作业快速缓存设置：

1. 停止 Salt Master 服务：

   ```
   root@master # systemctl stop salt-master
   ```

2. 通过编辑 `/etc/salt/master` 来更改与作业快速缓存相关的 Salt Master 配置：

   ```
   job_cache: False
   keep_jobs: 1
   ```

3. 清除 Salt Master 作业快速缓存：

   ```
   rm -rfv /var/cache/salt/master/jobs/*
   ```

4. 启动 Salt Master 服务：

   ```
   root@master # systemctl start salt-master
   ```

术语表

常规

CRUSH、CRUSH 地图

通过计算数据存储位置来确定如何存储和检索数据的算法。CRUSH 需要获取集群的地图来以伪随机的方式在 OSD 中存储和检索数据，并以一致的方式在整个集群中分布数据。

OSD 节点

一个集群节点，用于存储数据、处理数据复制、恢复、回填、重新平衡以及通过检查其他 Ceph OSD 守护进程为 Ceph monitor 提供某些监视信息。

存储池

用于存储磁盘映像等对象的逻辑分区。

桶

将其他节点聚合成物理位置层次结构的一个点。

监视器节点，MON

用于维护集群状态地图的集群节点，包括监视器地图或 OSD 地图。

管理节点

用于运行 `ceph-deploy` 实用程序以在 OSD 节点上部署 Ceph 的节点。

节点

Ceph 集群中的任何一台计算机或服务器。

规则组

用于确定存储池的数据归置的规则。

Ceph 特定术语

Ceph 存储集群

用于存储用户数据的存储软件的核心集合。此类集合由 Ceph monitor 和 OSD 组成。

也称为“Ceph 对象存储”。

对象网关

Ceph 对象存储的 S3/Swift 网关组件。

A 手动安装 Ceph 的示例过程

下面的过程说明了手动安装 Ceph 存储集群时需要使用的命令。

1. 为要运行的 Ceph 服务生成密钥机密。可以使用下面的命令生成密钥机密：

```
python -c "import os ; import struct ; import time; import base64 ; \
 key = os.urandom(16) ; header =
 struct.pack('<hiih',1,int(time.time()),0,len(key)) ; \
 print base64.b64encode(header + key)"
```

2. 将密钥添加到相关的密钥环。添加顺序依次为 `client.admin`、监视器、其他相关服务（例如 OSD、对象网关或 MDS）：

```
ceph-authtool -n client.admin \
 --create-keyring /etc/ceph/ceph.client.admin.keyring \
 --cap mds 'allow *' --cap mon 'allow *' --cap osd 'allow *'
ceph-authtool -n mon. \
 --create-keyring /var/lib/ceph/bootstrap-mon/ceph-osceph-03.keyring \
 --set-uid=0 --cap mon 'allow *'
ceph-authtool -n client.bootstrap-osd \
 --create-keyring /var/lib/ceph/bootstrap-osd/ceph.keyring \
 --cap mon 'allow profile bootstrap-osd'
ceph-authtool -n client.bootstrap-rgw \
 --create-keyring /var/lib/ceph/bootstrap-rgw/ceph.keyring \
 --cap mon 'allow profile bootstrap-rgw'
ceph-authtool -n client.bootstrap-mds \
 --create-keyring /var/lib/ceph/bootstrap-mds/ceph.keyring \
 --cap mon 'allow profile bootstrap-mds'
```

3. 创建 monmap，这是集群中所有监视器的数据库：

```
monmaptool --create --fsid eaac9695-4265-4ca8-ac2a-f3a479c559b1 \
 /tmp/tmpuuhxm3/monmap
monmaptool --add osceph-02 192.168.43.60 /tmp/tmpuuhxm3/monmap
monmaptool --add osceph-03 192.168.43.96 /tmp/tmpuuhxm3/monmap
monmaptool --add osceph-04 192.168.43.80 /tmp/tmpuuhxm3/monmap
```

4. 在该数据库中创建新的密钥环，并从管理员和监视器的密钥环导入密钥。然后使用密钥环来启动监视器：

```
ceph-authtool --create-keyring /tmp/tmpuuhxm3/keyring \
 --import-keyring /var/lib/ceph/bootstrap-mon/ceph-osceph-03.keyring
ceph-authtool /tmp/tmpuuhxm3/keyring \
 --import-keyring /etc/ceph/ceph.client.admin.keyring
sudo -u ceph ceph-mon --mkfs -i osceph-03 \
 --monmap /tmp/tmpuuhxm3/monmap --keyring /tmp/tmpuuhxm3/keyring
systemctl restart ceph-mon@osceph-03
```

5. 在 `systemd` 中检查监视器状态：

```
systemctl show --property ActiveState ceph-mon@osceph-03
```

6. 检查 Ceph 是否正在运行并报告监视器状态：

```
ceph --cluster=ceph \
 --admin-daemon /var/run/ceph/ceph-mon.osceph-03.asok mon_status
```

7. 使用现有密钥检查特定服务的状态：

```
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin -f json-pretty status
[...]
ceph --connect-timeout 5 \
 --keyring /var/lib/ceph/bootstrap-mon/ceph-osceph-03.keyring \
 --name mon. -f json-pretty status
```

8. 从现有 Ceph 服务中导入密钥环并检查状态：

```
ceph auth import -i /var/lib/ceph/bootstrap-osd/ceph.keyring
ceph auth import -i /var/lib/ceph/bootstrap-rgw/ceph.keyring
ceph auth import -i /var/lib/ceph/bootstrap-mds/ceph.keyring
ceph --cluster=ceph \
 --admin-daemon /var/run/ceph/ceph-mon.osceph-03.asok mon_status
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin -f json-pretty status
```

9. 使用 XFS 文件系统为 OSD 准备磁盘/分区：

```
ceph-disk -v prepare --fs-type xfs --data-dev --cluster ceph \
 --cluster-uuid eaac9695-4265-4ca8-ac2a-f3a479c559b1 /dev/vdb
ceph-disk -v prepare --fs-type xfs --data-dev --cluster ceph \
 --cluster-uuid eaac9695-4265-4ca8-ac2a-f3a479c559b1 /dev/vdc
[...]
```

10. 激活分区：

```
ceph-disk -v activate --mark-init systemd --mount /dev/vdb1
ceph-disk -v activate --mark-init systemd --mount /dev/vdc1
```

11. 如果 SUSE Enterprise Storage 为 2.1 或更低版本，请创建默认存储池：

```
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .users.swift 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .intent-log 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .rgw.gc 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .users.uid 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .rgw.control 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .users 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .usage 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .log 16 16
ceph --connect-timeout 5 --keyring /etc/ceph/ceph.client.admin.keyring \
 --name client.admin osd pool create .rgw 16 16
```

12. 根据引导密钥创建对象网关实例密钥：

```
ceph --connect-timeout 5 --cluster ceph --name client.bootstrap-rgw \
 --keyring /var/lib/ceph/bootstrap-rgw/ceph.keyring auth get-or-create \
```

```
 client.rgw.0dc1e13033d2467eace46270f0048b39 osd 'allow rwx' mon 'allow rw'
 \
 -o /var/lib/ceph/radosgw/ceph-rgw.rgw_name/keyring
```

13. 启用并启动对象网关：

```
systemctl enable ceph-radosgw@rgw.rgw_name
systemctl start ceph-radosgw@rgw.rgw_name
```

14. （可选）根据引导密钥创建 MDS 实例密钥，然后启用并启动它：

```
ceph --connect-timeout 5 --cluster ceph --name client.bootstrap-mds \
 --keyring /var/lib/ceph/bootstrap-mds/ceph.keyring auth get-or-create \
 mds.mds.rgw_name osd 'allow rwx' mds allow mon \
 'allow profile mds' \
 -o /var/lib/ceph/mds/ceph-mds.rgw_name/keyring
systemctl enable ceph-mds@mds.rgw_name
systemctl start ceph-mds@mds.rgw_name
```

B 文档更新

本章列出了本文档自 SUSE Enterprise Storage 1 初始版本发布以来的内容更改。

文档在以下日期进行了更新：

B.1 2018 年 9 月（SUSE Enterprise Storage 5.5 发布）

一般更新

- 扩充了第 15.4.3 节 “管理 RADOS 块设备 (RBD)”，主要是添加了有关快照的小节 (Fate #325642)。
- 添加了第 7.3 节 “存储池迁移”(Fate#322006)。
- 在第 3 章 “操作 Ceph 服务”中插入了第 3.2 节 “使用 DeepSea 重启动 Ceph 服务”。

错误修复

- 在第 1.7 节 “恢复重新安装的 OSD 节点” https://bugzilla.suse.com/show_bug.cgi?id=1095937 中，添加了 `salt` 的 `state.apply` 部分。
- 添加了第 15.1.1 节 “启用使用 SSL 安全访问 openATTIC 的功能” https://bugzilla.suse.com/show_bug.cgi?id=1083216 。

- 添加了第 11.12 节 “使用 HAProxy 在对象网关服务器间实现负载平衡” https://bugzilla.suse.com/show_bug.cgi?id=1093513 。
- 在第 6.6 节 “在同一个节点上混用 SSD 和 HDD” https://bugzilla.suse.com/show_bug.cgi?id=1093583 中，更新了有关自定义 `ceph.conf` 的信息。
- 添加了第 6.5 节 “整理 (Scrub)” https://bugzilla.suse.com/show_bug.cgi?id=1079256 。
- 在第 18.9 节 “Ceph 的防火墙设置” https://bugzilla.suse.com/show_bug.cgi?id=1070087 中，优化了防火墙设置的端口列表。
- 在第 3.2.2 节 “重启动特定服务” https://bugzilla.suse.com/show_bug.cgi?id=1091075 中，根据 DeepSea 版本区分了重启动的角色。
- 添加了第 1.4 节 “重新部署监视器节点” https://bugzilla.suse.com/show_bug.cgi?id=1038731 。
- 添加了第 11.9.1.1 节 “动态重分片” https://bugzilla.suse.com/show_bug.cgi?id=1076001 。
- 添加了第 11.9 节 “桶索引分片” https://bugzilla.suse.com/show_bug.cgi?id=1076000 。
- 在第 11.6 节 “为对象网关启用 HTTPS/SSL”中，为对象网关 SSL 更新了 DeepSea 方法（ https://bugzilla.suse.com/show_bug.cgi?id=1083756 和 https://bugzilla.suse.com/show_bug.cgi?id=1077809 ）。
- 如果 DeepSea 中发生了更改，则需要使用对象网关对 NFS Ganesha 部署进行修改。请参见第 14.3.1 节 “NFS Ganesha 的不同对象网关用户” (https://bugzilla.suse.com/show_bug.cgi?id=1058821)。一旦超出每个 OSD 的归置组限制，`ceph osd pool create` 便会失败。请参见第 7.2.2 节 “创建存储池”(https://bugzilla.suse.com/show_bug.cgi?id=1076509)。
- 添加了第 1.7 节 “恢复重新安装的 OSD 节点”(https://bugzilla.suse.com/show_bug.cgi?id=1057764)。
- 在第 1.12 节 “运行时 Ceph 配置”中添加了可靠性警告 (https://bugzilla.suse.com/show_bug.cgi?id=989349)。

- 添加了第 11.2 节 “部署对象网关”(https://bugzilla.suse.com/show_bug.cgi?id=1088895)。
- 在第 7.5.3 节 “全局压缩选项”中，从压缩算法列表中删除了 `lz4` (https://bugzilla.suse.com/show_bug.cgi?id=1088450)。
- 在第 1.6 节 “删除 OSD”中添加了有关删除多个 OSD 的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1070791)。
- 添加了第 18.2 节 “在不重新平衡的情况下停止 OSD”(https://bugzilla.suse.com/show_bug.cgi?id=1051039)。
- 在《部署指南》, 第 11 章 “安装 CephFS”, 第 11.2.2 节 “配置元数据服务器”中添加了 MDS 快速缓存大小可配置项 (https://bugzilla.suse.com/show_bug.cgi?id=1062692)。
- 添加了第 11.10 节 “集成 OpenStack Keystone”(https://bugzilla.suse.com/show_bug.cgi?id=1077941)。
- 在《部署指南》, 第 10 章 “安装 iSCSI 网关”, 第 10.4.3 节 “通过 iSCSI 导出 RBD 映像”中添加了有关同步 iSCSI 网关配置的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1073327)。
- 如果 DeepSea 中发生了更改，则需要使用对象网关对 NFS Ganesha 部署进行修改。请参见第 14.3.1 节 “NFS Ganesha 的不同对象网关用户”(https://bugzilla.suse.com/show_bug.cgi?id=1058821)。

B.2 2017 年 11 月（文档维护性更新）

一般更新

- 添加了第 18.11 节 “更换存储磁盘”(Fate#321032)。

错误修复

- 添加了第 9.4 节 “含 RADOS 块设备的纠删码池”(https://bugzilla.suse.com/show_bug.cgi?id=1075158)。
- 添加了有关向 OSD 节点添加磁盘的小节。请参见第 1.5 节 “为节点添加 OSD”(https://bugzilla.suse.com/show_bug.cgi?id=1066005)。
- 使用 `salt-run remove.osd` 时需要提供 OSD_ID 数字，但不需要前导 `osd.`。请参见第 1.6 节 “删除 OSD”。
- 使用 `ceph tell` 时需要提供 OSD_ID 数字和前导 `osd.`。请参见第 20.8 节 “查找运行缓慢的 OSD”。
- 添加了第 20.11 节 “`/var` 空间不足”(https://bugzilla.suse.com/show_bug.cgi?id=1069255)。
- 添加了第 11.1 节 “对象网关限制和命名限制”(https://bugzilla.suse.com/show_bug.cgi?id=1067613)。
- 更正了第 8.5.1 节 “rbd-mirror 守护进程”中的 rbd-mirror 启动和停止命令 (https://bugzilla.suse.com/show_bug.cgi?id=1068061)。

B.3 2017 年 10 月（SUSE Enterprise Storage 5 发布）

一般更新

- 删除了 Calamari，由 openATTIC 取代。
- 添加了第 15.4.6 节 “管理 NFS Ganesha”(Fate#321620)。

- 添加了第 8.4 节 “rbdmap：在引导时映射 RBD 设备”
- 添加了第 8 章 “RADOS 块设备”(Fate#321061)。
- 添加了第 15.4.9 节 “管理对象网关用户和桶”(Fate#320318)。
- 添加了第 11.8 节 “LDAP 身份验证”(Fate#321631)。
- 添加了第 13.4 节 “多个活动 MDS 守护进程（主动/主动 MDS）”(Fate#322976)。
- 添加了第 15.4.7 节 “管理 iSCSI 网关”(Fate#321370)。
- 添加了 openATTIC 的 iSCSI 网关和对象网关配置，请参见第 15.1.5 节 “对象网关管理”和第 15.1.6 节 “iSCSI 网关管理”（Fate #320318 和 #321370）。
- 更新了第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”（https://bugzilla.suse.com/show_bug.cgi?id=1036495 。https://bugzilla.suse.com/show_bug.cgi?id=1031444 ）。
- RBD 映像现在可存储在 EC 池中，请参见第 8.1.2 节 “在纠删码池中创建块设备映像”(https://bugzilla.suse.com/show_bug.cgi?id=1040752)。
- 添加了有关备份 DeepSea 配置的章节，请参见《部署指南》, 第 6 章 “备份集群配置”(https://bugzilla.suse.com/show_bug.cgi?id=1046497)。
- 对象网关故障转移和灾难恢复，请参见第 11.11.12 节 “故障转移和灾难恢复”(https://bugzilla.suse.com/show_bug.cgi?id=1036084)。
- BlueStore 允许针对存储池进行数据压缩，请参见第 7.5 节 “数据压缩”(FATE#318582)。
- 允许进行 CephFS 的 CIFS 导出，请参见《部署指南》, 第 13 章 “通过 Samba 导出 CephFS”(FATE#321622)。
- 添加了集群重引导的过程，请参见第 1.10 节 “停止或重引导集群”(https://bugzilla.suse.com/show_bug.cgi?id=1047638)。
- DeepSea 阶段 0 无需重引导即可更新，请参见第 1.9 节 “更新集群节点”。
- 取代了 `ceph fs`，请参见第 13.4.3 节 “减小级别数”和第 13.4.2 节 “增加 MDS 活动集群的大小”(https://bugzilla.suse.com/show_bug.cgi?id=1047638)。
- 添加了第 18.10 节 “测试网络性能”(FATE#321031)。

错误修复

- 在第 6.6 节 “在同一个节点上混用 SSD 和 HDD”中，更新了命令输出以反映存储类别 (https://bugzilla.suse.com/show_bug.cgi?id=1061299)。
- 在第 11.5.1 节 “访问对象网关”中，swift 客户端包现在是“Public Cloud”模块的一部分 (https://bugzilla.suse.com/show_bug.cgi?id=1057591)。
- 添加了第 1.12 节 “运行时 Ceph 配置”(https://bugzilla.suse.com/show_bug.cgi?id=1061435)。
- 在第 6.6 节 “在同一个节点上混用 SSD 和 HDD”中，将命令更改为配置选项 (https://bugzilla.suse.com/show_bug.cgi?id=1059561)。
- 在第 6.6 节 “在同一个节点上混用 SSD 和 HDD”中，更新了 `ceph osd pool set` 命令以与 Luminous 语法相匹配 (https://bugzilla.suse.com/show_bug.cgi?id=1059593)。
- 在提示：绑定到多个端口中，CivetWeb 绑定到多个端口 (https://bugzilla.suse.com/show_bug.cgi?id=1055181)。
- 在第 11.4 节 “配置参数”中加入了 3 个影响性能的对象网关选项 (https://bugzilla.suse.com/show_bug.cgi?id=1052983)。
- 导入了第 1.11 节 “自定义 `ceph.conf` 文件”并添加了阶段 3 的需要 (https://bugzilla.suse.com/show_bug.cgi?id=1057273)。
- 在第 16.1 节 “配置 Ceph”中添加了 libvirt 密钥环创建步骤 (https://bugzilla.suse.com/show_bug.cgi?id=1055610)。
- 添加了例 1.1 “从集群中删除 Salt Minion”(https://bugzilla.suse.com/show_bug.cgi?id=1054516)。
- 更新了第 4.2 节 “监视集群”(https://bugzilla.suse.com/show_bug.cgi?id=1053638)。
- 在第 15.1 节 “openATTIC 部署和配置”中将 Salt REST API 变量设为可选（https://bugzilla.suse.com/show_bug.cgi?id=1054748 和 https://bugzilla.suse.com/show_bug.cgi?id=1054749）。

- 在第 15.1.3 节 “openATTIC 初始设置”中删除了 `oaconfig install` (https://bugzilla.suse.com/show_bug.cgi?id=1054747)。
- 在第 7.1 节 “将存储池与应用关联”中添加了有关显示存储池元数据的章节 (https://bugzilla.suse.com/show_bug.cgi?id=1053327)。
- 在第 4.1 节 “检查集群运行状况”中导入了运行状况代码列表 (https://bugzilla.suse.com/show_bug.cgi?id=1052939)。
- 更新了第 15.4.9.1 节 “添加新的对象网关用户”和第 15.4.9.3 节 “编辑对象网关用户”中的截图及相关文本（https://bugzilla.suse.com/show_bug.cgi?id=1051814 和 https://bugzilla.suse.com/show_bug.cgi?id=1051816）。
- 在第 15.4.9 节 “管理对象网关用户和桶”中添加了对象网关桶 (https://bugzilla.suse.com/show_bug.cgi?id=1051800)。
- 在第 13.1.3 节 “挂载 CephFS”的挂载示例中包含了 cephx (https://bugzilla.suse.com/show_bug.cgi?id=1053022)。
- 更新并改进了第 7.2.4 节 “删除存储池”中的存储池删除说明 (https://bugzilla.suse.com/show_bug.cgi?id=1052981)。
- 在第 7.5.2 节 “存储池压缩选项”中添加了压缩算法说明 (https://bugzilla.suse.com/show_bug.cgi?id=1051457)。
- 替换了第 20.10 节 “网络问题导致集群性能不佳”中的网络诊断和基准测试 (https://bugzilla.suse.com/show_bug.cgi?id=1050190)。
- 扩展了第 20.5 节 ““nn pg 停滞在非活动状态”状态讯息”(https://bugzilla.suse.com/show_bug.cgi?id=1050183)。
- 在第 20.4 节 ““每个 OSD 的 PG 数过多”状态讯息”中提到了存储池重新创建 (https://bugzilla.suse.com/show_bug.cgi?id=1050178)。
- 更正了《部署指南》, 第 9 章 “Ceph Object Gateway”, 第 9.1 节 “手动安装对象网关”中 `ceph.conf` 内的 RGW 段落名称 (https://bugzilla.suse.com/show_bug.cgi?id=1050170)。
- 更新了第 4.3 节 “检查集群的用量统计数字”和第 4.2 节 “监视集群”中的命令输出 (https://bugzilla.suse.com/show_bug.cgi?id=1050175)。

- 删除了第 4.1 节 “检查集群运行状况”中的预防性 HEALTCH_WARN 段落 (https://bugzilla.suse.com/show_bug.cgi?id=1050174)。
- 更正了第 11.5.2.1 节 “添加 S3 和 Swift 用户”中的 sudo (https://bugzilla.suse.com/show_bug.cgi?id=1050177)。
- 删除了第 20.2 节 “使用 `rados` 发送大型对象失败并显示“OSD 已满””中对 RADOS striper 的引用 (https://bugzilla.suse.com/show_bug.cgi?id=1050171)。
- 改进了第 19.5 节 “日记磁盘出现故障时会发生什么情况？”中有关日记失败导致 OSD 故障的相关章节 (https://bugzilla.suse.com/show_bug.cgi?id=1050169)。
- 在第 1.9 节 “更新集群节点”中添加了阶段 0 期间有关 `zypper patch` 的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1050165)。
- 添加了第 7.1 节 “将存储池与应用关联”(https://bugzilla.suse.com/show_bug.cgi?id=1049940)。
- 改进了第 20.9 节 “解决时钟偏差警告”中的时间同步信息 (https://bugzilla.suse.com/show_bug.cgi?id=1050186)。
- 用正确的“纠删码池”取代了“纠删池”(https://bugzilla.suse.com/show_bug.cgi?id=1050093)。
- 用 `systemctl` 取代了 `rcceph` (https://bugzilla.suse.com/show_bug.cgi?id=1050111)。
- 更新了第 13.1.1 节 “客户端准备”中的 CephFS 挂载准备 (https://bugzilla.suse.com/show_bug.cgi?id=1049451)。
- 更正了第 16.1 节 “配置 Ceph”中的 `qemu-img` 命令 (https://bugzilla.suse.com/show_bug.cgi?id=1047190)。
- 在第 1.3 节 “删除和重新安装集群节点”中指定了删除角色时要运行的 DeepSea 阶段 (https://bugzilla.suse.com/show_bug.cgi?id=1047430)。
- 添加了新的 DeepSea 角色“Ceph Manager”(https://bugzilla.suse.com/show_bug.cgi?id=1047472)。
- 调整了第 13.1.1 节 “客户端准备”中 12 SP3 的简介 (https://bugzilla.suse.com/show_bug.cgi?id=1043739)。

- 更正了第 16.4 节 “配置 VM”中 XML 实体的拼写错误 (https://bugzilla.suse.com/show_bug.cgi?id=1042917)。
- 在第 1.3 节 “删除和重新安装集群节点”中添加了针对角色删除重新运行 DeepSea 阶段 2-5 的信息 (https://bugzilla.suse.com/show_bug.cgi?id=1041899)。
- 在第 18.9 节 “Ceph 的防火墙设置”中添加了需要在 SUSE 防火墙中打开的对象网关、iSCSI 网关和 NFS Ganesha 端口号 (https://bugzilla.suse.com/show_bug.cgi?id=1034081)。
- 添加了 CRUSH 地图树迭代的说明，请参见第 6.3.1 节 “在节点树中迭代”。
- 在 CRUSH 规则中添加了 indep 参数，请参见第 6.3.2 节 “firstn 和 indep”。(https://bugzilla.suse.com/show_bug.cgi?id=1025189)
- 通过 `/etc/fstab` 挂载 CephFS 需要 `_netdev` 参数。请参见第 13.3 节 “`/etc/fstab` 中的 CephFS”(https://bugzilla.suse.com/show_bug.cgi?id=989349)
- 在第 8.2 节 “挂载和卸载 RBD 映像”中添加了有关现有 `rbdmap` `systemd` 服务文件的提示 (https://bugzilla.suse.com/show_bug.cgi?id=1015748)。
- 在第 10.6.1.4 节 “对命中集使用 GMT”中添加了对 `use_gmt_hitset` 选项的说明 (https://bugzilla.suse.com/show_bug.cgi?id=1024522)。
- 将挂载 CephFS 移回了管理指南，并在第 13.1.1 节 “客户端准备”中添加了客户端准备章节 (https://bugzilla.suse.com/show_bug.cgi?id=1025447)。

B.4 2017 年 2 月（SUSE Enterprise Storage 4 维护性更新 1 发布）

一般更新

- 添加了第 14 章 “NFS Ganesha：通过 NFS 导出 Ceph 数据”。

错误修复

- 在《部署指南》, 第 5 章 “从旧版本升级”, 第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”中添加了指向 CRUSH 可调变量的链接 (https://bugzilla.suse.com/show_bug.cgi?id=1024718)。
- 在《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”中检查是否启用且启动 `systemd` 服务 (https://bugzilla.suse.com/show_bug.cgi?id=1023752)。
- 在第 13.1.3 节 “挂载 CephFS”中告知用户需要具有 MDS 根目录下的读取权限 (https://bugzilla.suse.com/show_bug.cgi?id=1014051)。
- 将 iSCSI 网关升级引用移至《部署指南》, 第 5 章 “从旧版本升级”, 第 5.2 节 “一般升级过程”(https://bugzilla.suse.com/show_bug.cgi?id=1014194)。
- 在《部署指南》, 第 5 章 “从旧版本升级”, 第 5.2 节 “一般升级过程”中为升级工作流程添加了管理节点 (https://bugzilla.suse.com/show_bug.cgi?id=1012155)。
- 重新编写了第 3 章 “操作 Ceph 服务”以免统配服务 (https://bugzilla.suse.com/show_bug.cgi?id=1009500)。
- 删除了介绍 Salt sls 文件的附录 (https://bugzilla.suse.com/show_bug.cgi?id=1014155)。
- 添加了有关 XFS 损坏文件系统的第 20.3 节 “XFS 文件系统损坏”(https://bugzilla.suse.com/show_bug.cgi?id=1012551)。
- 添加了有关时间同步的第 18.3 节 “节点时间同步”(https://bugzilla.suse.com/show_bug.cgi?id=1009653)。
- 以必须擦除磁盘的信息更新了《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”, 第 4.3 节 “集群部署”(https://bugzilla.suse.com/show_bug.cgi?id=1014039)。

B.5 2016 年 12 月（SUSE Enterprise Storage 4 发布）

一般更新

- 对引入 DocBook“部分”的整篇文档的结构进行了调整，以将相关章节组合在一起。
- 引入了第 15 章 “openATTIC”(Fate #321085)。

《部署指南》, 第 5 章 “从旧版本升级”

- 使用《部署指南》, 第 5 章 “从旧版本升级”, 第 5.4 节 “从 SUSE Enterprise Storage 4（DeepSea 部署）升级到版本 5”取代了旧的升级过程。

错误修复

- 在《部署指南》, 第 2 章 “硬件要求和建议”, 第 2.1.1 节 “最低要求”中增加了 OSD 的内存要求 (https://bugzilla.suse.com/show_bug.cgi?id=982496)。
- 改进了《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”(https://bugzilla.suse.com/show_bug.cgi?id=993499)。
- 改进了第 20.9 节 “解决时钟偏差警告”(https://bugzilla.suse.com/show_bug.cgi?id=999856)。
- 改进了《部署指南》, 第 11 章 “安装 CephFS”, 第 11.2.1 节 “添加元数据服务器”(https://bugzilla.suse.com/show_bug.cgi?id=992769)。

B.6 2016 年 6 月（SUSE Enterprise Storage 3 发布）

一般更新

- 添加了附录 A “手动安装 Ceph 的示例过程”。
- 添加了第 5 章 “使用 cephx 进行身份验证”。

- 添加了第 11.11 节 “多站点对象网关”(Fate#320602)。
- 添加了《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”。
- 添加了第 13 章 “集群文件系统”(Fate#318586)。

第 12 章 “Ceph iSCSI 网关”

- 添加了《部署指南》, 第 10 章 “安装 iSCSI 网关”, 第 10.4.4 节 “可选设置”。
- 添加了第 12.1.1.1 节 “多路径配置”。

错误修复

- 改进了第 10.6 节 “设置示例分层存储”中设置热存储和冷存储的过程并添加了第 10.6.1 节 “配置快速缓存层”(https://bugzilla.suse.com/show_bug.cgi?id=982607)。
- 在《部署指南》, 第 11 章 “安装 CephFS”, 第 11.2.1 节 “添加元数据服务器”中添加了用于在 MDS 服务器上安装 Ceph 的命令 (https://bugzilla.suse.com/show_bug.cgi?id=993820)。
- 在《部署指南》, 第 10 章 “安装 iSCSI 网关”, 第 10.4 节 “安装和配置”中，当创建 RBD 卷时，格式 1 不再是默认值（改为格式 2）(https://bugzilla.suse.com/show_bug.cgi?id=987992)。
- 在第 6.3 节 “规则组”中添加了有关增加规则组数量的注释 (https://bugzilla.suse.com/show_bug.cgi?id=997051)。
- 指定了哪些客户端能够迁移到最佳可调变量 (https://bugzilla.suse.com/show_bug.cgi?id=982995)。
- 将《部署指南》, 第 10 章 “安装 iSCSI 网关”, 第 10.4.4 节 “可选设置”拆分成《部署指南》, 第 10 章 “安装 iSCSI 网关”, 第 10.4.5 节 “高级设置”并添加了配置选项说明 (https://bugzilla.suse.com/show_bug.cgi?id=986037)。
- 添加了第 6.6 节 “在同一个节点上混用 SSD 和 HDD”(https://bugzilla.suse.com/show_bug.cgi?id=982375)。
- 更新了《部署指南》, 第 2 章 “硬件要求和建议”, 第 2.1.1 节 “最低要求”中的最低建议 (https://bugzilla.suse.com/show_bug.cgi?id=981642)。

- 更正了第 8.3.3 节 “分层”中有关快照克隆的支持信息 (https://bugzilla.suse.com/show_bug.cgi?id=982713)。
- 改进了第 6.2 节 “桶”中对“桶”的解释 (https://bugzilla.suse.com/show_bug.cgi?id=985047)。
- 在《部署指南》, 第 2 章 “硬件要求和建议”, 第 2.2 节 “监视器节点”中明确解释了非混合工作负载的概念 (https://bugzilla.suse.com/show_bug.cgi?id=982497)。
- 更新了《部署指南》, 第 2 章 “硬件要求和建议”, 第 2.1.1 节 “最低要求”中 OSD 的 RAM 要求 (https://bugzilla.suse.com/show_bug.cgi?id=982496)。
- 更正了第 7.2.7 节 “设置存储池的值”中 `hit_set_count` 的默认值，并在第 10.6 节 “设置示例分层存储”中添加了含有外部链接的注释 (https://bugzilla.suse.com/show_bug.cgi?id=982284)。
- 更新了第 6.4.2 节 “添加/移动 OSD”、第 6.3 节 “规则组”和第 6.2 节 “桶”中的若干处，以与当前 Ceph 版本相匹配 (https://bugzilla.suse.com/show_bug.cgi?id=982563)。
- 在第 7.2.7 节 “设置存储池的值”中，添加了以下巡回检测参数的解释：`hashpspool`、`expected_num_objects`、`cache_target_dirty_high_ratio`、`hit_set_grade_decay_rate`、`hit_set_grade_search_last_n`、`fast_read`、`scrub_min_interval`、`scrub_max_interval`、`deep_scrub_interval`、`nodelete`、`nopgchange`、`nosizechangee`、`noscrub`、`nodeep-scrub` (https://bugzilla.suse.com/show_bug.cgi?id=982512)。
- 添加了第 18.7 节 “如何对包含 OSD 日记的 OSD 使用现有分区”(https://bugzilla.suse.com/show_bug.cgi?id=970104)。
- 在《部署指南》, 第 2 章 “硬件要求和建议”, 第 2.1.1 节 “最低要求”和第 6.1 节 “设备数”中删除了有关 OSD 磁盘归置的 RAID 建议 (https://bugzilla.suse.com/show_bug.cgi?id=981611)。
- 在第 6.2 节 “桶”中，更新了 CRUSH 地图的桶的默认设置 (https://bugzilla.suse.com/show_bug.cgi?id=981756)。

- 删除了“data”和“metadata”池，不再是默认值 (https://bugzilla.suse.com/show_bug.cgi?id=981758)。
- 在第 12 章 “Ceph iSCSI 网关”中，更正了带商标的第三方产品名称，由实体取代 (https://bugzilla.suse.com/show_bug.cgi?id=983018)。
- 在受影响的章节中，将对象网关服务名称更新为 `ceph-radosgw@radosgw.gateway_name` (https://bugzilla.suse.com/show_bug.cgi?id=980594)。
- 添加了第 10.2 节 “需考虑的要点”(https://bugzilla.suse.com/show_bug.cgi?id=968290)。
- 更改了第 6.3 节 “规则组”中 `min_size` 的默认值 (https://bugzilla.suse.com/show_bug.cgi?id=977556)。
- 更正了《部署指南》, 第 4 章 “使用 DeepSea/Salt 部署”中的 `master:dns_name_of_salt_master` 选项 (https://bugzilla.suse.com/show_bug.cgi?id=977187)。
- 为对象网关主机添加了 `rgw.` 前缀 (https://bugzilla.suse.com/show_bug.cgi?id=974472)。
- 在第 20.5 节 ““nn pg 停滞在非活动状态”状态讯息”中添加了有关一直处于停滞状态的 PG 的信息 (https://bugzilla.suse.com/show_bug.cgi?id=968067)。

B.7 2016 年 1 月（SUSE Enterprise Storage 2.1 发布）

一般更新

- 删除了 Btrfs，因为自 SUSE Enterprise Storage 2 起不再支持 Btrfs。
- 将第 9.3 节 “纠删码池和快速缓存层”从第 10 章 “快速缓存分层”移到第 9 章 “纠删码池”，以使所提供的信息顺序正确。
- 添加了第 12 章 “Ceph iSCSI 网关”。

第 2 章 “简介”

- 删除了第 4 章 “确定集群状态”中的检查 MDS 状态一节，因为尚未涵盖 MDS。

第 17 章 “Ceph 用作 QEMU KVM 实例的后端”

- 添加了第 17.1 节 “安装”。

错误修复

- 在清除已过时的集群时添加了 `systemctl stop cthulhu.service` (https://bugzilla.suse.com/show_bug.cgi?id=967849)。
- 更正了拼写错误 (https://bugzilla.suse.com/show_bug.cgi?id=967937)
- 更正了 `ceph-deploy rgw` 命令语法中的拼写错误 (https://bugzilla.suse.com/show_bug.cgi?id=962976)。
- 调整了整个第 11 章 “Ceph Object Gateway”的结构，添加了第 11.3 节 “操作对象网关服务”、第 11.5 节 “管理对象网关的访问方式”和第 11.5.2.3 节 “更改 S3 和 Swift 用户的访问钥与机密密钥” (https://bugzilla.suse.com/show_bug.cgi?id=946873)。
- 在第 4 章 “确定集群状态”中将“监视集群”重命名为“确定集群状态”(https://bugzilla.suse.com/show_bug.cgi?id=958302)。
- 添加了第 20.4 节 ““每个 OSD 的 PG 数过多”状态讯息”(https://bugzilla.suse.com/show_bug.cgi?id=948375)。
- 建议客户手动阻止 Apache 监听默认端口 80（如果客户喜欢使用其他端口号的话）(https://bugzilla.suse.com/show_bug.cgi?id=942703)。
- 删除了第 11 章 “Ceph Object Gateway”中出现的 FastCGI 和文件引用 (https://bugzilla.suse.com/show_bug.cgi?id=946877)。
- 添加了安装/迁移对象网关实例的 `ceph-deploy` 方法 (https://bugzilla.suse.com/show_bug.cgi?id=946771)。
- 更正了 Apache 支持信息 (https://bugzilla.suse.com/show_bug.cgi?id=946769)。

B.8 2015 年 10 月（SUSE Enterprise Storage 2 发布）

常规

- 添加了《部署指南》, 第 5 章 “从旧版本升级”。
- 添加了第 16 章 “将 `libvirt` 与 Ceph 搭配使用”。
- 添加了第 17 章 “Ceph 用作 QEMU KVM 实例的后端”。

第 11 章 “Ceph Object Gateway”

- 更正了《部署指南》, 第 9 章 “Ceph Object Gateway”, 第 9.1.1 节 “对象网关配置”中的 `systemctl radosgw` 命令 (https://bugzilla.suse.com/show_bug.cgi?id=940483)。
- 用嵌入式 CivetWeb 取代了 Apache。