Red Hat Ceph Storage 4

生产型对象网关指南

为生产环境规划、设计和部署 Ceph 存储集群和 Ceph 对象网关集群。

Last Updated: 2021-08-26

Red Hat Ceph Storage 4 生产型对象网关指南

为生产环境规划、设计和部署 Ceph 存储集群和 Ceph 对象网关集群。

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律通告

摘要

本指南涵盖规划一个集群，考虑硬件，开发存储策略，配置网关和负载平衡器，以及使用 Ceph 对象网关。 红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始：master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看 CTO Chris Wright 信息。

目录

第 1 章 简介

欢迎使用 **Ceph 对象网关生产指南**。本指南涵盖为生产用途构建 Ceph 存储集群和 Ceph 对象网关集群的主题。

1.1. 受众

本指南适用于计划在生产环境中部署 Ceph 对象网关环境的用户。它提供一系列后续主题，用于规划、设计和部署生产 Ceph 存储集群和 Ceph 对象网关集群，并在适当的时候提供与一般 Ceph 文档的链接。

1.2. 假设

本指南假定读者对 Ceph 存储群集和 Ceph 对象网关有了基本了解。没有 Ceph 经验的读者应考虑设置小型 Ceph 测试环境或使用 Ceph Sandbox 环境，以便在继续阅读本指南之前熟悉 Ceph 概念。

本指南假定有一个单一站点群集，由同一区域中的单个 Ceph 存储集群和多个 Ceph 对象网关实例组成。本指南假定单站点群集将扩展至多区和多站点群集，方法是为每个 zone group 和 zone 重复本指南中的步骤以及二级区组和区域所需的命名修改。

1.3. SCOPE

本指南在为生产环境设置 Ceph 存储集群和 Ceph 对象网关时涵盖以下主题：

- 规划集群
- 考虑硬件
- 配置集群
- 部署 Ceph
- 开发存储策略
- 配置网关
- 其他用例

注意

本文档旨在补充硬件、安装、管理和 Ceph 对象网关指南。本指南不会替换其他指南。

第 2 章 规划集群

规划集群以与 Ceph 对象网关搭配使用涉及几个重要注意事项：

- 识别用例
- 选择数据持久性方法
- 考虑多站点部署

这些因素 在考虑硬件 时将产生显著影响。在选择硬件前请仔细考虑这些因素。

2.1. 识别用例

Ceph 存储能够提供许多不同类型的存储用例。对于 Ceph 对象存储，典型的用例是：

- **优化吞吐量：优化** 吞吐量的集群以确保快速访问数据。主机总线适配器(HBA)、具有快速顺序读/写特征的存储介质和高网络带宽为图形、流式音频和流传输视频等应用程序提供功能。优化吞吐量的集群还考虑写性能是否是一个考虑因素。使用 SSD 进行日志优化的集群可显著提高写入性能，这对于存储 CCTV 流的应用程序非常重要。优化吞吐量的集群应考虑主机总线适配器(HBA)控制器的吞吐量特征以及密集应用程序（如流传输 4K 视频）的网络吞吐量。基于 HBA 的硬件控制器比板载控制器提供显著的性能改进。

- **容量优化：** 优化容量的集群试图确保每 TB 的存储成本最低。容量优化的集群通常使用最便宜的存储媒体，通常会避免为应用程序（如不频繁访问的传统财务记录、旧电子邮件等）提供单独的 SSD 日志费用。

- **IOPS 优化：优化** IOPS 群集，旨在为读写密集型工作负载提供高性能。虽然 IOPS 优化的工作负载不像 Ceph 对象网关那样常见，但可使用 SSD、Flash 内存或 NVMe CRUSH 层次结构支持它们。

仔细考虑存储用例 **BEFORE** 考虑硬件，因为它可能会严重影响集群的价格和性能。例如，如果使用案例经过容量优化，且硬件更适合通过吞吐量优化的使用案例，硬件的成本将高于必要成本。相反，如果用例的吞吐量优化，且硬件更适合容量优化使用案例，则群集的性能可能会受到影响。

此外，请注意，由于 Ceph 对象网关支持存储策略，因此可以创建 CRUSH 层次结构， 以用于各种各样的场景，并通过 API 中支持的存储策略来调用它们。详情请参阅 创建数据放置策略。

2.2. 选择数据持续时间方法

群集设计还应考虑数据持久性策略。Ceph 存储使用复制或纠删代码来确保数据持久性。

在出现硬件故障时，复制存储在故障域中的一个或多个数据冗余副本。但是，冗余的数据副本规模可能会变得昂贵。例如，要存储 1 PB 字节并带有三倍复制的数据，将需要至少具有 3 PB 存储容量的集群。

红帽 Ceph 存储 4 的存储策略指南中的 纠删代码部分 介绍了纠删代码如何将数据存储为数据区块和编码区块。如果数据区块丢失，纠删代码可以使用剩余的数据区块和编码区块来恢复丢失的数据区块。纠删代码比复制更经济。例如，使用带有 8 个数据区块和 3 个编码区块的纠删代码提供与 3 个数据副本相同的冗余。但是，与复制相比，此类编码方案使用约 1.5x 的初始数据。

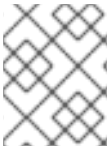
注意

数据存储 池可以使用纠删代码。存储服务数据和 bucket 索引的池使用复制。

2.3. 考虑多站点部署

设计集群的另一个重要方面是确定集群会位于一个数据中心站点，还是跨越多个数据中心站点。多站点群集得益于地理分散的故障切换和灾难恢复，如长期停电、农业、风暴、风暴、水水或其他灾难。此外，主动配置中的多站点群集能够以内容交付网络的方式将客户端应用程序定向到最接近的可用集群。对于流传输 4k 视频等吞吐量密集型工作负载而言，尽可能将数据放置在客户端上变得越来越重要。

有关多站点集群的详细信息，请参阅红帽 Ceph 存储 4 的 Ceph 对象网关配置和管理指南中 的多站点 章节。



注意

红帽建议识别域、zone group 和 zone 名称 BEFORE 创建 Ceph 存储池。按照惯例，某些池名称应预先带有区域名称。

第 3 章 考虑硬件

考虑硬件是在生产环境中构建 Ceph 存储集群和 Ceph 对象网关集群的重要部分。高级考虑包括：

- 考虑存储大小
- 考虑存储密度
- 考虑 Uninterrupted Power 供应商
- 考虑网络硬件
- 为用例选择硬件
- 为索引选择 Media
- 为 monitor 节点选择 Media

重要

请考虑以下因素： **BEFORE** 为集群识别和购买计算和网络硬件。

3.1. 考虑存储大小

设计集群时最重要的因素之一是确定存储要求（大小）。Ceph 存储设计为可以扩展到 PB 以上。以下示例是 Ceph 存储集群的常见大小。

- **small**: 250 TB
- **Medium：1** Ptabyte
- **大：2** PB 或更高版本.

规模调整应包括当前需求和近期的需求。请考虑网关客户端向群集添加新数据的速率。这可能因用例而异。例如，记录 CCTV 视频、4k 视频或医疗成像可能会更快地添加大量数据，然后再减少存储密集型信息（如金融市场数据）。此外，考虑到复制与纠删代码等 数据持久性 方法将对所需的存储介质产生重大影响。

如需有关大小调整的更多信息，请参阅《 红帽 Ceph 存储硬件选择指南》及其用于选择 OSD 硬件 的相关链接。

3.2. 考虑存储密度

集群设计的另一个重要方面包括存储密度。通常，集群应该在至少 10 个节点之间存储数据，以确保在复制、回填和恢复时具有合理的性能。如果节点失败，集群中至少有 10 个节点，则只有 10% 的数据必须移到存活的节点。如果节点数量大大减少，则必须将更高的数据百分比移到存活的节点。另外，需要设置 **full_ratio** 和 **near_full_ratio** 以适应节点故障，以确保集群可以写入数据。因此，务必要考虑存储密度。更高的存储密度不一定是一个不错的想法。

与更高的存储密度相比，另一个因素是纠删代码。当使用纠删代码编写对象并将 **node** 用作最小 CRUSH 故障域时，集群将需要尽可能多的节点作为数据和编码区块。例如：使用 **k=8, m=3** 的集群应该至少有 11 个节点，以便每个数据或编码块存储在单独的节点上。

热插拔也是重要的考虑因素。大多数现代服务器都支持驱动器热插拔。但是，一些硬件配置需要删除多个驱动器来替换驱动器。红帽建议避免此类配置，因为它们可在交换失败的磁盘时导致超过所需数量的 OSD。

3.3. 考虑网络硬件

Ceph 存储的主要优势在于，它允许独立扩展容量、IOPS 和吞吐量。云存储解决方案的一个重要方面是，由于网络延迟和其他因素，集群可能会耗尽 IOPS，或者因为集群用尽存储容量前的带宽限制而耗尽吞吐量。这意味着网络硬件配置必须支持用例，以便满足性价比目标。在考虑使用 SSD、闪存、NVMe 和其他高性能存储方法时，网络性能变得越来越重要。

Ceph 存储的另一个重要考虑因素是，它支持用于客户端和监控数据的前端或公共网络，以及用于心跳、数据复制和恢复的后端或集群网络。这意味着后端或集群网络 **始终** 需要比前端或公共网络更多的网络资源。根据数据池是否将复制或纠删代码用于数据持久性，后端或集群网络的网络要求应适当地进行量化。

最后，在安装和测试 Ceph 之前验证网络吞吐量。Ceph 中大多数与性能相关的问题通常以网络问题开头。简单的网络问题（如粒度或 Bean Cat-6 电缆）可能会导致带宽下降。至少将 10Gbe 用于前端网络。对于大型集群，请考虑将 40Gbe 用于后端或集群网络。或者，使用 LACP 模式 4 来绑定网络。另外，使用巨型帧(MTU 9000)，特别是在后端或集群网络中。

3.4. 考虑 UNINTERRUPTED POWER 供应商

由于 Ceph 写入是 atomic-all 或 no-- 并不是必须投资 Ceph OSD 节点的不可中断电源(UPS)。但是，红帽建议为 Ceph 监控节点投资 UPS。监控器使用 **leveldb**，这对同步写入延迟来说非常敏感。断电可能会导致崩溃，需要技术支持才能恢复集群的状态。

如果存储控制器使用回写缓存，Ceph OSD 可能会从使用 UPS 中受益。在这种情况下，如果控制器未及时清除回写缓存，UPS 可能会帮助防止电源中断时出现文件系统损坏。

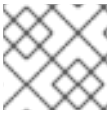
3.5. 为用例选择硬件

Ceph 存储的一个主要优点是它可以配置为支持许多用例。通常，红帽建议为特定用例配置 OSD 主机。Ceph 存储集群的三个主要用例是：

- 优化 IOPS
- 优化吞吐量
- 优化容量

由于这些用例通常具有不同的驱动器、HBA 控制器和网络要求，因此最好配置一系列相同的主机来促进所有这些使用案例（使用单一节点配置），但不建议这样做。

使用相同的主机来促进多个 CRUSH 层次结构将涉及使用逻辑而不是 CRUSH map 中的实际主机名。此外，Ansible 等部署工具需要为每个用例考虑组，而不是部署默认 **[osds]** 组中的所有 OSD。



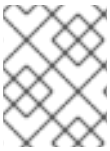
注意

通常而言，配置和管理服务于单一用例的主机更为简单，如高 IOPS、高吞吐量或高容量。

3.6. 为索引选择 MEDIA

在选择要与 Ceph 对象网关一起使用的 OSD 硬件时，--从用例的角度来看，您需要具有至少一个用于存储索引池的 OSD 节点。当 bucket 包含大量对象时，这一点尤为重要。

主机应至少有一个 SSD 或 NVMe 驱动器。在红帽实验测试中，NVMe 驱动器具有足够的性能来支持同一驱动器上的 OSD 日志和索引池，但 NVMe 驱动器的不同分区中也支持这些日志。



注意

红帽不支持索引池的 HDD 设备。如需有关支持的配置的更多信息，请参阅 *Red Hat Ceph Storage: 支持的配置* 文章。

索引条目大约为 200 字节的数据，以对象映射(omap)存储在 **leveldb** 中。虽然这一数据量很简单，但使用 Ceph 对象网关的一些用途可能会在单个 bucket 中产生数十亿或数百亿对象。通过将索引池映射到具有高性能存储介质的 CRUSH 层次结构，如果 bucket 包含大量对象，则延迟会显著提高性能。

重要

在生产集群中，典型的 OSD 节点至少有一个 SSD 或 NVMe 驱动器来存储 OSD 日志和索引池，它们将使用相同的物理驱动器时使用单独的分区或逻辑卷。

3.7. 为 MONITOR 节点选择 MEDIA

Ceph 监视器使用 **leveldb**，这对同步写入延迟而言非常敏感。红帽强烈建议使用 SSD 来存储监控数据。确保所选 SSD 具有足够的连续写入和吞吐量特征。

第 4 章 配置集群

生产集群的初始配置与配置概念验证系统相同。唯一的因素区别在于，初始部署将使用生产级硬件。首先，按照《红帽 Ceph 存储 4 安装指南》中的安装红帽 Ceph 存储要求 章节，并为每个节点执行相应的步骤。以下小节提供有关生产集群的额外指导。

4.1. 命名主机

在命名主机时，请考虑它们的用例和性能配置文件。例如，如果主机存储客户端数据，请考虑根据其硬件配置和性能配置文件对其进行命名。例如：

- **data-ssd-1**, **data-ssd-2**
- **hot-storage-1**, **hot-storage-2**
- **sata-1**, **sata-2**
- **sas-ssd-1**, **sas-ssd-2**

借助命名规则，可以更轻松地管理集群，并在硬件问题出现时进行故障排除。

如果主机包含用于多个用例的硬件-例如，该主机包含用于数据的 SSD、带有 SSD 的 SAS 驱动器和 SATA 驱动器以及带有冷存储日志的 SATA 驱动器，为主机选择通用名称。例如：

- **osd-node-1 osd-node-2**

在使用 CRUSH 层次结构中的逻辑主机名时，可以根据需要在 CRUSH 层次结构中扩展通用主机名。例如：

- **osd-node-1-ssd osd-node-1-sata osd-node-1-sas-ssd osd-node-1-bucket-index**
- **osd-node-2-ssd osd-node-2-sata osd-node-2-sas-ssd osd-node-2-bucket-index**

如需了解更多详细信息，请参阅 CRUSH map 中使用逻辑主机名。

4.2. 调整内核

生产集群可从调优操作系统（特别是限值和内存分配）中受益。确保为集群中的所有节点设置了调整。请联系红帽支持以获取其他指导。

4.2.1. 为 OSD 保留可用内存

为了帮助防止 OSD 内存分配请求期间与内存相关的错误，请在 **ceph-ansible** 节点上的 **group_vars/all.yml** 设置 **os_tuning_params** 选项。这个选项指定要保留的物理内存量。推荐的设置基于系统 RAM 量。例如：

- 对于 64GB RAM，请保留 1GB。

```
vm.min_free_kbytes = 1048576
```

- 对于 128GB RAM，请保留 2GB。

```
vm.min_free_kbytes = 2097152
```

- 对于 256GB RAM，请保留 3GB。

```
vm.min_free_kbytes = 3145728
```

4.2.2. 增加文件描述符

如果 Ceph 对象网关缺少文件描述符，它可能会挂起。修改 Ceph 对象网关节点上的 **/etc/security/limits.conf**，以增加 Ceph 对象网关的文件描述符。例如：

```
ceph      soft    nofile     unlimited
```

4.2.3. 在大型集群中调整 ulimit

对于将在大型集群上运行 Ceph 管理员命令的系统管理员，例如 1024 个 OSD 或更多 OSD，在每个节点上运行具有以下内容的管理员命令的节点上创建一个 **/etc/security/limits.d/50-ceph.conf** 文件：

```
<username>      soft    nproc     unlimited
```

使用将运行 Ceph 管理员命令的非 root 帐户的名称替换 **<username>**。

注意

默认情况下，root 用户的 ulimit 在 Red Hat Enterprise Linux 上已设置为"无限"。

4.3. 配置 ANSIBLE 组

此过程仅与使用 Ansible 部署 Ceph 相关。**ceph-ansible** 软件包已配置了默认的 **osds** 组。如果集群只有一个用例和存储策略，请按照《Red Hat Ceph Storage 安装指南》的"安装 Red Hat Ceph Storage Cluster "一节中记录的步骤进行操作。如果集群支持多个用例和存储策略，请为每个集群创建一个组。每个用例都应将 **/usr/share/ceph-ansible/group_vars/osd.sample** 复制到为组名称命名的文件中。例如，如果存储集群具有 IOPS 优化、吞吐量优化和容量优化的用例，请为每个用例创建单独的文件来代表组：

```
cd /usr/share/ceph-ansible/group_vars/
cp osds.sample osds-iops
cp osds.sample osds-throughput
cp osds.sample osds-capacity
```

然后，根据用例配置每个文件。

配置组变量文件后，编辑 **site.yml** 文件以确保其包含每个新组。例如：

```
- hosts: osds-iops
  gather_facts: false
  become: True
  roles:
  - ceph-osd

- hosts: osds-throughput
  gather_facts: false
  become: True
  roles:
  - ceph-osd
```

```
- hosts: osds-capacity
  gather_facts: false
  become: True
  roles:
  - ceph-osd
```

最后，在 **/etc/ansible/hosts** 文件中，将 OSD 节点放在对应的组名下。例如：

```
[osds-iops]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"

[osds-throughput]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"

[osds-capacity]
<ceph-host-name> devices="[ '<device_1>', '<device_2>' ]"
```

4.4. 配置 CEPH

通常，管理员应当使用 **/usr/share/ceph-ansible/group_vars** 目录中找到的 Ceph Ansible 配置文件，在初始部署前配置红帽 Ceph 存储集群。

如安装 Red Hat Ceph Storage 安装指南中的安装 Red Hat Ceph Storage Cluster 部分所述：

- 对于 monitor，请从 **sample.mons.yml** 文件创建一个 **mons.yml** 文件。
- 对于 OSD，请从 **sample.osds.yml** 文件创建 **osds.yml** 文件。
- 对于集群，从 **sample.all.yml** 文件创建 **all.yml** 文件。

按照《 安装指南 》中的指示修改设置。

另请参阅 安装 Ceph 对象网关，并从 **sample.rgws.yml** 创建 **rgws.yml** 文件。

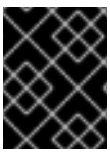
备注

应用文件中的设置可能优先于 **ceph_conf_overrides** 中的设置。

要在 **mons.yml**、**osds.yml** 或 **rgws.yml** 文件中没有对应的值配置设置，请在 **all.yml** 文件的 **ceph_conf_overrides** 部分添加配置设置。例如：

```
ceph_conf_overrides:
  global:
    osd_pool_default_pg_num: <number>
```

有关 配置文件部分的详情，请参阅配置文件结构。



重要

在 Ansible 配置文件中指定 Ceph 配置设置和 Ceph 配置文件中呈现方式之间存在语法差异。

在 RHCS 版本 3.1 及更早版本中，Ceph 配置文件使用 **ini** 风格概念。Ceph 配置文件中的 **[global]** 等部分应指定为 **global:**，并在它们自己的行中缩进。也可以指定特定守护进程实例的配置部分。例如，将

osd.1: 放置到 **all.yml** 文件的 **ceph_conf_overrides** 部分中，将在 Ceph 配置文件中呈现为 **[osd.1]**，此部分下的设置仅适用于 **osd.1**。

Ceph 配置设置 SHOULD 包含短划线(**-**)或下划线(_)而不是空格，并且应使用冒号(**:**)而非等号(**=**)终止。

在部署 Ceph 集群前，请考虑以下配置设置：在设置 Ceph 配置设置时，红帽建议在 **ceph-ansible** 配置文件中设置值，这将自动生成 Ceph 配置文件。

4.4.1. 设置日志大小

设置 Ceph 集群的日志大小。Ansible 等配置工具可能具有默认值。通常，日志大小应该找到同步间隔的产品，以及磁盘和网络吞吐量较慢的产品，并将产品乘以二(2)。

详情请参阅《红帽 Ceph 存储 4 配置指南》中的 日志设置 部分。

4.4.2. 调整回填和恢复设置

I/O 受到回填和恢复操作的负面影响，导致性能低下和最终用户不满意。要帮助满足集群扩展或恢复期间的 I/O 需求，请在 Ceph 配置文件中设置以下选项和值：

```
[osd]
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

4.4.3. 调整 cluster map Size

对于红帽 Ceph 存储版本 2 和更早版本，当集群有数千个 OSD 时，请下载 cluster map 并检查其文件大小。默认情况下，**ceph-osd** 守护进程会缓存 500 以前 osdmaps。即使使用 deduplication，映射也可能消耗每个守护进程的大量内存。在 Ceph 配置文件中调优缓存大小可能有助于显著减少内存消耗。例如：

```
[global]
osd_map_message_max=10

[osd]
osd_map_cache_size=20
osd_map_max_advance=10
osd_map_share_max_epochs=10
osd_pg_epoch_persisted_max_stale=10
```

对于 Red Hat Ceph Storage 版本 3 及更新的版本，**ceph-manager** 守护进程会处理 PG 查询，因此集群映射不会影响性能。

4.4.4. 调整清理

默认情况下，Ceph 每周执行轻型清理和深度清理。轻型清理检查对象大小和校验和，以确保 PG 存储相同的对象数据。随着时间的推移，磁盘扇区的对象大小和校验和可能会变得不良。深度清理检查对象的副本内容，以确保实际内容相同。就此而言，深度清理可确保数据的完整性，但该过程会对集群施加 I/O 后果。**fsck**甚至轻型清理也会影响 I/O。

默认设置允许 Ceph OSD 在内向期间启动清理，如峰值操作时间或负载过重的期间。当清理操作与最终用户操作冲突时，最终用户可能会遇到延迟和性能不佳的情况。

为了防止最终用户性能下降，Ceph 提供了多个清理设置，可将清理限制为负载较低或非高峰时段的期间。详情请参阅《 *红帽 Ceph 存储配置指南* 》中的 *清理 OSD* 部分。

如果集群在一天中出现高负载，当晚晚出现低负载时，请考虑将清理限制为夜间小时。例如：

```
[osd]
osd_scrub_begin_hour = 23   #23:01H, or 10:01PM.
osd_scrub_end_hour = 6      #06:01H or 6:01AM.
```

如果时间限制不是确定清理计划的有效方法，请考虑使用 **osd_scrub_load_threshold**。默认值为 **0.5**，但可以根据低负载条件进行修改。例如：

```
[osd]
osd_scrub_load_threshold = 0.25
```

4.4.5. 增加 objecter_inflight_ops

在 RHCS 3.0 及更早的版本中，请考虑将 **objecter_inflight_ops** 增加到版本 3.1 及更新的版本的默认大小，以提高可扩展性。

```
objecter_inflight_ops = 24576
```

4.4.6. 增加 rgw_thread_pool_size

在 RHCS 3.0 及更早的版本中，请考虑将 **rgw_thread_pool_size** 增加到版本 3.1 及更新的版本的默认大小，以提高可扩展性。例如：

```
rgw_thread_pool_size = 512
```

4.4.7. 调整 Garbage Collection 设置

Ceph 对象网关为新的和覆盖的对象立即分配存储。此外，多部分上传的部分也消耗了一些存储。

从 bucket 索引中删除对象后，Ceph 对象网关会清除用于已删除对象的存储空间。类似地，Ceph 对象网关将在多部分上传完成后删除与多部分上传相关的数据，或者上传未激活或未能完成时可配置的时间。从 Red Hat Ceph Storage 集群清除已删除对象数据的过程称为垃圾回收(GC)。

可使用以下命令查看等待垃圾回收的对象：

```
radosgw-admin gc list
```

垃圾回收是一种后台活动，它会持续或低负载期间执行，具体取决于存储管理员如何配置 Ceph 对象网关。默认情况下，Ceph 对象网关持续执行垃圾收集操作。由于垃圾回收操作是 Ceph 对象网关的常规功能，特别是对象删除操作，因此大多数时间都存在符合垃圾回收条件的对象。

某些工作负载可临时或永久超过垃圾收集活动的速度。对于删除密集型工作负载而言，这尤其适用，其中很多对象都会在短时间内存储，然后将其删除。对于这些类型的工作负载，存储管理员可以使用以下配置参数，相对于其他操作增加垃圾回收操作的优先级：

- **rgw_gc_obj_min_wait** 配置选项在清除已删除对象数据前等待最短时间（以秒为单位）。默认值为 2 小时或 7200 秒。对象不会立即清除，因为客户端可能会读取对象。在删除大量工作负载时，此设置可能会消耗过多的存储，或者有大量已删除的对象可以清除。红帽建议不要在 30 分

钟内设置这个值，或设置 1800 秒。

- **rgw_gc_processor_period** 配置选项是垃圾回收周期运行时间。也就是说，连续运行垃圾回收线程之间的时间长度。如果垃圾回收的时间超过这一时间段，Ceph 对象网关不会在再次运行垃圾回收循环前等待。

- **rgw_gc_max_concurrent_io** 配置选项指定网关垃圾回收线程在清除已删除数据时使用的最大并发 IO 操作数。在删除重度工作负载时，请考虑将此设置增加到更多并发 IO 操作。

- **rgw_gc_max_trim_chunk** 配置选项指定要在单个操作中从垃圾收集器日志中移除的最大密钥数量。在删除重量操作时，请考虑增加密钥的最大数量，以便在每次垃圾收集操作期间清除更多对象。

从 Red Hat Ceph Storage 4.1 开始，从垃圾回收日志中卸载索引对象的 OMAP 有助于降低垃圾回收活动对存储集群的性能影响。在 Ceph 对象网关中添加了一些新配置参数来调优垃圾回收队列，如下所示：

- **rgw_gc_max_deferred_entries_size** 配置选项在垃圾回收队列中设置延迟条目的最大大小。

- **rgw_gc_max_queue_size** 配置选项设置用于垃圾回收的最大队列大小。这个值不应该大于 **osd_max_object_size** 减去 1 KB。**rgw_gc_max_deferred_entries_size**

- **rgw_gc_max_deferred** 配置选项设置保存在垃圾回收队列中的最大延迟条目数。

注意

这些垃圾回收配置参数适用于红帽 Ceph 存储 4 及更高版本。

注意

在测试中，存储群集使用均匀均衡的删除写入工作负载（如 50% 删除和 50% 的写入操作），该存储群集占用了 11 小时。这是因为 Ceph 对象网关垃圾回收无法与删除操作同步。如果发生这种情况，集群状态将切换到 **HEALTH_ERR** 状态。对并行垃圾回收可调项的主动设置会显著延迟存储群集的测试，可能对很多工作负载有用。典型的实际存储集群工作负载可能无法导致存储集群主要因为垃圾回收而填满。

管理员也可以在部署后在运行时修改 Ceph 配置设置。详情请参阅 在运行时设置特定配置集。

第 5 章 部署 CEPH

完成先决条件和初始调优后，请考虑部署 Ceph 群集。在部署生产集群时，红帽建议设置初始监控集群和足够的 OSD 节点，以达到 **active + clean** 状态。详情请参阅《 红帽 Ceph 存储 4 安装指南》中的安装红帽 Ceph 存储集群 一节。

然后，在管理节点上安装 Ceph CLI 客户端。详情请参阅《红帽 Ceph 存储 4 安装指南》中的安装 Ceph 命令行界面 一节。

初始群集运行后，请考虑将下列部分中的设置添加到 Ceph 配置文件：

第 6 章 扩展集群

初始集群运行并处于 **active+clean** 状态后，在集群中添加额外的 OSD 节点和 Ceph 对象网关节点。 将调优内核中详述的步骤 应用到每个节点。如需了解有关 添加节点的详细信息，请参阅红帽 Ceph 存储 4 管理指南中的添加和删除 OSD 节点一节。

对于添加到集群中的每个 OSD 节点，为要存储客户端数据的节点中的每个驱动器添加 OSD 到集群中。如需了解更多详细信息，请参阅《红帽 Ceph 存储 4 管理指南》中的 添加 OSD 部分。使用 Ansible 添加 OSD 节点时，请参阅配置 Ansible 组，并在集群支持多个用例时将 OSD 节点添加到适当的组中。

对于每个 Ceph 对象网关节点，安装网关实例。详情请参阅《红帽 Ceph 存储 4 安装指南》中的安装 Ceph 对象网关 章节。

集群返回到 **active+clean** 状态后，删除任何 覆盖 并继续 开发存储策略。

注意

从 开发 CRUSH 层次结构开始，将回顾通过命令行界面添加 OSD 的第 3 步 和 步骤 10。

第 7 章 开发存储策略

为生产设置 Ceph 存储集群和 Ceph 对象网关的一个更具挑战性的方面是定义有效的存储策略。存储策略包括以下因素：

- 开发 CRUSH 层次结构
- 创建 CRUSH Roots
- 在 CRUSH map 中使用逻辑主机名
- 创建 CRUSH 规则
- 创建 Root 池
- 创建 Realm
- 创建服务池
- 创建数据放置策略

有关存储策略和命令行使用情况的一般指导，请参阅红帽 Ceph 存储 4 存储策略指南。

7.1. 开发 CRUSH 层次结构

在部署 Ceph 群集和对象网关时，对象网关通常具有默认的 zone group 和 zone。Ceph 存储群集将具有默认的池，后者使用 CRUSH 层次结构和默认 CRUSH 规则的 CRUSH map。

重要

默认 **rbd** 池可能使用默认的 CRUSH 规则。如果 Ceph 客户端已使用它们存储客户端数据，请不要 **删除** 默认规则或层次结构。

如需有关 CRUSH 层次结构的常规详细信息，请参见《存储策略指南》中的 CRUSH 管理 小节。

生产网关通常使用自定义 域、zone group 和 zone，具体取决于网关的使用和地理位置。此外，Ceph 群集将具有具有多个 CRUSH 层次结构的 CRUSH map。

- **服务池：** 至少一个 CRUSH 层次结构将用于服务池，并且可能用于数据。服务池包括 **.rgw.root** 和与区关联的服务池。服务池通常位于单个 CRUSH 层次结构下，并使用复制来实现数据持久性。数据池也可能使用 CRUSH 层次结构，但池通常配置有纠删代码以实现数据持久性。
- **索引：** 至少有一个 CRUSH 层次结构 **SHOULD** 用于索引池，其中 CRUSH 层次结构映射到 SSD 或 NVMe 驱动器等高性能介质。bucket 索引可能会成为性能瓶颈。强烈建议在此 CRUSH 层次结构中使用 SSD 或 NVMe 驱动器。要经济化，请在用于 OSD 日志的 SSD 或 NVMe 驱动器上创建索引分区。另外，索引应该配置有存储桶分片。详情请参阅 创建索引池 和支持链接。
- **放置池：** 每个放置目标的放置池包括 bucket 索引、数据存储桶和 bucket 额外内容。这些池可能属于单独的 CRUSH 层次结构。由于 Ceph 对象网关可以支持多种存储策略，存储策略的存储桶池可能会与不同的 CRUSH 层次结构关联，分别反映不同用例，如 IOPS 优化、吞吐量优化和容量优化。bucket 索引池 **SHOULD** 使用自己的 CRUSH 层次结构将 bucket 索引池映射到更高性能存储介质，如 SSD 或 NVMe 驱动器。

7.1.1. 创建 CRUSH Roots

从管理节点上的命令行，为各个 CRUSH 层次结构在 CRUSH map 中创建 CRUSH roots。**必须** 至少有一个 CRUSH 层次结构，用于可能也提供数据存储池的服务池。**SHOULD** 至少有一个 CRUSH 层次结构用于 bucket 索引池，映射到 SSD 或 NVMe 驱动器等高性能存储介质。

如需有关 CRUSH 层次结构的详细信息，请参见《*红帽 Ceph 存储策略指南*》中的 *CRUSH 层次结构章节*。

若要手动编辑 CRUSH map，*请参阅红帽 Ceph 存储策略指南 4 中的编辑 CRUSH map 部分*。

在以下示例中，名为 **data0**、**data1** 和 **data2** 的主机在 CRUSH 映射中使用扩展逻辑名称，如 **data0-sas-ssd**、**data0-index** 等，因为有多个 CRUSH 层次结构指向相同的物理主机。

典型的 CRUSH root 可能代表具有 SAS 驱动器和 SSD 的节点（用于日志）。例如：

```
##
# SAS-SSD ROOT DECLARATION
##

root sas-ssd {
  id -1    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-sas-ssd weight 4.000
  item data1-sas-ssd weight 4.000
  item data0-sas-ssd weight 4.000
}
```

用于 bucket 的 CRUSH root 索引 **SHOULD** 代表高性能介质，如 SSD 或 NVMe 驱动器。考虑在存储 OSD 日志的 SSD 或 NVMe 介质上创建分区。例如：

```
##
# INDEX ROOT DECLARATION
##

root index {
  id -2    # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item data2-index weight 1.000
  item data1-index weight 1.000
  item data0-index weight 1.000
}
```

7.1.2. 在 CRUSH map 中使用逻辑主机名

在 RHCS 3 及更高版本中，CRUSH 支持存储设备"类"的概念，此概念在 RHCS 2 及早期版本中不受支持。在 RHCS 3 集群中，包含多类存储设备的主机或节点，如 NVMe、SSD 或 HDD，使用单一 CRUSH 层次结构和设备类别来区分不同类型的存储设备。这无需使用逻辑主机名。在 RHCS 2 和更早的版本中，使用多个 CRUSH 层次结构，分别对应于每类设备，以及逻辑主机名，以区分 CRUSH 层次结构中的主机或节点。

在 CRUSH map 中，主机名必须是唯一的，并且仅使用一次。当主机服务多个 CRUSH 层次结构和使用案例时，CRUSH map 可以使用逻辑主机名而不是实际的主机名，以确保主机名仅使用一次。例如，节点可

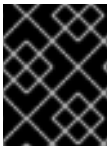
能有多个类型的驱动器，如 SSD、SAS 驱动器和 SSD 日志，以及带有并置日志的 SATA 驱动器。要在 RHCS 2 和更早的版本中为同一主机创建多个 CRUSH 层次结构，这些层次结构将需要使用逻辑主机名代替实际主机名，使得存储桶名称在 CRUSH 层次结构中是唯一的。例如，如果主机名为 **data2**，CRUSH 层次结构可能会使用逻辑名称，如 **data2-sas-ssd** 和 **data2-index**。例如：

```
host data2-sas-ssd {
  id -11   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.0 weight 1.000
  item osd.1 weight 1.000
  item osd.2 weight 1.000
  item osd.3 weight 1.000
}
```

在示例中，主机 **data2** 使用逻辑名称 **data2-sas-ssd** 将带有 SSD 上日志的 SAS 驱动器映射到一个层次结构中。for 示例中的 **osd.0** 到 **osd.3** 的 OSD ID 代表使用高吞吐量硬件配置中的 SSD 日志的 SAS 驱动器。以下示例中的 OSD ID 与 OSD ID 不同。

在以下示例中，主机 **data2** 使用逻辑名称 **data2-index** 将存储桶索引的 SSD 驱动器映射到第二个层次结构中。以下示例中的 OSD ID **osd.4** 代表 SSD 驱动器或其他高速存储介质，专门用于存储桶索引池。

```
host data2-index {
  id -21   # do not change unnecessarily
  # weight 0.000
  alg straw
  hash 0  # rjenkins1
  item osd.4 weight 1.000
}
```



重要

在使用逻辑主机名时，请确保 Ceph 配置文件中存在下列设置之一，以防止 OSD 启动脚本在启动时使用实际主机名，因而无法在 CRUSH map 中查找数据：

当 CRUSH map 使用逻辑主机名时，如示例中所示，OSD 启动脚本会阻止 OSD 启动脚本在初始化时根据其实际主机名识别主机。在 Ceph 配置文件的 **[global]** 部分，添加以下设置：

```
osd_crush_update_on_start = false
```

另一种定义逻辑主机名的方法是在 Ceph 配置文件的 **[osd.<ID>]** 部分中定义 CRUSH map 的位置。这将覆盖 OSD 启动脚本定义的任何位置。在示例中，条目可能类似如下：

```
[osd.0]
osd crush location = "host=data2-sas-ssd"

[osd.1]
osd crush location = "host=data2-sas-ssd"

[osd.2]
osd crush location = "host=data2-sas-ssd"

[osd.3]
```

```
osd crush location = "host=data2-sas-ssd"

[osd.4]
osd crush location = "host=data2-index"
```

重要

如果 CRUSH map 在重新启动时使用逻辑主机名而不是实际主机名时，Ceph 存储群集将假设 OSD map 到实际主机名，并且实际的主机名不会在 CRUSH map 中找到，而 Ceph 存储群集客户端将无法找到 OSD 及其数据，则其中一种建议方式不予以使用。

7.1.3. 创建 CRUSH 规则

与默认的 CRUSH 层次结构一样，CRUSH map 也包含默认的 CRUSH 规则。

注意

默认 **rbd** 池可能会使用这个规则。如果其他池已使用它存储客户数据，请不要删除默认规则。

有关 CRUSH 规则的一般详情，请参见《红帽 Ceph 存储 4 存储策略指南》中的 CRUSH 规则部分。若要手动编辑 CRUSH map，请参阅红帽 Ceph 存储 4 存储策略指南中的编辑 CRUSH map 部分。

对于每一 CRUSH 层次结构，创建一个 CRUSH 规则。下例演示了 CRUSH 层次结构的规则，该层次结构将存储服务池，包括 **.rgw.root**。在本例中，根 **sas-ssd** 用作 CRUSH 主层次结构。它使用名称 **rgw-service** 来区分其自身与默认规则。**step take sas-ssd** 行告知池使用创建 CRUSH Root 中创建的 **sas-ssd** root， 其子存储桶包含带有 SAS 驱动器的 OSD 和高性能存储介质，如 SSD 或 NVMe 驱动器，用于高吞吐量硬件配置中的日志。**step chooseleaf** 的 **type rack** 部分是故障域。在以下示例中，这是一个机架。

```
##
# SERVICE RULE DECLARATION
##

rule rgw-service {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type rack
 step emit
}
```

注意

在示例中，如果数据被复制三次，集群中的至少应该有三个机架，其中包含相似数量的 OSD 节点。

提示

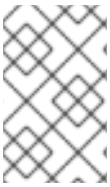
type replicated 设置 **不适用于数据** 持久性、副本数或纠删代码。只支持 **replicated**。

下例演示了将存储数据池的 CRUSH 层次结构的规则。在本例中，根 **sas-ssd** 用作主要 CRUSH 层次结

构- 与服务规则相同的 CRUSH 层次结构。它使用 **rgw-throughput** 与默认规则和 **rgw-service** 区分。**step take sas-ssd** 行告知池使用创建 CRUSH Roots 中创建 的 **sas-ssd** root，其子存储桶包含具有 SAS 驱动器的 OSD 和高性能存储介质，如 SSD 或 NVMe 驱动器。**step chooseleaf** 的 **type host** 部分是故障域。在以下示例中，这是 主机。注意该规则使用相同的 CRUSH 层次结构，但使用了不同的故障 realm。

```
##
# THROUGHPUT RULE DECLARATION
##

rule rgw-throughput {
 type replicated
 min_size 1
 max_size 10
 step take sas-ssd
 step chooseleaf firstn 0 type host
 step emit
}
```



注意

在示例中，如果池将纠删代码与更多的数据进行纠删代码，且编码区块数超过默认值，则集群中的机架应至少包含数量相似的 OSD 节点，以便于纠删代码区块。对于较小的集群，这可能不实际，因此示例中使用 **host** 作为 CRUSH 故障域。

下例演示了 CRUSH 层次结构的规则，该规则将存储索引池。在本例中，根 **index** 用作 CRUSH 主层次结构。它使用 **rgw-index** 与 **rgw-service** 和 **rgw-throughput** 区分。**step take index** 行告知池使用创建 CRUSH Root 中创建的 **index** root，其子 存储桶包含高性能存储介质，如 SSD 或 NVMe 驱动器或者 SSD 或 NVMe 驱动器上也存储 OSD 日志的分区。**step chooseleaf** 的 **type rack** 部分是故障域。在以下示例中，这是一个机架。

```
##
# INDEX RULE DECLARATION
##

rule rgw-index {
 type replicated
 min_size 1
 max_size 10
 step take index
 step chooseleaf firstn 0 type rack
 step emit
}
```

7.2. 创建 ROOT 池

Ceph 对象网关配置存储在名为 **.rgw.root** 的池中，包括 realm、zone group 和 zone。按照惯例，其名称不会以区域名称开头。

.rgw.root

如果 Ceph 存储集群正在运行，请使用新规则创建一个 **.rgw.root** 池。如需有关 PG 数量的详细信息，请参阅《存储策略指南》中的每个池 PG(PG)和 放置组 一章。有关 创建池 的详情，请参阅存储策略指南中的创建池部分。

在本例中，池将使用 **replicated** 而不是 **erasure** 作为 数据的持久性。例如：

```
# ceph osd pool create .rgw.root 32 32 replicated sas-ssd
```

注意

对于服务池（包括 **.rgw.root** ），Ceph 放置组(PG)的建议 PG 数显著 低于每个 OSD 的目标 PG。此外，还要确保在计算器的第 3 步中设置 OSD 数量。

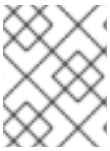
创建此池后，Ceph 对象网关就能将其配置数据存储在池中。

7.3. 创建 REALM

支持 Ceph 对象网关的 Ceph 存储池应用到 zone group 中的 zone。默认情况下，Ceph 对象网关将定义默认的 zone group 和 zone。

对于 master zone group 和 zone，红帽建议创建新的 realm、zone group 和 zone。然后，删除默认区域及其池（如果已生成）。使用配置主区 作为最佳实践，因为这为 多站点 操作配置集群。

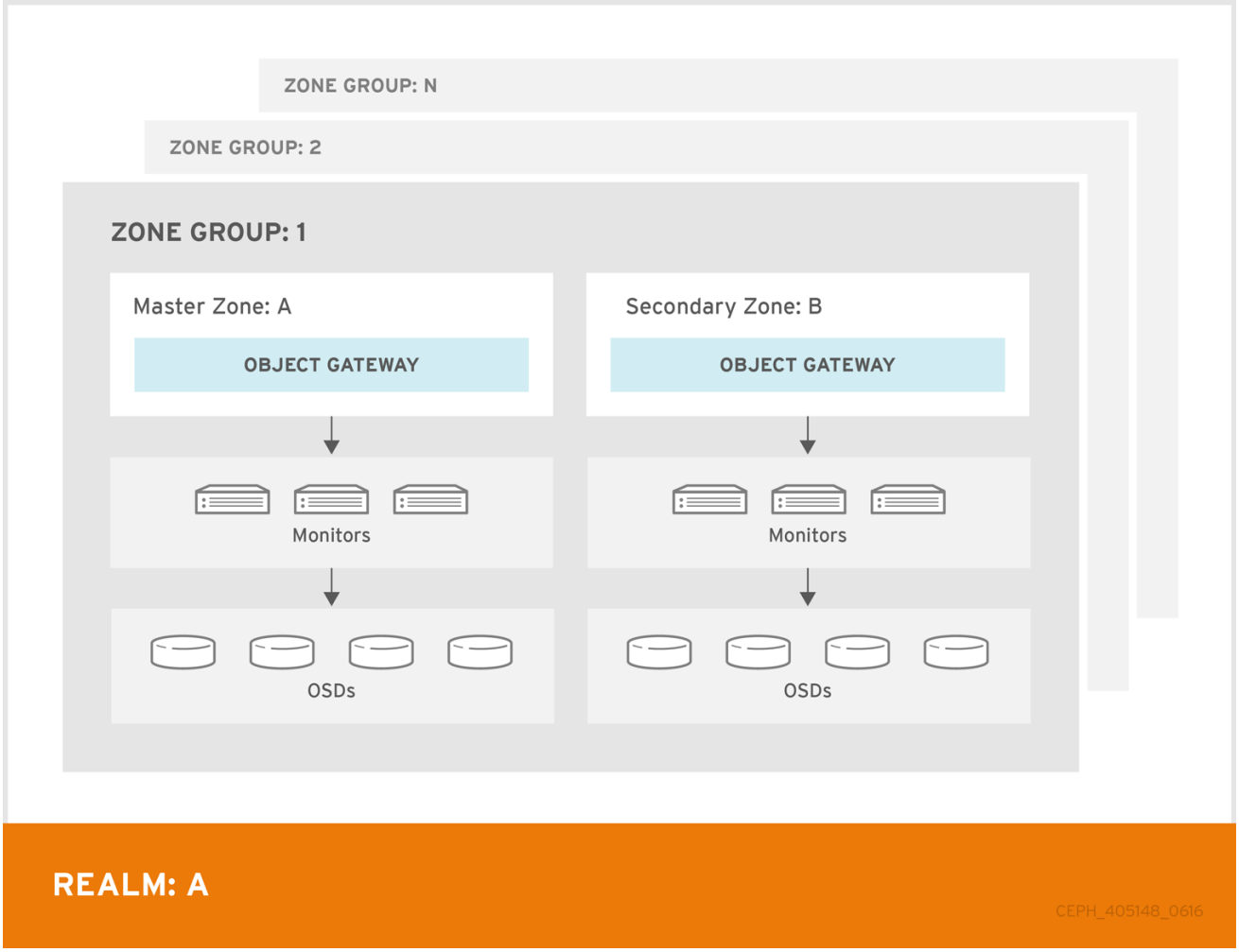
1. 创建 realm。如需了解更多详细信息，请参阅 Realms。
2. 创建 master zone group。有关 zone group 的详情，请参阅 zone group。
3. 创建 master zone。有关区域的详情，请参阅 区域。
4. 删除 default zone group 和 zone 。如果您已创建 **默认** 池，MAY 删除它们，并且不会存储客户端数据。**不要** 删除 **.rgw.root** 池。
5. 创建系统用户.
6. 更新 period。
7. 更新 Ceph 配置文件。



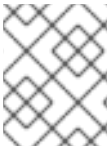
注意

此流程省略了启动网关的步骤，因为网关可能会手动创建池。若要指定特定的 CRUSH 规则和数据持久性方法，可以手动创建池。

通过设置新的 realm、zone group 和 zone，集群现在已准备好扩展到多站点集群，其中 zone group 中有多个 zone。这意味着可以扩展和配置集群，以便进行故障转移和灾难恢复。如需了解更多详细信息，请参阅使用多站点扩展 集群。



在红帽 Ceph 存储 2 中，多站点配置默认为主动主动。在部署多站点集群时，区域及其底层 Ceph 存储集群可能位于不同的地理区域中。由于每个区域都有同一命名空间中每个对象的深度副本，因此用户可从物理上最接近的区域访问副本，从而减少延迟。但是，如果辅助区域仅用于故障转移和灾难恢复，则集群可以配置为主动 - 被动模式。



注意

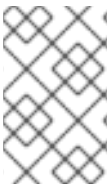
支持使用带有多个 zone 的 zone group。使用多个 zone group 只是一个技术预览，在生产环境中不支持。

7.4. 创建服务池

Ceph 对象网关将许多池用于各种服务功能，以及一组单独的放置池来存储 bucket 索引、数据和其他信息。

由于对池放置组的计算成本比较高，红帽通常建议 Ceph 对象网关的服务池使用比数据存储池少得多的放置组。

服务池存储与服务控制、垃圾收集、日志记录、用户信息、使用情况等相关的对象。按照惯例，这些池名称的前置为池名称的区域名称。



注意

自红帽 Ceph 存储 4.1 起，垃圾回收使用带有常规 RADOS 对象的日志池，而非 OMAP。将来，更多功能会将元数据存储到日志池。因此，强烈建议将 NVMe/SSD OSD 用于日志池。

- **.<zone-name>.rgw.control**：控制池。
- **.<zone-name>.log**：日志池包含所有 bucket/container 的日志，以及创建、读取、更新和删除等对象操作的日志。
- **.<zone-name>.rgw.buckets.index**：此池存储 buckes 的索引。
- **.<zone-name>.rgw.buckets.data**：此池存储存储桶的数据。
- **.<zone-name>.rgw.meta**: 元数据池存储 **user_keys** 和其他关键元数据。
- **.<zone-name>.meta:users.uid**：用户 ID 池包含唯一用户 ID 的映射。
- **.<zone-name>.meta:users.keys**：密钥池包含每个用户 ID 的访问密钥和密钥。
- **.<zone-name>.meta:users.email**：电子邮件池包含与用户 ID 关联的电子邮件地址。
- **.<zone-name>.meta:users.swift**：Swift 池包含用于用户 ID 的 Swift 子用户信息。

执行 Get a Zone 步骤来查看池名称。

```
# radosgw-admin zone get [--rgw-zone=<zone>]
```

当 **radosgw-admin** 创建一个区时，池名称为 **SHOULD**，并带有区名称。例如：一个名为 **us-west SHOULD** 的池名称如下所示：

```
{ "domain_root": ".rgw.root",
  "control_pool": ".us-west.rgw.control",
  "gc_pool": ".us-west.rgw.gc",
  "log_pool": ".us-west.log",
  "intent_log_pool": ".us-west.intent-log",
  "usage_log_pool": ".us-west.usage",
  "user_keys_pool": ".us-west.users.keys",
  "user_email_pool": ".us-west.users.email",
  "user_swift_pool": ".us-west.users.swift",
  "user_uid_pool": ".us-west.users.uid",
  "system_key": { "access_key": "", "secret_key": ""},
  "placement_pools": [
    {  "key": "default-placement",
       "val": { "index_pool": ".us-west.rgw.buckets.index",
                "data_pool": ".us-west.rgw.buckets",
                "data_extra_pool": ".us-west.rgw.buckets.non-ec"
                "index_type": 0
              }
    }
  ]
}
```

从 **control_pool** 开始并以 **user_uid_pool** 结尾，如果区名称前面是池名称的，则使用区名称创建池。按照前面的示例，池创建可能类似如下：

```
# ceph osd pool create .us-west.rgw.control 32 32 replicated rgw-service
...
# ceph osd pool create .us-west.users.uid 32 32 replicated rgw-service
```

从前面的示例中，**rgw-service** 规则代表带有 SSD 日志和 **rack** 作为 CRUSH 故障域的 SAS 驱动器的 CRUSH 层次结构。请参阅 创建 CRUSH Roots，以及 创建 CRUSH 规则，以用于前面的示例。

如需有关 PG 数量的详细信息，请参阅《存储策略指南》中的每个池 PG(PG)和 放置组 一章。有关 创建池 的详情，请参阅存储策略指南中的创建池部分。

注意

对于服务池，计算器的建议 PG 数显著低于每个 OSD 的目标 PG 数。确保计算器的第 3 步指定了正确的 OSD 数量。

通常，**.rgw.root** 池和服务池应使用相同的 CRUSH 层次结构，并且至少使用 **node** 与 CRUSH 规则中的故障域。与 **.rgw.root** 池一样，服务池应该使用 **replicated** 进行数据持久性，**而不是 erasure**。

7.5. 创建数据放置策略

Ceph 对象网关具有一个名为 **default-placement** 的默认存储策略。如果集群只有一个存储策略，**default-placement** 策略就足够了。此默认放置策略从 zone group 配置引用，并在 zone 配置中定义。

如需了解更多详细信息，请参阅红帽企业 Linux 的红帽 Ceph 存储 4 Ceph 对象网关指南中的存储 策略 部分。

对于支持多个使用案例的集群，如 IOPS 优化、吞吐量优化或容量优化，zone group 配置中的一组放置目标以及 zone 配置中的一组放置池代表每个存储策略。

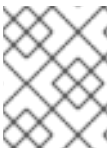
以下部分中的示例演示了如何创建存储策略并使其成为默认策略。此示例还假设默认策略将使用吞吐量优化的硬件配置文件。主题包括：

- 创建索引池
- 创建数据池
- 创建数据额外池
- 在区组中配置放置目标
- 在区中配置放置池
- 数据放置摘要

7.5.1. 创建索引池

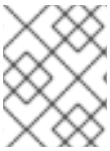
默认情况下，Ceph 对象网关将 bucket 的对象映射到索引，这使得网关客户端能够请求 bucket 中的对象列表等。虽然常见的用例可能涉及配额，即用户为每个 bucket 拥有存储桶和数量有限的对象，但 bucket 可以存储不可枚举的对象。当 bucket 存储数百万个或更多对象时，使用 SSD 或 NVMe 驱动器等高性能存储介质来存储其数据极大地提高了索引性能。另外，bucket 分片也可以显著提高性能。

如需有关 PG 数量的详细信息，请参阅《存储策略指南》中的每个池 PG(PG)和 放置组 一章。有关 创建池 的详情，请参阅存储策略指南中的创建池部分。



注意

每个池每个池的 PG 建议索引池每个池拥有较少数量的 PG；不过，PG 计数大约是服务池的 PG 数的两倍。



注意

红帽不支持索引池的 HDD 设备。如需有关支持的配置的更多信息，请参阅 *Red Hat Ceph Storage: 支持的配置* 文章。

要创建索引池，请使用池名称、PG 和 PGP 的数量、**replicated** 数据持久性方法以及规则名称来执行 **ceph osd pool create**。

重要

如果 bucket 将存储超过 100k 对象，请配置存储桶分片，以确保索引性能不会随着存储桶中对象数量的增加而降级。请参阅《Ceph 对象网关指南》的"配置 Bucket 划分"一节。另外，如果原始配置不再合适，请参阅 Ceph 对象网关指南配置和管理指南中的 Bucket Index Resharding 部分。

7.5.2. 创建数据池

数据池是 Ceph 对象网关存储特定存储策略的对象数据的位置。数据池应当包含 PG 的完整补充，而不是服务池的 PG 数量减少。数据池 **SHOULD** 考虑使用纠删代码，因为它比复制更高效，而且可以显著降低容量要求，同时保持数据持久性。

要使用纠删代码，请创建一个纠删代码 profile。如需了解更多详细信息，请参阅存储策略指南中的 Erasure Code Profiles 部分。

重要

选择正确的配置集非常重要，因为您创建池后无法更改配置集。若要修改配置文件，您必须创建一个具有不同配置文件的新池，并将对象从旧池中迁移到新池中。

默认配置是两个数据区块和一个编码区块，这意味着只能丢失一个 OSD。对于更高的弹性，请考虑大量数据和编码区块。例如，一些规模庞大的系统使用 8 个数据区块和 3 个编码区块，这允许三个 OSD 在不丢失数据的情况下出现故障。

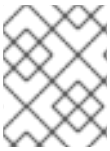
重要

每个数据和编码区块 SHOULD 至少会存储在不同的节点或主机上。对于较小的集群，这会导致在使用大量数据和编码区块时将 **rack** 用作最小 CRUSH 故障realm。因此，数据池通常使用单独的 CRUSH 层次结构 **host** 作为最小 CRUSH 故障域。红帽推荐 **host** 作为最小故障域。如果纠删代码区块存储在同一主机的 OSD 中，则发生故障日志或网卡等主机故障可能会导致数据丢失。

要创建数据池，请使用池名称、PG 和 PGP 的数量、**erasure** 数据持久性方法、纠删代码 profile 和规则名称来执行 **ceph osd pool create**。

7.5.3. 创建数据额外池

data_extra_pool 用于无法使用纠删代码的数据。例如，多部分上传允许在多个部分上传大型对象，如电影。这些部分必须首先存储而无纠删代码。纠删代码将应用到整个对象，而不是部分上传。



注意

每个池每个池的 PG 建议为 **data_extra_pool** 的每个池具有较少数量的 PG；不过，PG 数大约为服务池的 PG 数量两倍，与 bucket 索引池相同。

要创建额外的数据池，请使用池名称、PG 和 PGP 的数量、**replicated** 数据持久性方法以及规则名称来执行 **ceph osd pool create**。例如：

```
# ceph osd pool create .us-west.rgw.buckets.non-ec 64 64 replicated rgw-service
```

7.5.4. 在区组中配置放置目标

创建池后，在 zone group 中创建放置目标。要检索 zone group，请执行以下操作将 zone group 配置输出到名为 **zonegroup.json** 的文件中：

```
# radosgw-admin zonegroup get [--rgw-zonegroup=<zonegroup>] > zonegroup.json
```

文件内容类似如下：

```
{
  "id": "90b28698-e7c3-462c-a42d-4aa780d24eda",
  "name": "us",
  "api_name": "us",
  "is_master": "true",
  "endpoints": [
    "http:\/\/rgw1:80"
  ],
  "hostnames": [],
  "hostnames_s3website": [],
  "master_zone": "9248cab2-afe7-43d8-a661-a40bf316665e",
  "zones": [
     {
       "id": "9248cab2-afe7-43d8-a661-a40bf316665e",
       "name": "us-east",
       "endpoints": [
          "http:\/\/rgw1"
       ],
       "log_meta": "true",
       "log_data": "true",
       "bucket_index_max_shards": 0,
       "read_only": "false"
     },
     {
       "id": "d1024e59-7d28-49d1-8222-af101965a939",
       "name": "us-west",
       "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": "ae031368-8715-4e27-9a99-0c9468852cfe"
}
```

placement_targets 部分将列出每个存储策略。默认情况下，它将包含一个名为 **default-placement** 的放置目标。默认放置目标在 **placement_targets** 部分后马上标识。

假设名为 **throughput-optimized** 的放置目标为 **throughput-optimized** 作为默认目标，则 zone group 配置的 **placement_targets** 部分和 **default_placement** 设置应改为类似如下的内容：

```
{
  ...
  "placement_targets": [
     {
       "name": "throughput-optimized",
       "tags": []
     }
  ],
  "default_placement": "throughput-optimized",
  ...
}
```

最后，使用修改后的 **zonegroup.json** 文件中的设置设置区组配置，然后更新周期。例如：

```
# radosgw-admin zonegroup set [--rgw-zonegroup=<zonegroup>] --infile zonegroup.json
# radosgw-admin period update --commit
```

7.5.5. 在区中配置放置池

zone group 具有新的 **throughput-optimized** 放置目标后，在区配置中为 **throughput-optimized** 映射放置池。此步骤将 **default-placement** 的映射替换为使用一组 **throughput-optimized** 的放置池。

执行 Get a Zone 步骤来查看池名称。

```
# radosgw-admin zone get [--rgw-zone=<zone>] > zone.json
```

假设一个名为 **us-west** 的区域，文件内容将类似如下：

```
{ "domain_root": ".rgw.root",
  "control_pool": ".us-west.rgw.control",
  "gc_pool": ".us-west.rgw.gc",
  "log_pool": ".us-west.log",
  "intent_log_pool": ".us-west.intent-log",
  "usage_log_pool": ".us-west.usage",
  "user_keys_pool": ".us-west.users.keys",
  "user_email_pool": ".us-west.users.email",
  "user_swift_pool": ".us-west.users.swift",
  "user_uid_pool": ".us-west.users.uid",
  "system_key": { "access_key": "", "secret_key": ""},
  "placement_pools": [
   {  "key": "default-placement",
      "val": { "index_pool": ".us-west.rgw.buckets.index",
               "data_pool": ".us-west.rgw.buckets",
```

```
            "data_extra_pool": ".us-west.rgw.buckets.non-ec"
            "index_type": 0
          }
    }
  ]
}
```

区配置的 **placement_pools** 部分定义放置池集。每组放置池定义存储策略。修改 文件以删除 **default-placement** 条目，并将它替换为前面步骤中创建的池的 **throughput-optimized** 条目。例如：

```
{
...
"placement_pools": [
   {  "key": "throughput-optimized",
      "val": { "index_pool": ".us-west.rgw.buckets.index",
               "data_pool": ".us-west.rgw.buckets.throughput"}
               "data_extra_pool": ".us-west.rgw.buckets.non-ec",
               "index_type": 0
    }
  ]
}
```

最后，使用修改后的 **zone.json** 文件中的设置设置区配置，然后更新周期。例如：

```
# radosgw-admin zone set --rgw-zone={zone-name} --infile zone.json
# radosgw-admin period update --commit
```

注意

index_pool 使用 SSD 或其他高性能存储指向索引池和 CRUSH 层次结构， **data_pool** 指向具有完全补充 PG 的池，以及高吞吐量的主机总线适配器、SAS 驱动器和日志 SSD 的 CRUSH 层次结构。

7.5.6. 数据放置摘要

在处理客户端请求时，Ceph 对象网关将使用新的 **throughput-optimized** 目标作为默认存储策略。使用此流程在多站点配置中在不同 zone 和 zone group 中建立相同的目标，并根据情况替换池的区域名称。

使用此流程建立额外的存储策略。每个目标和放置池集合的命名是任意的。它可以是 **fast**、**streaming**、**cold-storage** 或者任何其他合适的名称。但是，每个集合都必须在区组的 **placement_targets** 下有一个对应的条目，并在 **default_placement** 设置中引用其中一个目标 **MUST** ；并且，区必须为每个策略配置对应的池集合。

客户端请求将始终使用默认目标，除非客户端请求指定了 **X-Storage-Policy** 和不同的目标。请参阅 为对象 网关客户端使用示例创建容器。

第 8 章 配置网关

为生产准备 Ceph 对象网关的最终步骤包括配置 Civetweb、防火墙端口、DNS 和负载平衡器。主题包括：

- 配置 Civetweb
- 配置防火墙端口
- 配置 DNS 通配符
- 配置负载均衡器

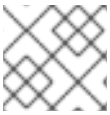
8.1. 配置 CIVETWEB

根据 安装 Ceph 对象网关期间所做的选择，Ceph 配置文件将已拥有 Ceph 对象网关的每个实例的条目，这些条目包含与 创建 Realm 相关的步骤。

从默认配置中最常见的配置更改是将默认 Ansible 配置的端口 **8080** 更改为另一个端口，如 **80**。请参阅 更改 CivetWeb 端口。

还有其他一些设置，特别是 Civetweb。详情请参阅 Civetweb 配置选项。

其他设置可能会被覆盖。详情请参阅 对象网关配置参考。



注意

有关其他用例的章节将提供将 Ceph 对象网关与第三方组件搭配使用的详细配置示例。

8.2. 配置防火墙端口

在更改 Civetweb 的默认端口时，请确保为客户端访问打开对应的端口。详情请查看 Red Hat Enterprise Linux Red Hat Ceph Storage 4 安装指南中的为 Red Hat Ceph Storage 4 配置防火墙 部分。

8.3. 配置 DNS 通配符

S3style 子域包含存储桶名称作为 CNAME 扩展名。向 DNS 添加通配符，以便于 S3 样式的子域。详情请参阅红帽 Ceph 存储 4 Ceph 对象网关指南配置和管理指南中的将 通配符添加到 DNS 部分。

8.4. 配置负载均衡器

个区域通常具有多个 Ceph 对象网关实例，以处理生产负载和维护高可用性。生产集群通常使用负载均衡器在网关实例之间分配请求。

此外，较早版本的 Civetweb 不支持 HTTPS。负载平衡器可以配置为接受 SSL 请求，终止 SSL 连接，并通过 HTTP 将请求传递到网关实例。

Ceph 存储旨在维护高可用性。因此，红帽建议使用 HAProxy 或 keepalived。详情请参阅 Ceph 对象网关《配置和管理指南》中的 HAProxy/keepalived 配置 一节。

8.5. 使用 BEAST 前端

Ceph 对象网关将 CivetWeb 和 Beast 嵌入式 HTTP 服务器作为前端提供。Beast 前端使用 **Boost.Beast**

库进行 HTTP 解析，使用 **Boost.Asio** 库进行异步网络 I/O。在 Red Hat Ceph Storage 版本 3.x 中，CivetWeb 是默认的前端，并使用 Red Hat Ceph Storage 配置文件中的 **rgw_frontends** 指定 Beast 前端。自 Red Hat Ceph Storage 版本 4.0 起，Beast 前端默认为 4.0，从 Red Hat Ceph Storage 3.x 升级会自动将 **rgw_frontends** 参数改为 Beast。

其它资源

- *Past 配置选项*

8.6. PAST 配置选项

以下 Beast 配置选项可以传递到 RADOS 网关的 Ceph 配置文件中的嵌入式 Web 服务器：每个选项都有一个默认值。如果没有指定值，则默认值为空。

选项	描述	Default（默认）
endpoint 和 **ssl_endpoint**	以 **address[:port]** 格式设置侦听地址，其中地址是以十进制格式的 IPv4 地址字符串，或者以十六进制表示法括起的 IPv6 地址。对于 **8080**，可选的端口默认为 **endpoint**，**443** 和 **ssl_endpoint** 默认为。它可多次指定，如 **endpoint=[::1] endpoint=192.168.0.100:8000** 中所示。	空
ssl_certificate	用于启用 SSL 端点的 SSL 证书文件路径。如果文件是包含多个项目的 PEM 文件，则顺序非常重要。文件必须以 RGW 服务器密钥、任何中间证书开头，最后是 CA 证书。	空
ssl_private_key	用于启用 SSL 端点的私钥文件的可选路径。如果没有提供 **ssl_certificate** 指定的文件，则会将该文件用作私钥。	空
tcp_nodelay	某些环境的性能优化.	空

使用 SSL 带有 Beast 选项的 /etc/ceph/ceph.conf 文件示例：

```
...

[client.rgw.node1]
rgw frontends = beast ssl_endpoint=192.168.0.100:443 ssl_certificate=<path to SSL certificate>
```

注意

默认情况下，Beast 前端写入一个访问日志行，记录服务器处理的所有请求到 RADOS 网关日志文件。

其它资源

- 如需更多信息，请参阅使用 Beast 前端。

第 9 章 其他用例

集群启动并运行后，需要考虑其他用例。

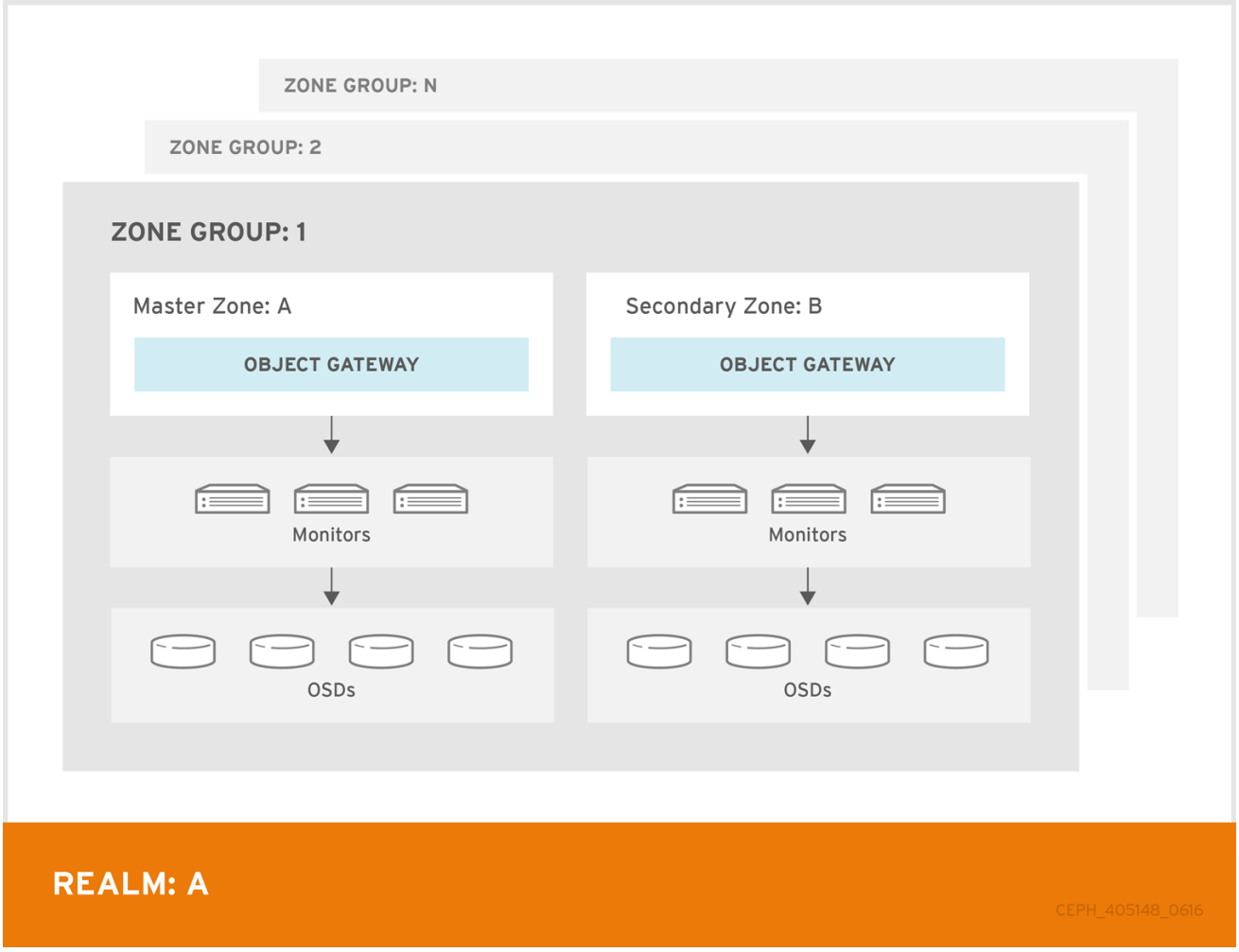
- 使用多站点扩展集群
- 使用 NFS-Ganesha 迁移数据
- 为静态 Webhosting 配置集群
- 使用 LDAP/AD 配置集群
- 将集群配置为使用 Keystone

9.1. 使用多站点扩展集群

在开发存储策略时，创建 Realm 的步骤可确保集群已配置为使用带有自己的 realm、master zone group 和 master zone 的多站点。

典型的生产集群将有一个 second zone，其自己的 Ceph 存储群集位于单独的物理位置，以便在发生灾难时充当备份。要设置二级区域，请重复本指南中的步骤。通常，second 区域应具有与 master zone 相同的硬件配置和大小调整。如需了解更多详细信息，请参阅配置第二个区域。

添加 second zone 为集群添加 Failover 和 Disaster Recovery 功能。



9.2. 使用 NFS GANESHA 迁移数据

如果 Ceph 对象网关和 Ceph 存储群集取代了基于文件系统的存储解决方案，请考虑使用 Ceph 的 NFS-Ganesha 解决方案将数据从文件系统迁移到 Ceph 对象网关。

请参阅 《Ceph 对象网关配置和管理指南》中的将命名空间导出到 NFS-Ganesha 章节。

9.3. 为静态 WEBHOSTING 配置集群

传统的 Web 主机有时涉及为每个网站设置 Web 服务器，当内容未动态变化时，传统 Web 服务器会以低效的方式使用资源。

Ceph 对象网关可以在 S3 buckets- 中托管静态网站，即不使用 PHP、servlet、数据库、nodejs 等服务器端服务的站点。这种方法比为每个站点设置具有 Web 服务器的虚拟机更经济。

如需了解更多详细信息，请参阅为静态 Web 托管配置网关。

9.4. 为 LDAP/AD 配置集群

组织为其用户和应用程序部署 Ceph 对象网关，可以选择使用轻量级目录访问协议(LDAP)或 Microsoft Active Directory(AD)来通过 Ceph 对象网关（而非创建 Ceph 对象网关）进行身份验证。

使用 LDAP/AD 表示 Ceph 对象网关可以与组织 LDAP/AD 单点登录方案集成。

详情请参阅红帽 Ceph 存储 4 的带有 LDAP/AD 的 Ceph 对象网关指南 。

9.5. 将集群配置为使用 OPENSTACK KEYSTONE

在使用 OpenStack Swift 部署 Ceph 对象网关时，可以将网关配置为使用 OpenStack Keystone 对用户进行身份验证，而不是创建 Ceph 对象网关用户。

详情请参阅 使用 Keystone 验证红帽 Ceph 存储 4 的 Ceph 对象网关用户指南 。

第 10 章 配合 LVM 使用 NVME

1. summary
   以下流程演示了在使用基于 NVMe 的 SSD 时，如何以最佳的方式部署 Ceph 以实现对象网关使用（这也适用于 SATA SSD）。日志和 bucket 索引将一起放在高速存储设备上，与将所有日志放在一个设备上相比，这可以提高性能。这个配置需要将 **osd_scenario** 设置为 **lvm**。

   提供了两个示例配置的步骤：

   - 使用一个存储桶索引有一个 NVMe 设备和至少四个 HDD： 一个 NVMe 设备
   - 使用两个存储桶索引的两个 NVMe 设备和至少四个 HDD： 两个 NVMe 设备

2. 详情
   最基本的 Ceph 设置使用 **osd_scenario** 设置 **collocated**。这会将 OSD 数据及其日志存储到一个存储设备上（它们都是"并置"）。典型的服务器配置包括 HDD 和 SSD。由于 HDD 通常大于 SSD，因此在并置配置中将 HDD 调出要选择的最大存储空间，单独将 OSD 数据和日志放在其中。不过，日志最好位于更快的 SSD 上。另一个选项是使用 **non-collocated** 的 **osd_scenario** 设置。这允许配置专用设备以用于日志，因此您可以将 OSD 数据放在 HDD 和 SSD 上的日志中。

   除了 OSD 数据和日志外，使用对象网关时，需要将 bucket 索引存储在设备上。在这种情形中，通常配置 Ceph，使 HDD 保存 OSD 数据，一个 SSD 保存日志，另一个 SSD 保存 bucket 索引。这可能会造成高度不稳定的情况，即所有日志的 SSD 变得饱和，而具有存储桶索引的 SSD 未被充分利用。

   解决方法是将 **osd_scenario** 设置为 **lvm**，并使用逻辑卷管理器(LVM)为多个目的划分单个 SSD 设备。这使得日志和 bucket 索引可以在单个设备上并排存在。最重要的是，它允许日志存在于多个 SSD 上，将日志的密集 IO 数据传输分布到多个设备上。

   用于安装 Ceph 的 **ceph-ansible** RPM 提供的普通 Ansible playbook（site.yml、osds.yml 等）不支持将一个设备用于多个目的。

   在未来，正常的 Ansible playbook 将支持将一个设备用于多个目的。同时提供了 playbook **lv-create.yml** 和 **lv-vars.yaml**，以帮助创建用于优化 SSD 使用所需的日志卷(LV)。**lv-create.yml** 运行 **site.yml** 后可以正常运行，它将使用新创建的 LV。

重要

这些步骤只适用于 FileStore 存储后端，而不是新的 BlueStore 存储后端。

10.1. 使用一个 NVME 设备

按照以下步骤，使用一个 NVMe 设备部署 Ceph 以使用对象网关。

10.1.1. 清除任何现有的 Ceph 集群

如果已经配置了 Ceph，请清除它，以便从头开始。Ansible playbook **purge-cluster.yml** 提供用于此目的。

```
$ ansible-playbook infrastructure-playbooks/purge-cluster.yml -i hosts
```

有关如何使用 **purge-cluster.yml** 的更多信息，请参阅 安装指南 中的 *Ansible 清除 Ceph 集群*。

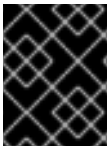
重要

清除集群可能不足以让服务器做好按照下列步骤重新部署 Ceph 的准备。Ceph 使用的存储设备上的任何文件系统、GPT、RAID 或其他签名都可能会导致问题。Run The lv-create.yml Ansible Playbook 下提供了删除任何使用 **wipefs** 的签名的说明。

10.1.2. 为常规安装配置集群

在设置任何 NVMe 和/或 LVM 注意事项之前，按常规方式配置集群，但在运行 Ansible playbook 前停止。之后，将专门针对最佳 NVMe/LVM 使用调整集群安装配置，以支持对象网关。只有在那个时候，Ansible playbook 才应当运行。

若要为普通安装配置存储集群，请参阅《 *红帽 Ceph 存储安装指南* 》。特别是，通过创建 Ansible 日志目录第 9 步完成安装 *红帽 Ceph 存储集群* 中的步骤。在第 10 步之前停止，何时运行 **ansible-playbook site.yml -i hosts**。



重要

在完成此步骤之后、为 NVMe 安装 Ceph 和 Verify Success 之前，不要运行 **ansible-playbook site.yml -i hosts**。

10.1.3. 识别 NVMe 和 HDD 设备

使用 **lsblk** 标识连接到服务器的 NVMe 和 HDD 设备。下面列出了 **lsblk** 的输出示例：

```
[root@c04-h05-6048r ~]# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda       8:0    0 465.8G  0 disk
├─sda1    8:1    0     4G  0 part
│ └─md1   9:1    0     4G  0 raid1 [SWAP]
├─sda2    8:2    0   512M  0 part
│ └─md0   9:0    0   512M  0 raid1 /boot
└─sda3    8:3    0 461.3G  0 part
  └─md2   9:2    0 461.1G  0 raid1 /
sdb       8:16   0 465.8G  0 disk
├─sdb1    8:17   0     4G  0 part
│ └─md1   9:1    0     4G  0 raid1 [SWAP]
├─sdb2    8:18   0   512M  0 part
│ └─md0   9:0    0   512M  0 raid1 /boot
└─sdb3    8:19   0 461.3G  0 part
  └─md2   9:2    0 461.1G  0 raid1 /
sdc       8:32   0   1.8T  0 disk
sdd       8:48   0   1.8T  0 disk
sde       8:64   0   1.8T  0 disk
sdf       8:80   0   1.8T  0 disk
sdg       8:96   0   1.8T  0 disk
sdh       8:112  0   1.8T  0 disk
sdi       8:128  0   1.8T  0 disk
sdj       8:144  0   1.8T  0 disk
sdk       8:160  0   1.8T  0 disk
sdl       8:176  0   1.8T  0 disk
sdm       8:192  0   1.8T  0 disk
sdn       8:208  0   1.8T  0 disk
sdo       8:224  0   1.8T  0 disk
sdp       8:240  0   1.8T  0 disk
```

```
sdq      65:0    0   1.8T  0 disk
sdr      65:16   0   1.8T  0 disk
sds      65:32   0   1.8T  0 disk
sdt      65:48   0   1.8T  0 disk
sdu      65:64   0   1.8T  0 disk
sdv      65:80   0   1.8T  0 disk
sdw      65:96   0   1.8T  0 disk
sdx      65:112  0   1.8T  0 disk
sdy      65:128  0   1.8T  0 disk
sdz      65:144  0   1.8T  0 disk
sdaa     65:160  0   1.8T  0 disk
sdab     65:176  0   1.8T  0 disk
sdac     65:192  0   1.8T  0 disk
sdad     65:208  0   1.8T  0 disk
sdae     65:224  0   1.8T  0 disk
sdaf     65:240  0   1.8T  0 disk
sdag     66:0    0   1.8T  0 disk
sdah     66:16   0   1.8T  0 disk
sdai     66:32   0   1.8T  0 disk
sdaj     66:48   0   1.8T  0 disk
sdak     66:64   0   1.8T  0 disk
sdal     66:80   0   1.8T  0 disk
nvme0n1 259:0    0 745.2G  0 disk
nvme1n1 259:1    0 745.2G  0 disk
```

在本例中，将使用以下原始块设备：

NVMe 设备

1. **/dev/nvme0n1**

HDD 设备

1. **/dev/sdc**
2. **/dev/sdd**
3. **/dev/sde**
4. **/dev/sdf**

lv_vars.yaml 文件在所选设备中配置逻辑卷创建。它在 NVMe、基于 NVMe 的 bucket 索引和基于 HDD 的 OSD 上创建日志。逻辑卷的实际创建是由 **lv-create.yml** 启动的，它显示为 **lv_vars.yaml**。

该文件一次仅应具有一个 NVMe 设备。有关将 Ceph 与两个 NVMe 设备搭配使用的最佳信息，请参见使用两个 NVMe 设备。

10.1.4. 将设备添加到 lv_vars.yaml

1. 作为 **root**，进入 **/usr/share/ceph-ansible/** 目录：

```
# cd /usr/share/ceph-ansible
```

2. 编辑该文件，使其包含以下行：

```
nvme_device: /dev/nvme0n1
hdd_devices:
  - /dev/sdc
  - /dev/sdd
  - /dev/sde
  - /dev/sdf
```

10.1.5. 运行 lv-create.yml Ansible Playbook

lv-create.yml playbook 的目的是在单个 NVMe 中为对象网关存储桶索引和日志创建逻辑卷。它使用 **osd_scenario=lvm** 进行此操作。通过自动化一些复杂的 LVM 创建和配置， **lv-create.yml** Ansible playbook 使以这种方式配置 Ceph 变得更加简单。

1. 确保存储设备是 raw
   在运行 **lv-create.yml** 在 NVMe 设备和 HDD 设备中创建逻辑卷前，请确保它们中没有文件系统、GPT、RAID 或其他签名。

   如果它们不是原始的，当运行 **lv-create.yml** 时可能会失败，并显示以下错误：

   ```
   device /dev/sdc excluded by a filter
   ```

2. wipe 存储设备签名（可选）：
   如果设备有签名，您可以使用 **wipefs** 擦除它们。

   以下是使用 **wipefs** 擦除设备的示例：

   ```
   [root@c04-h01-6048r ~]# wipefs -a /dev/sdc
   /dev/sdc: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdc: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdc: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/sdc: calling ioclt to re-read partition table: Success
   [root@c04-h01-6048r ~]# wipefs -a /dev/sdd
   /dev/sdd: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdd: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdd: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/sdd: calling ioclt to re-read partition table: Success
   [root@c04-h01-6048r ~]# wipefs -a /dev/sde
   /dev/sde: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/sde: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
   /dev/sde: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/sde: calling ioclt to re-read partition table: Success
   [root@c04-h01-6048r ~]# wipefs -a /dev/sdf
   /dev/sdf: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdf: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
   /dev/sdf: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
   /dev/sdf: calling ioclt to re-read partition table: Success
   ```

3. 运行 **lv-teardown.yml** Ansible playbook:
   总是在运行 **lv-create.yml** 前运行 **lv-teardown.yml** ：

   运行 **lv-teardown.yml** Ansible playbook:

   ```
   $ ansible-playbook infrastructure-playbooks/lv-teardown.yml -i hosts
   ```

警告

运行 **lv-teardown.yml** Ansible 脚本时请谨慎操作。它销毁数据。确保您已备份了任何重要数据。

4. 运行 **lv-create.yml** Ansible playbook:

```
$ ansible-playbook infrastructure-playbooks/lv-create.yml -i hosts
```

5. **lv-create.yml** 完成且没有错误后，继续到下一节以验证它是否正常工作。

10.1.6. 验证 LVM 配置

1. 查看 **lv-created.log**:
lv-create.yml Ansible playbook 成功完成后，配置信息将写入 **lv-created.log**。稍后，此信息将复制到 **group_vars/osds.yml**。打开 **lv-created.log** 并查找类似以下示例的信息：

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme0n1
  journal: ceph-journal-bucket-index-1-nvme0n1
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdc
  data_vg: ceph-hdd-vg-sdc
  journal: ceph-journal-sdc
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdd
  data_vg: ceph-hdd-vg-sdd
  journal: ceph-journal-sdd
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sde
  data_vg: ceph-hdd-vg-sde
  journal: ceph-journal-sde
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdf
  data_vg: ceph-hdd-vg-sdf
  journal: ceph-journal-sdf
  journal_vg: ceph-nvme-vg-nvme0n1
```

2. 查看 LVM 配置

根据一个 NVMe 设备和四个 HDD 的示例，应创建以下逻辑卷(LV)：

NVMe 上放置每个 HDD 的一个日志 LV（/dev/nvme0n1 上的四个 LV）

每个 HDD 放置一个数据 LV（每个 HDD 个 LV）

个用于 NVMe 上存储桶索引的日志 LV（位于 /dev/nvme0n1 上的一个 LV）

个用于存储桶索引的数据 LV 放置在 NVMe 上（/dev/nvme0n1 上的一个 LV）

LV 可在 **lsblk** 和 **lvscan** 输出中看到。在上图的示例中，Ceph 应当有十个 LV。粗略检查您可以计算 Ceph LV 的数量以确保至少有十个 LV，但最好确保在正确的存储设备（NVMe 与 HDD）上创建了适当的 LV。

lsblk 的输出示例如下所示：

```
[root@c04-h01-6048r ~]# lsblk
NAME                                  MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                     8:0    0 465.8G  0 disk
├─sda1                                  8:1    0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sda2                                  8:2    0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sda3                                  8:3    0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdb                                     8:16   0 465.8G  0 disk
├─sdb1                                  8:17   0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sdb2                                  8:18   0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sdb3                                  8:19   0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdc                                     8:32   0   1.8T  0 disk
└─ceph--hdd--vg--sdc-ceph--hdd--lv--sdc 253:6    0   1.8T  0 lvm
sdd                                     8:48   0   1.8T  0 disk
└─ceph--hdd--vg--sdd-ceph--hdd--lv--sdd 253:7    0   1.8T  0 lvm
sde                                     8:64   0   1.8T  0 disk
└─ceph--hdd--vg--sde-ceph--hdd--lv--sde 253:8    0   1.8T  0 lvm
sdf                                     8:80   0   1.8T  0 disk
└─ceph--hdd--vg--sdf-ceph--hdd--lv--sdf 253:9    0   1.8T  0 lvm
sdg                                     8:96   0   1.8T  0 disk
sdh                                     8:112  0   1.8T  0 disk
sdi                                     8:128  0   1.8T  0 disk
sdj                                     8:144  0   1.8T  0 disk
sdk                                     8:160  0   1.8T  0 disk
sdl                                     8:176  0   1.8T  0 disk
sdm                                     8:192  0   1.8T  0 disk
sdn                                     8:208  0   1.8T  0 disk
sdo                                     8:224  0   1.8T  0 disk
sdp                                     8:240  0   1.8T  0 disk
sdq                                     65:0   0   1.8T  0 disk
sdr                                     65:16  0   1.8T  0 disk
sds                                     65:32  0   1.8T  0 disk
sdt                                     65:48  0   1.8T  0 disk
sdu                                     65:64  0   1.8T  0 disk
sdv                                     65:80  0   1.8T  0 disk
sdw                                     65:96  0   1.8T  0 disk
sdx                                     65:112 0   1.8T  0 disk
sdy                                     65:128 0   1.8T  0 disk
sdz                                     65:144 0   1.8T  0 disk
sdaa                                    65:160 0   1.8T  0 disk
sdab                                    65:176 0   1.8T  0 disk
sdac                                    65:192 0   1.8T  0 disk
sdad                                    65:208 0   1.8T  0 disk
sdae                                    65:224 0   1.8T  0 disk
sdaf                                    65:240 0   1.8T  0 disk
```

```
sdag                                                    66:0    0   1.8T  0 disk
sdah                                                    66:16   0   1.8T  0 disk
sdai                                                   66:32   0   1.8T  0 disk
sdaj                                                   66:48   0   1.8T  0 disk
sdak                                                    66:64   0   1.8T  0 disk
sdal                                                   66:80   0   1.8T  0 disk
nvme0n1                                                  259:0    0 745.2G  0 disk
├─ceph--nvme--vg--nvme0n1-ceph--journal--bucket--index--1--nvme0n1 253:0    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdc                253:1    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdd                253:2    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sde                253:3    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdf               253:4    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme0n1-ceph--bucket--index--1              253:5    0 718.4G  0 lvm
nvme1n1                                                 259:1    0 745.2G  0 disk
```

lvscan 输出示例如下：

```
[root@c04-h01-6048r ~]# lvscan
  ACTIVE            '/dev/ceph-hdd-vg-sdf/ceph-hdd-lv-sdf' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sde/ceph-hdd-lv-sde' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdd/ceph-hdd-lv-sdd' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-bucket-index-1-nvme0n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdc' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdd' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sde' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdf' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-bucket-index-1' [<718.36 GiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdc/ceph-hdd-lv-sdc' [<1.82 TiB] inherit
```

10.1.7. 编辑 osds.yml 和 all.yml Ansible Playbook

1. 将前面提到的配置信息从 **lv-created.log** 复制到 **group_vars/osds.yml** 行下的 **lvm_volumes:** 行下。

2. 将 **osd_scenario:** 设置为 **lvm**:

```
osd_scenario: lvm
```

3. 在 **all.yml** 和 **osds.yml** 中设置 **osd_objectstore: filestore**。
osds.yml 文件应类似如下：

```
# Variables here are applicable to all host groups NOT roles

osd_objectstore: filestore
osd_scenario: lvm
lvm_volumes:
  - data: ceph-bucket-index-1
    data_vg: ceph-nvme-vg-nvme0n1
    journal: ceph-journal-bucket-index-1-nvme0n1
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdc
    data_vg: ceph-hdd-vg-sdc
```

```
    journal: ceph-journal-sdc
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdd
    data_vg: ceph-hdd-vg-sdd
    journal: ceph-journal-sdd
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sde
    data_vg: ceph-hdd-vg-sde
    journal: ceph-journal-sde
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdf
    data_vg: ceph-hdd-vg-sdf
    journal: ceph-journal-sdf
    journal_vg: ceph-nvme-vg-nvme0n1
```

10.1.8. 为 NVMe 安装 Ceph 并验证成功

将 Ceph 配置为以 LVM 最佳方式使用 NVMe 后，请安装它。

1. 运行 **site.yml** Ansible playbook 来安装 Ceph

```
$ ansible-playbook -v site.yml -i hosts
```

2. 在安装完成后验证 Ceph 是否正常运行

```
# ceph -s
```

```
# ceph osd tree
```

显示 Ceph 正常运行的 **ceph -s** 输出示例：

```
 # ceph -s
  cluster:
    id:     15d31a8c-3152-4fa2-8c4e-809b750924cd
    health: HEALTH_WARN
            Reduced data availability: 32 pgs inactive

  services:
    mon: 3 daemons, quorum b08-h03-r620,b08-h05-r620,b08-h06-r620
    mgr: b08-h03-r620(active), standbys: b08-h05-r620, b08-h06-r620
    osd: 35 osds: 35 up, 35 in

  data:
    pools:   4 pools, 32 pgs
    objects: 0 objects, 0 bytes
    usage:   0 kB used, 0 kB / 0 kB avail
    pgs:     100.000% pgs unknown
             32 unknown
```

显示 Ceph 正常运行的 **ceph osd tree** 输出示例：

```
[root@c04-h01-6048r ~]# ceph osd tree
ID  CLASS WEIGHT   TYPE NAME             STATUS REWEIGHT PRI-AFF
 -1       55.81212 root default
```

```
-15       7.97316     host c04-h01-6048r
 13   hdd  1.81799         osd.13            up  1.00000 1.00000
 20   hdd  1.81799         osd.20            up  1.00000 1.00000
 26   hdd  1.81799         osd.26            up  1.00000 1.00000
 32   hdd  1.81799         osd.32            up  1.00000 1.00000
  6   ssd  0.70119         osd.6             up  1.00000 1.00000
 -3       7.97316     host c04-h05-6048r
 12   hdd  1.81799         osd.12            up  1.00000 1.00000
 17   hdd  1.81799         osd.17            up  1.00000 1.00000
 23   hdd  1.81799         osd.23            up  1.00000 1.00000
 29   hdd  1.81799         osd.29            up  1.00000 1.00000
  2   ssd  0.70119         osd.2             up  1.00000 1.00000
-13       7.97316     host c04-h09-6048r
 11   hdd  1.81799         osd.11            up  1.00000 1.00000
 16   hdd  1.81799         osd.16            up  1.00000 1.00000
 22   hdd  1.81799         osd.22            up  1.00000 1.00000
 27   hdd  1.81799         osd.27            up  1.00000 1.00000
  4   ssd  0.70119         osd.4             up  1.00000 1.00000
 -5       7.97316     host c04-h13-6048r
 10   hdd  1.81799         osd.10            up  1.00000 1.00000
 15   hdd  1.81799         osd.15            up  1.00000 1.00000
 21   hdd  1.81799         osd.21            up  1.00000 1.00000
 28   hdd  1.81799         osd.28            up  1.00000 1.00000
  1   ssd  0.70119         osd.1             up  1.00000 1.00000
 -9       7.97316     host c04-h21-6048r
  8   hdd  1.81799         osd.8             up  1.00000 1.00000
 18   hdd  1.81799         osd.18            up  1.00000 1.00000
 25   hdd  1.81799         osd.25            up  1.00000 1.00000
 30   hdd  1.81799         osd.30            up  1.00000 1.00000
  5   ssd  0.70119         osd.5             up  1.00000 1.00000
-11       7.97316     host c04-h25-6048r
  9   hdd  1.81799         osd.9             up  1.00000 1.00000
 14   hdd  1.81799         osd.14            up  1.00000 1.00000
 33   hdd  1.81799         osd.33            up  1.00000 1.00000
 34   hdd  1.81799         osd.34            up  1.00000 1.00000
  0   ssd  0.70119         osd.0             up  1.00000 1.00000
 -7       7.97316     host c04-h29-6048r
  7   hdd  1.81799         osd.7             up  1.00000 1.00000
 19   hdd  1.81799         osd.19            up  1.00000 1.00000
 24   hdd  1.81799         osd.24            up  1.00000 1.00000
 31   hdd  1.81799         osd.31            up  1.00000 1.00000
  3   ssd  0.70119         osd.3             up  1.00000 1.00000
```

Ceph 现已设置为以一个 NVMe 设备和 LVM 优化方式用于 Ceph 对象网关。

10.2. 使用两个 NVME 设备

按照以下步骤，部署 Ceph 以使用两个 NVMe 设备的对象网关。

10.2.1. 清除任何现有的 Ceph 集群

如果已经配置了 Ceph，请清除它，以便从头开始。为此，提供了名为 **purge-cluster.yml** 的 ansible playbook 文件。

```
$ ansible-playbook purge-cluster.yml -i hosts
```

有关如何使用 **purge-cluster.yml** 的更多信息，请参阅 安装指南 中的 *Ansible 清除 Ceph 集群*。

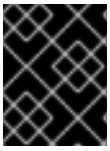
重要

清除集群可能不足以让服务器做好按照下列步骤重新部署 Ceph 的准备。Ceph 使用的存储设备上的任何文件系统、GPT、RAID 或其他签名都可能会导致问题。Run The lv-create.yml Ansible Playbook 下提供了删除任何使用 **wipefs** 的签名的说明。

10.2.2. 为常规安装配置集群

在设置任何 NVMe 和/或 LVM 注意事项之前，按常规方式配置集群，但在运行 Ansible playbook 前停止。之后，将专门针对最佳 NVMe/LVM 使用调整集群安装配置，以支持对象网关。只有在那个时候，Ansible playbook 才应当运行。

要配置集群以进行正常安装，请参阅《 安装指南 》。特别是，通过创建 Ansible 日志目录第 9 步完成安装 红帽 Ceph 存储集群 中的步骤。在运行 **ansible-playbook site.yml -i hosts** 的第 10 步前停止。



重要

在完成此步骤之后、为 NVMe 安装 Ceph 和 Verify Success 之前，不要运行 **ansible-playbook site.yml -i hosts**。

10.2.3. 识别 NVMe 和 HDD 设备

使用 **lsblk** 标识连接到服务器的 NVMe 和 HDD 设备。下面列出了 **lsblk** 的输出示例：

```
[root@c04-h09-6048r ~]# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0 465.8G  0 disk
├─sda1                      8:1    0   512M  0 part /boot
└─sda2                      8:2    0 465.3G  0 part
  ├─vg_c04--h09--6048r-lv_root 253:0    0 464.8G  0 lvm  /
  └─vg_c04--h09--6048r-lv_swap 253:1    0   512M  0 lvm  [SWAP]
sdb                         8:16   0 465.8G  0 disk
sdc                         8:32   0   1.8T  0 disk
sdd                         8:48   0   1.8T  0 disk
sde                         8:64   0   1.8T  0 disk
sdf                         8:80   0   1.8T  0 disk
sdg                         8:96   0   1.8T  0 disk
sdh                         8:112  0   1.8T  0 disk
sdi                         8:128  0   1.8T  0 disk
sdj                         8:144  0   1.8T  0 disk
sdk                         8:160  0   1.8T  0 disk
sdl                         8:176  0   1.8T  0 disk
sdm                         8:192  0   1.8T  0 disk
sdn                         8:208  0   1.8T  0 disk
sdo                         8:224  0   1.8T  0 disk
sdp                         8:240  0   1.8T  0 disk
sdq                        65:0    0   1.8T  0 disk
sdr                        65:16   0   1.8T  0 disk
sds                        65:32   0   1.8T  0 disk
sdt                        65:48   0   1.8T  0 disk
sdu                        65:64   0   1.8T  0 disk
sdv                        65:80   0   1.8T  0 disk
```

```
sdw                65:96   0   1.8T  0 disk
sdx                65:112  0   1.8T  0 disk
sdy                65:128  0   1.8T  0 disk
sdz                65:144  0   1.8T  0 disk
sdaa               65:160  0   1.8T  0 disk
sdab               65:176  0   1.8T  0 disk
sdac               65:192  0   1.8T  0 disk
sdad               65:208  0   1.8T  0 disk
sdae               65:224  0   1.8T  0 disk
sdaf               65:240  0   1.8T  0 disk
sdag               66:0    0   1.8T  0 disk
sdah               66:16   0   1.8T  0 disk
sdai               66:32   0   1.8T  0 disk
sdaj               66:48   0   1.8T  0 disk
sdak               66:64   0   1.8T  0 disk
sdal               66:80   0   1.8T  0 disk
nvme0n1              259:1    0 745.2G  0 disk
nvme1n1              259:0    0 745.2G  0 disk
```

在本例中，将使用以下原始块设备：

NVMe 设备

1. **/dev/nvme0n1**
2. **/dev/nvme1n1**

HDD 设备

1. **/dev/sdc**
2. **/dev/sdd**
3. **/dev/sde**
4. **/dev/sdf**

lv_vars.yaml 文件在所选设备中配置逻辑卷创建。它在 NVMe、基于 NVMe 的 bucket 索引和基于 HDD 的 OSD 上创建日志。逻辑卷的实际创建是由 **lv-create.yml** 启动的，它显示为 **lv_vars.yaml**。

该文件一次仅应具有一个 NVMe 设备。它还应仅引用与该特定 NVMe 设备关联的 HDD 设备。对于包含多个 NVMe 设备的 OSD，请为每个 NVMe 编辑 **lv_vars.yaml**，并为每个 NVMe 重复运行 **lv-create.yml**。下面将对此进行说明。

在这个示例中，这意味着 **lv-create.yml** 将首先在 **/dev/nvme0n1** 上运行，然后再次在 **/dev/nvme1n1** 上运行。

10.2.4. 将设备添加到 lv_vars.yaml

1. 作为 **root**，进入 **/usr/share/ceph-ansible/** 目录：

```
# cd /usr/share/ceph-ansible
```

2. 作为 **root**，将 **lv_vars.yaml** Ansible playbook 复制到当前目录中：

```
# cp infrastructure-playbooks/vars/lv_vars.yaml .
```

3. 对于第一次运行编辑该文件，使其包含以下行：

```
nvme_device: /dev/nvme0n1
hdd_devices:
  - /dev/sdc
  - /dev/sdd
```

日志大小、存储桶索引数、它们的大小和名称和存储桶索引的日志名称都可以在 **lv_vars.yaml** 中调整。如需更多信息，请参见文件中的注释。

10.2.5. 运行 lv-create.yml Ansible Playbook

lv-create.yml playbook 的目的是在单个 NVMe 中为对象网关存储桶索引和日志创建逻辑卷。它使用 **osd_scenario=lvm** 而不是使用 **osd_scenario=non-collocated**。通过自动化一些复杂的 LVM 创建和配置，**lv-create.yml** Ansible playbook 使以这种方式配置 Ceph 变得更加简单。

1. 作为 **root**，将 **lv-create.yml** Ansible playbook 复制到当前目录中：

```
# cp infrastructure-playbooks/lv-create.yml .
```

2. 确保存储设备是 raw
   在运行 **lv-create.yml** 在 NVMe 设备和 HDD 设备中创建逻辑卷前，请确保它们中没有文件系统、GPT、RAID 或其他签名。

   如果它们不是原始的，当运行 **lv-create.yml** 时可能会失败，并显示以下错误：

```
device /dev/sdc excluded by a filter
```

3. wipe 存储设备签名（可选）
   如果设备有签名，您可以使用 **wipefs** 擦除它们。

   以下是使用 **wipefs** 擦除设备的示例：

```
[root@c04-h01-6048r ~]# wipefs -a /dev/sdc
/dev/sdc: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdc: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdc: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdd
/dev/sdd: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sdd: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdd: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdd: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sde
/dev/sde: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sde: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sde: calling ioclt to re-read partition table: Success
[root@c04-h01-6048r ~]# wipefs -a /dev/sdf
/dev/sdf: 8 bytes were erased at offset 0x00000200 (gpt): 45 46 49 20 50 41 52 54
```

```
/dev/sdf: 8 bytes were erased at offset 0x1d19ffffe00 (gpt): 45 46 49 20 50 41 52 54
/dev/sdf: 2 bytes were erased at offset 0x000001fe (PMBR): 55 aa
/dev/sdf: calling ioclt to re-read partition table: Success
```

4. 运行 **lv-teardown.yml** Ansible playbook:
总是在运行 **lv-create.yml** 前运行 **lv-teardown.yml** ：

作为 **root**，将 **lv-teardown.yml** Ansible playbook 复制到当前目录中：

```
# cp infrastructure-playbooks/lv-teardown.yml .
```

运行 **lv-teardown.yml** Ansible playbook:

```
$ ansible-playbook lv-teardown.yml -i hosts
```

警告

运行 **lv-teardown.yml** Ansible 脚本时请谨慎操作。它销毁数据。确保您已备份了任何重要数据。

5. 运行 **lv-create.yml** Ansible playbook:

```
$ ansible-playbook lv-create.yml -i hosts
```

10.2.6. 复制第一个 NVMe LVM 配置

1. 查看 **lv-created.log**
lv-create.yml Ansible playbook 成功完成后，配置信息将写入 **lv-created.log**。打开 **lv-created.log** 并查找类似以下示例的信息：

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme0n1
  journal: ceph-journal-bucket-index-1-nvme0n1
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdc
  data_vg: ceph-hdd-vg-sdc
  journal: ceph-journal-sdc
  journal_vg: ceph-nvme-vg-nvme0n1
- data: ceph-hdd-lv-sdd
  data_vg: ceph-hdd-vg-sdd
  journal: ceph-journal-sdd
  journal_vg: ceph-nvme-vg-nvme0n1
```

2. 将此信息复制到 **lvm_volumes:** 下的 **group_vars/osds.yml** 中。

10.2.7. 在 NVMe 设备 2 上运行 lv-create.yml Playbook

以下说明是设置第二个 NVMe 设备的简短步骤。如果需要，请参阅上面的相关步骤以了解更多上下文。

1. 修改 **lv-vars.yaml** 以使用第二个 NVMe 和关联的 HDD。
   在上例中，**lv-vars.yaml** 现在会设置以下设备：

```
nvme_device: /dev/nvme1n1
hdd_devices:
  - /dev/sde
  - /dev/sdf
```

2. 运行 **lv-teardown.yml**:

```
$ ansible-playbook lv-teardown.yml -i hosts
```

3. 再次运行 **lv-create.yml**

```
$ ansible-playbook lv-create.yml -i hosts
```

10.2.8. 复制第二个 NVMe LVM 配置

1. 查看 **lv-created.log**
   lv-create.yml Ansible playbook 成功完成后，配置信息将写入 **lv-created.log**。打开 **lv-created.log** 并查找类似以下示例的信息：

```
- data: ceph-bucket-index-1
  data_vg: ceph-nvme-vg-nvme1n1
  journal: ceph-journal-bucket-index-1-nvme1n1
  journal_vg: ceph-nvme-vg-nvme1n1
- data: ceph-hdd-lv-sde
  data_vg: ceph-hdd-vg-sde
  journal: ceph-journal-sde
  journal_vg: ceph-nvme-vg-nvme1n1
- data: ceph-hdd-lv-sdf
  data_vg: ceph-hdd-vg-sdf
  journal: ceph-journal-sdf
  journal_vg: ceph-nvme-vg-nvme1n1
```

2. 将此信息复制到 **group_vars/osds.yml** 下已输入的信息 **lvm_volumes:** 下。

10.2.9. 验证 LVM 配置

1. 查看 LVM 配置
   根据两个 NVMe 设备和四个 HDD 的示例，应创建以下逻辑卷(LV)：

   每个 HDD 放置在两个 NVMe 设备（/dev/nvme0n1 上的两个 LV，两个在 /dev/nvme1n1 上）

   每个 HDD 放置一个数据 LV（每个 HDD 个 LV）

   每个存储桶索引有一个日志 LV（/dev/nvme0n1 上的一个 LV，/dev/nvme1 上的一个 LV）

   每个 bucket 索引中的一个数据 LV 放置在两个 NVMe 设备上（一个 LV 在 /dev/nvme0n1 上，一个 LV on /dev/nvme1n1）

LV 可在 **lsblk** 和 **lvscan** 输出中看到。在上图的示例中，Ceph 应当有 12 个 LV。作为粗略的检查，您可以对 Ceph LV 进行计数以确保至少有 12 个 LV，但最好确保在正确的存储设备（NVMe 与 HDD）上创建了适当的 LV。

lsblk 的输出示例如下所示：

```
[root@c04-h01-6048r ~]# lsblk
NAME                                  MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sda                                     8:0    0 465.8G  0 disk
├─sda1                                  8:1    0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sda2                                  8:2    0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sda3                                  8:3    0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdb                                     8:16   0 465.8G  0 disk
├─sdb1                                  8:17   0     4G  0 part
│ └─md1                                 9:1    0     4G  0 raid1 [SWAP]
├─sdb2                                  8:18   0   512M  0 part
│ └─md0                                 9:0    0   512M  0 raid1 /boot
└─sdb3                                  8:19   0 461.3G  0 part
  └─md2                                 9:2    0 461.1G  0 raid1 /
sdc                                     8:32   0   1.8T  0 disk
└─ceph--hdd--vg--sdc-ceph--hdd--lv--sdc 253:4    0   1.8T  0 lvm
sdd                                     8:48   0   1.8T  0 disk
└─ceph--hdd--vg--sdd-ceph--hdd--lv--sdd 253:5    0   1.8T  0 lvm
sde                                     8:64   0   1.8T  0 disk
└─ceph--hdd--vg--sde-ceph--hdd--lv--sde 253:10   0   1.8T  0 lvm
sdf                                     8:80   0   1.8T  0 disk
└─ceph--hdd--vg--sdf-ceph--hdd--lv--sdf 253:11   0   1.8T  0 lvm
sdg                                     8:96   0   1.8T  0 disk
sdh                                     8:112  0   1.8T  0 disk
sdi                                     8:128  0   1.8T  0 disk
sdj                                     8:144  0   1.8T  0 disk
sdk                                     8:160  0   1.8T  0 disk
sdl                                     8:176  0   1.8T  0 disk
sdm                                     8:192  0   1.8T  0 disk
sdn                                     8:208  0   1.8T  0 disk
sdo                                     8:224  0   1.8T  0 disk
sdp                                     8:240  0   1.8T  0 disk
sdq                                     65:0   0   1.8T  0 disk
sdr                                     65:16  0   1.8T  0 disk
sds                                     65:32  0   1.8T  0 disk
sdt                                     65:48  0   1.8T  0 disk
sdu                                     65:64  0   1.8T  0 disk
sdv                                     65:80  0   1.8T  0 disk
sdw                                     65:96  0   1.8T  0 disk
sdx                                     65:112 0   1.8T  0 disk
sdy                                     65:128 0   1.8T  0 disk
sdz                                     65:144 0   1.8T  0 disk
sdaa                                    65:160 0   1.8T  0 disk
sdab                                    65:176 0   1.8T  0 disk
sdac                                    65:192 0   1.8T  0 disk
sdad                                    65:208 0   1.8T  0 disk
sdae                                    65:224 0   1.8T  0 disk
sdaf                                    65:240 0   1.8T  0 disk
```

```
sdag                                                    66:0    0   1.8T  0 disk
sdah                                                    66:16   0   1.8T  0 disk
sdai                                                    66:32   0   1.8T  0 disk
sdaj                                                    66:48   0   1.8T  0 disk
sdak                                                    66:64   0   1.8T  0 disk
sdal                                                    66:80   0   1.8T  0 disk
nvme0n1                                                   259:0    0 745.2G  0 disk
├─ceph--nvme--vg--nvme0n1-ceph--journal--bucket--index--1--nvme0n1 253:0    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdc                    253:1    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme0n1-ceph--journal--sdd                    253:2    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme0n1-ceph--bucket--index--1                  253:3    0 729.1G  0 lvm
nvme1n1                                                   259:1    0 745.2G  0 disk
├─ceph--nvme--vg--nvme1n1-ceph--journal--bucket--index--1--nvme1n1 253:6    0   5.4G  0
lvm
├─ceph--nvme--vg--nvme1n1-ceph--journal--sde                    253:7    0   5.4G  0 lvm
├─ceph--nvme--vg--nvme1n1-ceph--journal--sdf                    253:8    0   5.4G  0 lvm
└─ceph--nvme--vg--nvme1n1-ceph--bucket--index--1                  253:9    0 729.1G  0 lvm
```

lvscan 的输出示例如下所示：

```
[root@c04-h01-6048r ~]# lvscan
  ACTIVE            '/dev/ceph-hdd-vg-sde/ceph-hdd-lv-sde' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdc/ceph-hdd-lv-sdc' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdf/ceph-hdd-lv-sdf' [<1.82 TiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-bucket-index-1-nvme1n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-sde' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-journal-sdf' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme1n1/ceph-bucket-index-1' [<729.10 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-bucket-index-1-nvme0n1' [5.37
GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdc' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-journal-sdd' [5.37 GiB] inherit
  ACTIVE            '/dev/ceph-nvme-vg-nvme0n1/ceph-bucket-index-1' [<729.10 GiB] inherit
  ACTIVE            '/dev/ceph-hdd-vg-sdd/ceph-hdd-lv-sdd' [<1.82 TiB] inherit
```

10.2.10. 编辑 osds.yml 和 all.yml Ansible Playbook

1. 将 **osd_objectstore** 设置为 **bluestore**
   除了将 **lv-create.log** 中的第二组信息添加到 **osds.yml** 外，在 **osds.yml** 和 **all.yml** 文件中还需要将 **osd_objectstore** 设置为 **bluestore**。

   该行在 **osds.yml** 和 **all.yml** 中应类似如下：

   ```
   osd_objectstore: bluestore
   ```

2. 在 中将 **osd_scenario** 设置为 **lvm osds.yml**

   osds.yml 文件应类似以下示例：

   ```
   # Variables here are applicable to all host groups NOT roles

   osd_objectstore: bluestore
   ```

```
osd_scenario: lvm
lvm_volumes:
  - data: ceph-bucket-index-1
    data_vg: ceph-nvme-vg-nvme0n1
    journal: ceph-journal-bucket-index-1-nvme0n1
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdc
    data_vg: ceph-hdd-vg-sdc
    journal: ceph-journal-sdc
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-hdd-lv-sdd
    data_vg: ceph-hdd-vg-sdd
    journal: ceph-journal-sdd
    journal_vg: ceph-nvme-vg-nvme0n1
  - data: ceph-bucket-index-1
    data_vg: ceph-nvme-vg-nvme1n1
    journal: ceph-journal-bucket-index-1-nvme1n1
    journal_vg: ceph-nvme-vg-nvme1n1
  - data: ceph-hdd-lv-sde
    data_vg: ceph-hdd-vg-sde
    journal: ceph-journal-sde
    journal_vg: ceph-nvme-vg-nvme1n1
  - data: ceph-hdd-lv-sdf
    data_vg: ceph-hdd-vg-sdf
    journal: ceph-journal-sdf
    journal_vg: ceph-nvme-vg-nvme1n1
```

10.2.11. 为 NVMe 安装 Ceph 并验证成功

1. 运行 **site.yml** Ansible playbook 来安装 Ceph

```
$ ansible-playbook -v site.yml -i hosts
```

2. 在安装完成后验证 Ceph 是否正常运行

```
# ceph -s
```

```
# ceph osd tree
```

显示 Ceph 正常运行的 **ceph -s** 输出示例：

```
# ceph -s
  cluster:
    id:     9ba22f4c-b53f-4c49-8c72-220aaf567c2b
    health: HEALTH_WARN
            Reduced data availability: 32 pgs inactive

  services:
    mon: 3 daemons, quorum b08-h03-r620,b08-h05-r620,b08-h06-r620
    mgr: b08-h03-r620(active), standbys: b08-h05-r620, b08-h06-r620
    osd: 42 osds: 42 up, 42 in

  data:
    pools:   4 pools, 32 pgs
```

```
    objects: 0 objects, 0 bytes
    usage:   0 kB used, 0 kB / 0 kB avail
    pgs:     100.000% pgs unknown
             32 unknown
```

显示 Ceph 正常运行的 **ceph osd tree** 输出示例：

```
[root@c04-h01-6048r ~]# ceph osd tree
ID  CLASS WEIGHT   TYPE NAME             STATUS REWEIGHT PRI-AFF
 -1       60.86740 root default
 -7        8.69534     host c04-h01-6048r
 10   hdd  1.81799         osd.10            up  1.00000 1.00000
 13   hdd  1.81799         osd.13            up  1.00000 1.00000
 21   hdd  1.81799         osd.21            up  1.00000 1.00000
 27   hdd  1.81799         osd.27            up  1.00000 1.00000
  6   ssd  0.71169        osd.6             up  1.00000 1.00000
 15   ssd  0.71169         osd.15            up  1.00000 1.00000
 -3        8.69534     host c04-h05-6048r
  7   hdd  1.81799        osd.7             up  1.00000 1.00000
 20   hdd  1.81799         osd.20            up  1.00000 1.00000
 29   hdd  1.81799         osd.29            up  1.00000 1.00000
 38   hdd  1.81799         osd.38            up  1.00000 1.00000
  4   ssd  0.71169        osd.4             up  1.00000 1.00000
 25   ssd  0.71169         osd.25            up  1.00000 1.00000
-22        8.69534     host c04-h09-6048r
 17   hdd  1.81799         osd.17            up  1.00000 1.00000
 31   hdd  1.81799         osd.31            up  1.00000 1.00000
 35   hdd  1.81799         osd.35            up  1.00000 1.00000
 39   hdd  1.81799         osd.39            up  1.00000 1.00000
  5   ssd  0.71169        osd.5             up  1.00000 1.00000
 34   ssd  0.71169         osd.34            up  1.00000 1.00000
 -9        8.69534     host c04-h13-6048r
  8   hdd  1.81799        osd.8             up  1.00000 1.00000
 11   hdd  1.81799         osd.11            up  1.00000 1.00000
 30   hdd  1.81799         osd.30            up  1.00000 1.00000
 32   hdd  1.81799         osd.32            up  1.00000 1.00000
  0   ssd  0.71169        osd.0             up  1.00000 1.00000
 26   ssd  0.71169         osd.26            up  1.00000 1.00000
-19        8.69534     host c04-h21-6048r
 18   hdd  1.81799         osd.18            up  1.00000 1.00000
 23   hdd  1.81799         osd.23            up  1.00000 1.00000
 36   hdd  1.81799         osd.36            up  1.00000 1.00000
 40   hdd  1.81799         osd.40            up  1.00000 1.00000
  3   ssd  0.71169        osd.3             up  1.00000 1.00000
 33   ssd  0.71169         osd.33            up  1.00000 1.00000
-16        8.69534     host c04-h25-6048r
 16   hdd  1.81799         osd.16            up  1.00000 1.00000
 22   hdd  1.81799         osd.22            up  1.00000 1.00000
 37   hdd  1.81799         osd.37            up  1.00000 1.00000
 41   hdd  1.81799         osd.41            up  1.00000 1.00000
  1   ssd  0.71169        osd.1             up  1.00000 1.00000
 28   ssd  0.71169         osd.28            up  1.00000 1.00000
 -5        8.69534     host c04-h29-6048r
  9   hdd  1.81799        osd.9             up  1.00000 1.00000
 12   hdd  1.81799         osd.12            up  1.00000 1.00000
 19   hdd  1.81799         osd.19            up  1.00000 1.00000
```

```
24   hdd  1.81799         osd.24             up  1.00000 1.00000
 2   ssd  0.71169        osd.2               up  1.00000 1.00000
14   ssd  0.71169         osd.14             up  1.00000 1.00000
```

Ceph 现已设置为将两个 NVMe 设备设置为最佳地用于对象存储网关。