Red Hat OpenStack Platform 16.1

使用容器化 Red Hat Ceph 部署 overcloud

配置 director 以部署和使用容器化 Red Hat Ceph 集群

Last Updated: 2023-11-07

Red Hat OpenStack Platform 16.1 使用容器化 Red Hat Ceph 部署 overcloud

配置 director 以部署和使用容器化 Red Hat Ceph 集群

OpenStack Team
rhos-docs@redhat.com

法律通告

Copyright © 2023 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本指南提供有关使用 Red Hat OpenStack Platform director 创建带有容器化 Red Hat Ceph Storage 集群的 overcloud 的信息。这包括通过 director 自定义 Ceph 集群的说明。

目录

使开源包含更多

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

对红帽文档提供反馈

我们感谢您对文档提供反馈信息。与我们分享您的成功秘诀。

使用直接文档反馈(DDF)功能

使用 **添加反馈** DDF 功能，用于特定句子、段落或代码块上的直接注释。

1. 以 *Multi-page HTML* 格式查看文档。
2. 请确定您看到文档右上角的 **反馈** 按钮。
3. 用鼠标指针高亮显示您想评论的文本部分。
4. 点 **添加反馈**。
5. 在**添加反馈**项中输入您的意见。
6. 可选：添加您的电子邮件地址，以便文档团队可以联系您以讨论您的问题。
7. 点 **Submit**。

第 1 章 简介

Red Hat OpenStack Platform director 创建名为 overcloud 的云环境。您可以使用 director 为 overcloud 配置额外的功能，包括与 Red Hat Ceph Storage 集成（与 director 或现有 Ceph Storage 集群创建的 Ceph 存储集群）。

本指南包含有关如何将现有 Ceph Storage 集群与 overcloud 集成的说明。这意味着 director 将 overcloud 配置为使用 Ceph Storage 集群来满足存储需要。您在 overcloud 配置之外管理并扩展集群。

本指南包含使用 overcloud 部署容器化 Red Hat Ceph Storage 集群的说明。director 使用通过 **ceph-ansible** 软件包提供的 Ansible playbook 来部署容器化 Ceph 集群。director 还管理集群的配置和扩展操作。

有关 Red Hat OpenStack Platform 中容器化服务的更多信息，请参阅 *Director 安装和使用*中的使用CLI工具配置基本的 overcloud。

1.1. CEPH STORAGE 简介

Red Hat Ceph Storage 是一个分布式数据对象存储，旨在提供卓越的性能、可靠性和可扩展性。分布式对象存储是未来的存储，因为它们适用于非结构化数据，并且因为客户端可以同时使用现代对象接口和旧接口。在每个 Ceph 部署的核心是 Ceph Storage 集群，它由几种类型的守护进程组成，但主要是这两者：

Ceph OSD （对象存储守护进程）

Ceph OSD 代表 Ceph 客户端存储数据。此外，Ceph OSD 利用 Ceph 节点的 CPU 和内存来执行数据复制、重新平衡、恢复、监控和报告功能。

Ceph monitor

Ceph 监控器维护 Ceph 存储集群映射的主副本，以及存储集群的当前状态。

有关 Red Hat Ceph Storage 的更多信息，请参阅 Red Hat Ceph Storage 架构指南 。

1.2. 要求

本指南包含 Director 安装和使用指南 补充的信息。

在使用 overcloud 部署容器化 Ceph Storage 集群前，您的环境必须包含以下配置：

- 安装了 Red Hat OpenStack Platform (RHOSP) director 的 undercloud 主机。请参阅在 undercloud 上安装 director。
- 建议用于 Red Hat Ceph Storage 的额外硬件。有关推荐硬件的更多信息，请参阅 Red Hat Ceph Storage 硬件指南。

重要

Ceph Monitor 服务在 overcloud Controller 节点上安装，因此您必须提供适当的资源来避免性能问题。确保环境中的 Controller 节点对 Ceph 监控数据的内存使用至少 16 GB RAM，并使用固态驱动器(SSD)存储。对于大型 Ceph 安装，提供至少 500 GB 的 Ceph 监控数据。当集群变得不稳定时，需要这个空间来避免 levelDB 增长。以下示例是 Ceph 存储集群的常见大小：

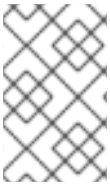
- Small: 250 terabytes
- Medium: 1 PB
- 大型：2 PB或更多。

如果您使用 Red Hat OpenStack Platform director 创建 Ceph Storage 节点，请注意以下要求：

1.2.1. Ceph Storage 节点要求

Ceph Storage 节点负责在 Red Hat OpenStack Platform 环境中提供对象存储。

有关如何为 Ceph Storage 节点选择处理器、内存、网络接口卡 (NIC) 和磁盘布局的信息，请参阅 *Red Hat Ceph Storage 硬件指南*中的 Red Hat Ceph Storage 的硬件选择建议 。每个 Ceph Storage 节点在服务器的主板上还需要有一个受支持的电源管理接口，如智能平台管理接口 (IPMI) 功能。



注意

Red Hat OpenStack Platform (RHOSP) director 使用 **ceph-ansible**，它不支持在 Ceph Storage 节点的根磁盘上安装 OSD。这意味着所支持的每个 Ceph Storage 节点需要至少两个磁盘。

Ceph Storage 节点和 RHEL 兼容性

- RHEL 8.2 支持 RHOSP 16.1。但是，映射到 Ceph Storage 角色的主机更新至最新的主 RHEL 版本。在升级到 RHOSP 16.1 及更新的版本前，请参阅红帽知识库文章 Red Hat Ceph Storage: 支持的配置。

放置组 (PG)

- Ceph Storage 使用放置组 (PG) 大规模推动动态高效的对象跟踪。如果 OSD 出现故障或集群进行重新平衡，Ceph 可移动或复制放置组及其内容，这意味着 Ceph Storage 集群可以有效地重新平衡并恢复。
- director 创建的默认放置组数量并非始终最佳，因此一定要根据要求计算正确的放置组数量。您可以使用放置组计算器计算正确的数量。要使用 PG 计算器，请输入每个服务的预计存储使用量（百分比表示），以及 Ceph 集群的其他属性，如 OSD 数量。计算器返回每个池的最佳 PG 数量。有关更多信息，请参阅每个池计算器的放置组 (PG)。
- 自动扩展是管理放置组的替代方法。借助自动扩展功能，您可以将每个服务的预期 Ceph Storage 要求设置为百分比而不是特定数量的放置组。Ceph 根据集群的使用方式自动扩展放置组。有关更多信息，请参阅 *Red Hat Ceph Storage 策略指南* 中的自动扩展放置组。

处理器

- 支持 Intel 64 或 AMD64 CPU 扩展的 64 位 x86 处理器。

网络接口卡

- 最少一个 1 Gbps 网络接口卡 (NIC)，但红帽建议您在生产环境中至少使用两个 NIC。对绑定的接口使用额外的 NIC，或代理标记的 VLAN 流量。为存储节点使用 10 Gbps 接口，特别是所创建的 Red Hat OpenStack Platform (RHOSP) 环境需要处理大量流量时。

电源管理

- 每个 Controller 节点在服务器的主板上都要有一个受支持的电源管理接口，如智能平台管理接口 (IPMI) 功能。

1.3. 其他资源

/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml 环境文件指示 director 使用从 ceph-ansible 项目派生的 playbook。这些 playbook 安装在 undercloud 的 **/usr/share/ceph-ansible/** 中。特别是，以下文件包含应用 playbook 的所有默认设置：

- **/usr/share/ceph-ansible/group_vars/all.yml.sample**

警告

虽然 **ceph-ansible** 使用 playbook 来部署容器化 Ceph 存储，但请不要编辑这些文件来自定义部署。取而代之，使用 heat 环境文件来覆盖这些 playbook 设置的默认值。如果您直接编辑 **ceph-ansible** playbook，您的部署会失败。

有关 director 为容器化 Ceph Storage 应用的默认设置的信息，请参阅 **/usr/share/openstack-tripleo-heat-templates/deployment/ceph-ansible** 中的 heat 模板。

注意

阅读这些模板时，需要深入了解环境文件和 heat 模板在 director 中的工作方式。如需更多信息，请参阅了解 Heat 模板和环境文件。

如需有关 RHOSP 中容器化服务的更多信息，请参阅 *Director 安装和使用指南中的使用 CLI 工具配置基本的 overcloud*。

第 2 章 为 OVERCLOUD 部署准备 CEPH STORAGE 节点

这种情境中的所有节点都是使用 IPMI 进行电源管理的裸机系统。这些节点不需要操作系统，因为 director 会将 Red Hat Enterprise Linux 8 镜像复制到每个节点。此外，这些节点上的 Ceph Storage 服务也容器化。director 在内省和置备过程中通过 Provisioning 网络与每个节点通信。所有节点都通过原生 VLAN 连接到此网络。

2.1. 清理 CEPH STORAGE 节点磁盘

Ceph Storage OSD 和日志分区需要 GPT 磁盘标签。这意味着，Ceph 存储上的额外磁盘需要在安装 Ceph OSD 服务前转换为 GPT。您必须从磁盘中删除所有元数据，以允许 director 在它们上设置 GPT 标签。

您可以通过在 **/home/stack/undercloud.conf** 文件中添加以下设置，将 director 配置为删除所有磁盘元数据：

```
clean_nodes=true
```

使用此选项时，裸机置备服务会运行一个额外的步骤来引导节点，并在每次将节点设置为 **可用时** 清理磁盘。这个过程会在第一个内省和每次部署前添加额外的电源周期。裸机置备服务使用 **wipefs --force --all** 命令来执行清理。

设置此选项后，运行 **openstack undercloud install** 命令来执行此配置更改。

警告

wipefs --force --all 命令可删除磁盘上的所有数据和元数据，但不执行安全擦除。安全的清除需要更长的时间。

2.2. 注册节点

将 JSON 格式的节点清单文件(**instackenv.json**)导入到 director，以便 director 可以与节点通信。此清单文件包含 director 可用于注册节点的硬件和电源管理详情：

```
{
    "nodes":[
        {
            "mac":[
                "b1:b1:b1:b1:b1:b1"
            ],
            "cpu":"4",
            "memory":"6144",
            "disk":"40",
            "arch":"x86_64",
            "pm_type":"ipmi",
            "pm_user":"admin",
            "pm_password":"p@55w0rd!",
            "pm_addr":"192.0.2.205"
        },
```

```
{
    "mac":[
        "b2:b2:b2:b2:b2:b2"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.206"
},
{
    "mac":[
        "b3:b3:b3:b3:b3:b3"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.207"
},
{
    "mac":[
        "c1:c1:c1:c1:c1:c1"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.208"
},
{
    "mac":[
        "c2:c2:c2:c2:c2:c2"
    ],
    "cpu":"4",
    "memory":"6144",
    "disk":"40",
    "arch":"x86_64",
    "pm_type":"ipmi",
    "pm_user":"admin",
    "pm_password":"p@55w0rd!",
    "pm_addr":"192.0.2.209"
},
{
    "mac":[
        "c3:c3:c3:c3:c3:c3"
    ],
```

```
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.210"
      },
      {
        "mac":[
          "d1:d1:d1:d1:d1:d1"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.211"
      },
      {
        "mac":[
          "d2:d2:d2:d2:d2:d2"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.212"
      },
      {
        "mac":[
          "d3:d3:d3:d3:d3:d3"
        ],
        "cpu":"4",
        "memory":"6144",
        "disk":"40",
        "arch":"x86_64",
        "pm_type":"ipmi",
        "pm_user":"admin",
        "pm_password":"p@55w0rd!",
        "pm_addr":"192.0.2.213"
      }
    ]
}
```

流程

1. 创建清单文件后，将文件保存到 stack 用户的主目录(**/home/stack/instackenv.json**)。
2. 初始化 stack 用户，然后将 **instackenv.json** 清单文件导入到 director：

```
$ source ~/stackrc
$ openstack overcloud node import ~/instackenv.json
```

openstack overcloud node import 命令导入清单文件，并将每个节点注册到 director。

3. 将内核和 ramdisk 镜像分配给每个节点：

```
$ openstack overcloud node configure <node>
```

结果

节点在 director 中注册和配置。

2.3. CEPH STORAGE 的预部署验证

为帮助避免 overcloud 部署失败，请验证服务器上是否存在所需的软件包。

2.3.1. 验证 ceph-ansible 软件包版本

undercloud 包含基于 Ansible 的验证，您可以在部署 overcloud 之前运行它们来识别潜在的问题。这些验证可以帮助您通过在问题发生前确定常见问题来避免 overcloud 部署失败。

流程

验证是否已安装了 **ceph-ansible** 软件包的修正版本：

```
$ ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-ansible-installed.yaml
```

2.3.2. 为预置备节点验证软件包

Ceph 只能服务具有特定的软件包集合的 overcloud 节点。使用预置备节点时，您可以验证这些软件包是否存在。

有关预置备节点的更多信息，请参阅 使用预置备节点配置基本 overcloud。

流程

验证服务器是否包含所需的软件包：

```
ansible-playbook -i /usr/bin/tripleo-ansible-inventory /usr/share/ansible/validation-playbooks/ceph-dependencies-installed.yaml
```

2.4. 手动将节点标记为配置集

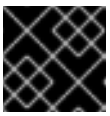
注册每个节点后，您必须检查硬件并将节点标记到特定的配置集中。使用 profile 标签将节点与类别匹配，然后将类别分配给部署角色。

流程

1. 触发硬件内省以检索每个节点的硬件属性：

```
$ openstack overcloud node introspect --all-manageable --provide
```

- **--all-manageable** 选项仅内省处于受管状态的节点。在此示例中，所有节点都处于受管理状态。
- **--provide** 选项会在内省后将所有节点重置 为**活动状**态。



重要

确保此过程成功完成。它可能需要 15 分钟来检查这些裸机节点。

2. 检索节点列表来识别它们的 UUID：

```
$ openstack baremetal node list
```

3. 在每个节点的 **properties/capabilities** 参数中添加 profile 选项，来手动将节点标记到特定的配置集。添加 **profile** 选项会将节点标记为相关的配置集。

注意

作为手动标记的替代选择，请使用 Automated Health Check (AHC)工具根据基准测试数据自动标记更多节点。

例如，典型的部署包含三个配置文件： **control**、**compute**、和 **ceph-storage**。运行以下命令为每个配置集标记三个节点：

```
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
$ openstack baremetal node set --property capabilities='profile:control,boot_option:local'
6faba1a9-e2d8-4b7c-95a2-c7fbdc12129a
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:compute,boot_option:local'
d930e613-3e14-44b9-8240-4f3559801ea6
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' 484587b2-b3b3-40d5-925b-a26a2fa3036f
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d010460b-38f2-4800-9cc4-d69f0d067efe
$ openstack baremetal node set --property capabilities='profile:ceph-
storage,boot_option:local' d930e613-3e14-44b9-8240-4f3559801ea6
```

提示

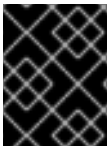
您还可以配置新的自定义配置集，可用于为 Ceph MON 和 Ceph MDS 服务标记节点。详情请查看 第 3 章 *在专用节点上部署 Ceph 服务*。

2.5. 为多磁盘集群定义根磁盘

如果节点使用多个磁盘，则 Director 在置备过程上必须识别根磁盘。例如，大多数 Ceph Storage 节点使用多个磁盘。默认情况下，director 在置备过程中将 overcloud 镜像写入根磁盘

您可以定义多个属性以帮助 director 识别根磁盘：

- **model**（字符串）：设备识别码。
- **vendor**（字符串）：设备厂商。
- **serial**（字符串）：磁盘序列号。
- **hctl**（字符串）：SCSI 的 Host:Channel:Target:Lun。
- **size**（整数）：设备的大小（以 GB 为单位）。
- **wwn**（字符串）：唯一的存储 ID。
- **wwn_with_extension**（字符串）：唯一存储 ID 附加厂商扩展名。
- **wwn_vendor_extension**（字符串）：唯一厂商存储标识符。
- **rotational**（布尔值）：旋转磁盘设备为 true (HDD)，否则为 false (SSD)。
- **name**（字符串）：设备名称，例如：/dev/sdb1。



重要

仅对具有持久名称的设备使用 **name** 属性。不要使用 **name** 来设置任何其他设备的根磁盘，因为此值在节点引导时可能会改变。

您可以使用其序列号指定根设备。

步骤

1. 从每个节点的硬件内省检查磁盘信息。运行以下命令以显示节点的磁盘信息：

```
(undercloud)$ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"
```

例如，一个节点的数据可能会显示 3 个磁盘：

```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
```

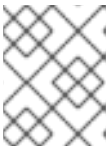
```
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380d00",
    "serial": "61866da04f380d001ea4e13c12e36ad6"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdc",
    "wwn_vendor_extension": "0x1ea4e31e121cfb45",
    "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f37fc00",
    "serial": "61866da04f37fc001ea4e31e121cfb45"
  }
]
```

2. 输入 **openstack baremetal node set --property root_device=**，为节点设置根磁盘。包括用于定义根磁盘的最合适的硬件属性值。

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":"
<serial_number>"}' <node-uuid>
```

例如：要将根设备设定为磁盘 2，其序列号为 **61866da04f380d001ea4e13c12e36ad6**，输入以下命令：

```
(undercloud)$ openstack baremetal node set --property root_device='{"serial":
"61866da04f380d001ea4e13c12e36ad6"}' 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0
```



注意

确保配置每个节点的 BIOS 以包括从您选择的根磁盘引导。将引导顺序配置为首先从网络引导，然后从根磁盘引导。

director 识别特定磁盘以用作根磁盘。运行 **openstack overcloud deploy** 命令时，director 置备 overcloud 镜像并将其写入根磁盘。

2.6. 使用 OVERCLOUD-MINIMAL 镜像来避免使用红帽订阅授权

默认情况下，director 在置备过程中将 QCOW2 **overcloud-full** 镜像写入根磁盘。**overcloud-full** 镜像使用有效的红帽订阅。但是，如果您不希望运行其他 OpenStack 服务或消耗您的订阅授权，您还可以使用 **overcloud-minimal** 镜像来置备裸机操作系统。

在您希望只使用 Ceph 守护进程来置备节点时，会发生此情况的常见用例。对于此情况和类似用例，使用 **overcloud-minimal** 镜像选项以避免达到您购买的红帽订阅的极限。有关如何获取 **overcloud-minimal** 镜像的详情，请参考 获取 overcloud 节点的镜像。

注意

Red Hat OpenStack Platform (RHOSP) 订阅包含 Open vSwitch (OVS)，但使用 **overcloud-minimal** 镜像时核心服务（如 OVS）不可用。部署 Ceph Storage 节点不需要 OVS。使用 **linux_bond** 定义绑定，而不使用 **ovs_bond**。有关 **linux_bond** 的更多信息，请参阅 Linux 绑定选项。

步骤

1. 要配置 director 使用 **overcloud-minimal** 镜像，创建包含以下镜像定义的环境文件：

```
parameter_defaults:
  <roleName>Image: overcloud-minimal
```

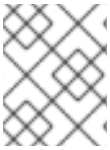
2. 将 **<roleName>** 替换为角色的名称，并将 **Image** 加到角色名称的后面。以下示例显示了 Ceph 存储节点的 **overcloud-minimal** 镜像：

```
parameter_defaults:
  CephStorageImage: overcloud-minimal
```

3. 在 **roles_data.yaml** 角色定义文件中，将 **rhsm_enforce** 参数设置为 **False**。

```
rhsm_enforce: False
```

4. 将环境文件传递给 **openstack overcloud deploy** 命令。

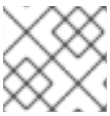


注意

overcloud-minimal 镜像仅支持标准 Linux 网桥，不支持 OVS，因为 OVS 是需要 Red Hat OpenStack Platform 订阅权利的 OpenStack 服务。

第 3 章 在专用节点上部署 CEPH 服务

默认情况下，director 在 Controller 节点上部署 Ceph MON 和 Ceph MDS 服务。这适用于小型部署。但是，如果部署较大的部署，红帽建议您在专用节点上部署 Ceph MON 和 Ceph MDS 服务，以提高 Ceph 集群的性能。为要在专用节点上隔离的服务创建一个自定义角色。



注意

如需有关自定义角色的更多信息，请参阅高级 Overcloud 自定义指南中的创建新角色。

director 使用以下文件作为所有 overcloud 角色的默认引用：

- **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml**

3.1. 创建自定义角色文件

要创建自定义角色文件，请完成以下步骤：

流程

1. 在 **/home/stack/templates/** 中生成 **roles_data.yaml** 文件的副本，以便您可以添加自定义角色：

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles_data.yaml
/home/stack/templates/roles_data_custom.yaml
```

2. 在 **openstack overcloud deploy** 命令中包含新的自定义角色文件。

3.2. 为 CEPH MON 服务创建自定义角色和类别

完成以下步骤，为 Ceph MON 角色创建自定义角色 **CephMon**，以及类别 **ceph-mon**。您必须已经有默认角色数据文件的副本，如 第 3 章 *在专用节点上部署 Ceph 服务* 所述。

流程

1. 打开 **/home/stack/templates/roles_data_custom.yaml** 文件。
2. 从 Controller 角色移除 Ceph MON 服务 **OS::TripleO::Services::CephMon** 的服务条目。
3. 将 **OS::TripleO::Services::CephClient** 服务添加到 Controller 角色：

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

4. 在 **roles_data_custom.yaml** 文件的末尾，添加一个自定义 **CephMon** 角色，其中包含 Ceph MON 服务和所有其他必要的节点服务：

```
- name: CephMon
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMon
```

5. 输入 **openstack flavor create** 命令，为 **CephMon** 角色定义一个名为 **ceph-mon** 的新类别：

```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mon
```

注意

有关此命令的更多信息，请输入： **openstack flavor create --help**。

6. 将此类别映射到一个新的配置文件，也称为 **ceph-mon** ：

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mon" ceph-mon
```

注意

有关此命令的更多信息，请输入 **openstack flavor set --help**。

7. 将节点标记到新的 **ceph-mon** 配置集中：

```
$ openstack baremetal node set --property capabilities='profile:ceph-mon',boot_option:local'
UUID
```

8. 将以下配置添加到 **node-info.yaml** 文件中，将 **ceph-mon** 类别与 CephMon 角色关联：

```
parameter_defaults:
  OvercloudCephMonFlavor: CephMon
  CephMonCount: 3
```

有关标记节点的更多信息，请参阅 第 2.4 节 “手动将节点标记为配置集”。有关自定义角色配置集的更多信息，请参阅 标记节点 Into Profiles。

3.3. 为 CEPH MDS 服务创建自定义角色和类别

完成以下步骤，为 Ceph MDS 角色创建自定义角色 **CephMDS** 和类别 **ceph-mds**。您必须已经有默认角色数据文件的副本，如 第 3 章 *在专用节点上部署 Ceph 服务* 所述。

流程

1. 打开 **/home/stack/templates/roles_data_custom.yaml** 文件。
2. 从 Controller 角色中删除 Ceph MDS 服务 **OS::TripleO::Services::CephMds** 的服务条目：

```
[...]
- name: Controller # the 'primary' role goes first
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    # - OS::TripleO::Services::CephMds (1)
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CinderApi
[...]
```

(1) 注释掉此行。在下一步中，您要将此服务添加到新的自定义角色。

3. 在 **roles_data_custom.yaml** 文件的末尾，添加一个自定义 **CephMDS** 角色，其中包含 Ceph MDS 服务以及所有其他必要的节点服务：

```
- name: CephMDS
  ServicesDefault:
    # Common Services
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::FluentdClient
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    # Role-Specific Services
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephClient (1)
```

1 Ceph MDS 服务需要 admin 密钥环，您可以使用 Ceph MON 或 Ceph 客户端服务进行设置。如果您在没有 Ceph MON 服务的专用节点上部署 Ceph MDS，还必须将 Ceph 客户端

4. 输入 **openstack flavor create** 命令，为此角色定义一个名为 **ceph-mds** 的新类别：

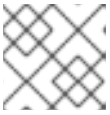
```
$ openstack flavor create --id auto --ram 6144 --disk 40 --vcpus 4 ceph-mds
```

注意

有关这个命令的更多信息，请输入 **openstack flavor create --help**。

5. 将新的 **ceph-mds** 类别映射到新配置文件，也称为 **ceph-mds** ：

```
$ openstack flavor set --property "cpu_arch"="x86_64" --property
"capabilities:boot_option"="local" --property "capabilities:profile"="ceph-mds" ceph-mds
```



注意

有关此命令的更多信息，请输入 **openstack flavor set --help**。

6. 将节点标记到新的 **ceph-mds** 配置集中：

```
$ openstack baremetal node set --property capabilities='profile:ceph-mds,boot_option:local'
UUID
```

有关标记节点的更多信息，请参阅 第 2.4 节 “手动将节点标记为配置集”。有关自定义角色配置集的更多信息，请参阅 标记节点 Into Profiles。

第 4 章 自定义存储服务

director 提供的 heat 模板集合已包含必要的模板和环境文件，以启用基本的 Ceph Storage 配置。

director 使用 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 环境文件来创建 Ceph 集群，并将它与 overcloud 在部署过程中集成。此集群具有容器化 Ceph Storage 节点。如需有关 OpenStack 中容器化服务的更多信息，请参阅 *Director 安装和使用指南中的使用 CLI 工具配置基本的 overcloud*。

Red Hat OpenStack director 还对部署的 Ceph 集群应用基本默认设置。您还必须在自定义环境文件中定义任何其他配置：

流程

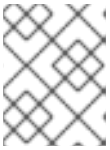
1. 在 **/home/stack/templates/** 中创建 **storage-config.yaml** 文件。在本例中，**~/templates/storage-config.yaml** 文件包含环境中大多数与 overcloud 相关的自定义设置。您在自定义环境文件中包含的参数会覆盖 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 文件中对应的默认设置。

2. 在 **~/templates/storage-config.yaml** 中添加一个 **parameter_defaults** 部分。本节包含 overcloud 的自定义设置。例如，要将 **vxlan** 设置为网络服务(**neutron**)的网络类型，请在自定义环境文件中添加以下片断：

```
parameter_defaults:
  NeutronNetworkType: vxlan
```

3. 如有必要，根据您的要求在 **parameter_defaults** 下设置以下选项：

Option	描述	默认值
CinderEnableIscsiBackend	启用 iSCSI 后端	false
CinderEnableRbdBackend	启用 Ceph Storage 后端	true
CinderBackupBackend	将 ceph 或 swift 设置为卷备份的后端。更多信息请参阅 第 4.4 节 “将备份服务配置为使用 Ceph”。	ceph
NovaEnableRbdBackend	为 Nova 临时存储启用 Ceph Storage	true
GlanceBackend	定义镜像服务应使用的后端：**rbd** (Ceph)、**swift** 或 **file**	rbd
GnocchiBackend	定义块存储服务应使用的后端：**rbd** (Ceph)、**swift** 或 **file**	rbd



注意

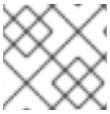
如果要使用默认设置，您可以从 **~/templates/storage-config.yaml** 中省略一个选项。

自定义环境文件的内容会根据您在以下部分中应用的设置而改变。有关已完成的示例，请参阅 附录 A, *示例环境文件：创建 Ceph Storage 集群*。

以下小节包含有关覆盖 director 应用的通用默认存储服务设置的信息。

4.1. 启用 CEPH 元数据服务器

Ceph 元数据服务器(MDS)运行 **ceph-mds** 守护进程，后者管理与 CephFS 上存储文件相关的元数据。CephFS 可以通过 NFS 使用。有关通过 NFS 使用 CephFS 的更多信息，请参阅 File System Guide 和 CephFS via NFS Back End Guide for the Shared File Systems service .



注意

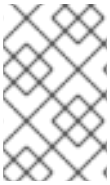
红帽支持对共享文件系统服务通过 NFS 后端使用 CephFS 部署 Ceph MDS。

流程

要启用 Ceph 元数据服务器，在创建 overcloud 时调用以下环境文件：

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml**

更多信息请参阅 第 7.2 节 “启动 overcloud 部署”。有关 Ceph 元数据服务器的更多信息，请参阅配置元数据服务器守护进程。



注意

默认情况下，Ceph 元数据服务器将部署到 Controller 节点上。您可以在自己的专用节点上部署 Ceph 元数据服务器。更多信息请参阅 第 3.3 节 “为 Ceph MDS 服务创建自定义角色和类别”。

4.2. 启用 CEPH 对象网关

Ceph 对象网关(RGW)为应用提供 Ceph 存储群集中对象存储功能的接口。在部署 RGW 时，您可以将默认的 Object Storage 服务(**swift**)替换为 Ceph。如需更多信息，请参阅 对象网关配置和管理指南。

流程

要在部署中启用 RGW，在创建 overcloud 时调用以下环境文件：

- **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml**

更多信息请参阅 第 7.2 节 “启动 overcloud 部署”。

默认情况下，Ceph Storage 允许每个 OSD 有 250 个放置组。启用 RGW 时，Ceph Storage 会创建 RGW 所需的 6 个额外池。新池有：

- .rgw.root
- default.rgw.control
- default.rgw.meta
- default.rgw.log
- default.rgw.buckets.index

- default.rgw.buckets.data

注意

在您的部署中，**default** 替换为池所属的区域的名称。

因此，当您启用 RGW 时，使用 **CephPoolDefaultPgNum** 参数设置默认的 **pg_num**，以考虑新池。有关如何为 Ceph 池计算放置组数量的更多信息，请参阅 第 5.4 节 "为不同的 Ceph 池分配自定义属性"。

Ceph 对象网关是默认对象存储服务的直接替换。因此，通常使用 **swift** 的所有其他服务都可以无缝地使用 Ceph 对象网关，而无需进一步配置。如需更多信息，请参阅块存储备份指南。

4.3. 配置 CEPH 对象存储以使用外部 CEPH 对象网关

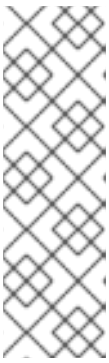
Red Hat OpenStack Platform (RHOSP) director 支持将外部 Ceph 对象网关(RGW)配置为对象存储服务。要使用外部 RGW 服务进行身份验证，您必须配置 RGW，以验证 Identity 服务(keystone)中的用户及其角色。

有关如何配置外部 Ceph 对象网关的更多信息，请参阅 *Using Keystone with the Ceph Object Gateway Guide* 中的 Configuring the Ceph Object Gateway to use Keystone authentication 。

流程

1. 将以下 **parameter_defaults** 添加到自定义环境文件中，如 **swift-external-params.yaml**，并调整值以适合您的部署：

```
parameter_defaults:
  ExternalSwiftPublicUrl: 'http://<Public RGW endpoint or
loadbalancer>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftInternalUrl: 'http://<Internal RGW endpoint>:8080/swift/v1/AUTH_%
(project_id)s'
  ExternalSwiftAdminUrl: 'http://<Admin RGW endpoint>:8080/swift/v1/AUTH_%(project_id)s'
  ExternalSwiftUserTenant: 'service'
  SwiftPassword: 'choose_a_random_password'
```



注意

代码片段示例包含的参数值可能与您在环境中使用的值不同：

- 远程 RGW 实例侦听的默认端口 **8080**。端口可能根据外部 RGW 的配置方式而有所不同。
- overcloud 中创建的 **swift** 用户使用 **SwiftPassword** 参数定义的密码。您必须使用 **rgw_keystone_admin_password**，将外部 RGW 实例配置为使用同一密码与 Identity 服务进行身份验证。

2. 将以下代码添加到 Ceph 配置文件中，将 RGW 配置为使用 Identity 服务。替换变量值以适合您的环境：

```
rgw_keystone_api_version = 3
rgw_keystone_url = http://<public Keystone endpoint>:5000/
rgw_keystone_accepted_roles = member, Member, admin
rgw_keystone_accepted_admin_roles = ResellerAdmin, swiftoperator
```

```
    rgw_keystone_admin_domain = default
    rgw_keystone_admin_project = service
    rgw_keystone_admin_user = swift
    rgw_keystone_admin_password =
<password_as_defined_in_the_environment_parameters>
    rgw_keystone_implicit_tenants = true
    rgw_keystone_revocation_interval = 0
    rgw_s3_auth_use_keystone = true
    rgw_swift_versioning_enabled = true
    rgw_swift_account_in_url = true
```

注意

director 默认在 Identity 服务中创建以下角色和用户：

- rgw_keystone_accepted_admin_roles: ResellerAdmin, swiftoperator
- rgw_keystone_admin_domain: default
- rgw_keystone_admin_project: service
- rgw_keystone_admin_user: swift

3. 使用与部署相关的任何其他环境文件，使用额外的环境文件部署 overcloud：

```
openstack overcloud deploy --templates \
-e <your_environment_files>
-e /usr/share/openstack-tripleo-heat-templates/environments/swift-external.yaml
-e swift-external-params.yaml
```

验证

1. 以 **stack** 用户的身份登录 undercloud。
2. 获取 **overcloudrc** 文件：

```
$ source ~/stackrc
```

3. 验证 Identity 服务(keystone)中是否存在端点：

```
$ openstack endpoint list --service object-store

+---------+-----------+-------+-------+---------+-----------+--------------+
| ID | Region    | Service Name | Service Type | Enabled | Interface | URL |
+---------+-----------+-------+-------+---------+-----------+--------------+
| 233b7ea32aaf40c1ad782c696128aa0e | regionOne | swift | object-store | True    | admin     |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
| 4ccde35ac76444d7bb82c5816a97abd8 | regionOne | swift | object-store | True    | public    |
https://192.168.24.2:13808/v1/AUTH_%(project_id)s |
| b4ff283f445348639864f560aa2b2b41 | regionOne | swift | object-store | True    | internal  |
http://192.168.24.3:8080/v1/AUTH_%(project_id)s |
+---------+-----------+-------+-------+---------+-----------+--------------+
```

4. 创建测试容器：

```
$ openstack container create <testcontainer>
+---------------+--------------+----------------------------------+
| account | container | x-trans-id |
+---------------+--------------+----------------------------------+
| AUTH_2852da3cf2fc490081114c434d1fc157 | testcontainer | tx6f5253e710a2449b8ef7e-
005f2d29e8 |
+---------------+--------------+----------------------------------+
```

5. 创建配置文件以确认您可以上传数据到容器：

```
$ openstack object create testcontainer undercloud.conf
+----------------+--------------+----------------------------------+
| object         | container    | etag                             |
+----------------+--------------+----------------------------------+
| undercloud.conf | testcontainer | 09fcffe126cac1dbac7b89b8fd7a3e4b |
+----------------+--------------+----------------------------------+
```

6. 删除测试容器：

```
$ openstack container delete -r <testcontainer>
```

4.4. 将备份服务配置为使用 CEPH

默认情况下，块存储备份服务(**cinder-backup**)被禁用。要启用块存储备份服务，请完成以下步骤：

流程

在创建 overcloud 时调用以下环境文件：

- **/usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml**

4.5. 为 CEPH 节点配置多个绑定接口

使用绑定接口来组合多个 NIC，并为网络连接添加冗余。如果您在 Ceph 节点上有足够的 NIC，可以在每个节点上创建多个绑定接口来扩展冗余能力。

然后，您可以对节点需要的每个网络连接使用绑定接口。这可为每个网络提供冗余和专用连接。

绑定接口的最简单实施涉及使用两个绑定，一个用于 Ceph 节点使用的每个存储网络。这些网络如下：

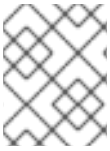
前端存储网络(StorageNet)

Ceph 客户端使用此网络与对应的 Ceph 集群交互。

后端存储网络(StorageMgmtNet)

Ceph 集群使用此网络来根据集群的放置组策略平衡数据。有关更多信息，请参阅 *Red Hat Ceph 架构指南中的* 放置组(PG)。

要配置多个绑定接口，您必须创建新的网络接口模板，因为 director 不提供任何可用于部署多个绑定 NIC 的样本模板。但是，director 提供部署单个绑定接口的模板。此模板是 **/usr/share/openstack-tripleo-heat-templates/network/bond-with-vlans/ceph-storage.yaml**。您可以在此模板中为额外 NIC 定义一个额外的绑定接口。



注意

有关创建自定义接口模板的更多信息，*请参阅高级 Overcloud 自定义指南中的* 创建自定义接口模板。

以下片段包含在 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 文件中定义的单个绑定接口的默认定义：

```
type: ovs_bridge // 1
name: br-bond
members:
 -
   type: ovs_bond // 2
   name: bond1 // 3
   ovs_options: {get_param: BondInterfaceOvsOptions} 4
   members: // 5
     -
       type: interface
       name: nic2
       primary: true
     -
       type: interface
       name: nic3
 -
   type: vlan // 6
   device: bond1 // 7
   vlan_id: {get_param: StorageNetworkVlanID}
   addresses:
     -
       ip_netmask: {get_param: StorageIpSubnet}
 -
   type: vlan
   device: bond1
   vlan_id: {get_param: StorageMgmtNetworkVlanID}
   addresses:
     -
       ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1. 名为 **br-bond** 的单个网桥包含此模板中定义的绑定。此行定义网桥类型，即 OVS。
2. **br-bond** 网桥的第一个成员是名为 **bond1** 的绑定接口本身。此行定义 **bond1** 的绑定类型，这也是 OVS。
3. 默认绑定名为 **bond1**。
4. **ovs_options** 条目指示 director 使用一组特定的绑定模块指令。这些指令通过 **BondInterfaceOvsOptions** 传递，您也可以在此文件中配置这些指令。有关配置绑定模块指令的详情请参考 第 4.5.1 节 “配置绑定模块指令”。
5. 绑定的 **members** 部分定义了哪些网络接口由 **bond1** 绑定。在本例中，绑定接口使用 **nic2** （设置为主接口）和 **nic3**。
6. **br-bond** 网桥有两个成员：一个用于前端(**StorageNetwork**)和后端(**StorageMgmtNetwork**)存储网络的 VLAN。

7 **device** 参数定义 VLAN 应使用的设备。在本例中，两个 VLAN 使用绑定接口 **bond1**。

对于至少两个 NIC，您可以定义一个额外的网桥和绑定接口。然后，您可以将其中一个 VLAN 移到新的绑定接口，这会增加两个存储网络连接的吞吐量和可靠性。

当您自定义 **/usr/share/openstack-tripleo-heat-templates/network/bond-with-vlans/ceph-storage.yaml** 文件时，红帽建议您使用 Linux 绑定(**type: linux_bond**)而不是默认的 OVS (类型：**ovs_bond**)。这个绑定类型更适合于企业生产部署。

以下编辑的代码片段定义了额外的 OVS 网桥(**br-bond2**)，它负责存储名为 **bond2** 的新 Linux 绑定。**bond2** 接口使用两个额外的 NIC， **nic4** 和 **nic5**，它仅用于后端存储网络流量：

```
    type: ovs_bridge
    name: br-bond
    members:
      -
        type: linux_bond
        name: bond1
        bonding_options: {get_param: BondInterfaceOvsOptions} // 1
        members:
          -
            type: interface
            name: nic2
            primary: true
          -
            type: interface
            name: nic3
      -
        type: vlan
        device: bond1
        vlan_id: {get_param: StorageNetworkVlanID}
        addresses:
          -
            ip_netmask: {get_param: StorageIpSubnet}
-
    type: ovs_bridge
    name: br-bond2
    members:
      -
        type: linux_bond
        name: bond2
        bonding_options: {get_param: BondInterfaceOvsOptions}
        members:
          -
            type: interface
            name: nic4
            primary: true
          -
            type: interface
            name: nic5
      -
        type: vlan
        device: bond1
        vlan_id: {get_param: StorageMgmtNetworkVlanID}
```

```
    addresses:
      -
        ip_netmask: {get_param: StorageMgmtIpSubnet}
```

1 因为 **bond1** 和 **bond2** 都是 Linux 绑定（而不是 OVS），它们使用 **bonding_options** 而不是 **ovs_options** 来设置绑定指令。更多信息请参阅 第 4.5.1 节 “配置绑定模块指令”。

有关此自定义模板的完整内容，请参阅 附录 B, *自定义接口模板示例：多个绑定接口*。

4.5.1. 配置绑定模块指令

添加并配置绑定接口后，使用 **BondInterfaceOvsOptions** 参数来设置您希望每个绑定接口使用的指令。您可以在 **/usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml** 文件的 parameter **:** 部分找到此信息。以下片段显示了此参数的默认定义（即空）：

```
BondInterfaceOvsOptions:
  default: ''
  description: The ovs_options string for the bond interface. Set
               things like lacp=active and/or bond_mode=balance-slb
               using this option.
  type: string
```

在 **default:** 行中定义您需要的选项。例如，要使用 802.3ad（模式 4）和 LACP 速率 1 (fast)，使用 **'mode=4 lacp_rate=1'**:

```
BondInterfaceOvsOptions:
  default: 'mode=4 lacp_rate=1'
  description: The bonding_options string for the bond interface. Set
               things like lacp=active and/or bond_mode=balance-slb
               using this option.
  type: string
```

有关其他支持的绑定选项的更多信息，*请参阅高级 Overcloud 优化* 指南中的 Open vSwitch 绑定选项。有关自定义 **/usr/share/openstack-tripleo-heat-templates/network/bond-with-vlans/ceph-storage.yaml** 模板的完整内容，请参阅 附录 B, *自定义接口模板示例：多个绑定接口*。

第 5 章 自定义 CEPH STORAGE 集群

director 使用默认配置部署容器化 Red Hat Ceph Storage。您可以通过覆盖默认设置来自定义 Ceph Storage。

先决条件

要部署容器化 Ceph Storage，您必须在 overcloud 部署期间包括 **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** 文件。此环境文件定义了以下资源：

- **CephAnsibleDisksConfig** - 此资源映射 Ceph Storage 节点磁盘布局。更多信息请参阅 第 5.3 节 “映射 Ceph Storage 节点磁盘布局”。
- **CephConfigOverrides** - 此资源将所有其他自定义设置应用到 Ceph Storage 集群。

使用这些资源覆盖 director 为容器化 Ceph Storage 设置的任何默认值。

流程

1. 启用 Red Hat Ceph Storage 4 工具存储库：

```
$ sudo subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
```

2. 在 undercloud 上安装 **ceph-ansible** 软件包：

```
$ sudo dnf install ceph-ansible
```

3. 要自定义 Ceph 存储集群，请在新环境文件中定义自定义参数，如 **/home/stack/templates/ceph-config.yaml**。您可以在环境文件的 **parameter_defaults** 部分中使用以下语法应用 Ceph Storage 集群设置：

```
parameter_defaults:
  CephConfigOverrides:
    section:
      KEY:VALUE
```

注意

您可以将 **CephConfigOverrides** 参数应用到 **ceph.conf** 文件的 **[global]** 部分，以及其他部分，如 **[osd]**、**[mon]** 和 **[client]**。如果指定一个部分，**key:value** 数据将进入指定的部分。如果您没有指定部分，则数据默认进入 **[global]** 部分。有关 Ceph Storage 配置、自定义和支持参数的信息，请参阅 Red Hat Ceph Storage 配置指南。

4. 将 **KEY** 和 **VALUE** 替换为您要应用的 Ceph 集群设置。例如，在 **global** 部分中，**max_open_files** 是 **KEY**，**131072** 是对应的 **VALUE** ：

```
parameter_defaults:
  CephConfigOverrides:
    global:
      max_open_files: 131072
    osd:
      osd_scrub_during_recovery: false
```

此配置会导致 Ceph 集群的配置文件中定义的以下设置：

```
[global]
max_open_files = 131072
[osd]
osd_scrub_during_recovery = false
```

5.1. 设置 CEPH-ANSIBLE 组变量

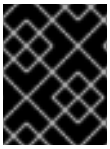
ceph-ansible 工具是用于安装和管理 Ceph 存储集群的 playbook。

ceph-ansible 工具具有一个 **group_vars** 目录，用于定义配置选项和这些选项的默认设置。使用 **group_vars** 目录设置 Ceph Storage 参数。

有关 **group_vars** 目录的信息，请参阅 *安装指南*中的安装 Red Hat Ceph Storage 集群 。

若要更改 director 中的变量默认值，可使用 **CephAnsibleExtraConfig** 参数传递 heat 环境文件中的新值。例如，要将 **ceph-ansible** 组变量 **journal_size** 设置为 40960，创建一个具有以下 **journal_size** 定义的环境文件：

```
parameter_defaults:
  CephAnsibleExtraConfig:
    journal_size: 40960
```



重要

使用覆盖参数更改 **ceph-ansible** 组变量；不要直接编辑 undercloud 的 **/usr/share/ceph-ansible** 目录中的组变量。

5.2. 用于带有 CEPH STORAGE 的 RED HAT OPENSTACK PLATFORM 的 CEPH 容器

Ceph 容器需要配置 OpenStack Platform 以使用 Ceph，即使具有外部 Ceph 集群。要与 Red Hat Enterprise Linux 8 兼容，Red Hat OpenStack Platform (RHOSP) 16 需要 Red Hat Ceph Storage 4。Ceph Storage 4 容器托管在 registry.redhat.io 中，这是需要身份验证的 registry。

您可以使用 heat 环境参数 **ContainerImageRegistryCredentials** 在 **registry.redhat.io** 进行身份验证。如需更多信息，请参阅 容器镜像准备参数。

5.3. 映射 CEPH STORAGE 节点磁盘布局

部署容器化 Ceph Storage 时，您必须映射磁盘布局，并为 Ceph OSD 服务指定专用块设备。您可以在之前创建的环境文件中执行此映射，以定义自定义 Ceph 参数：**/home/stack/templates/ceph-config.yaml**。

使用 **parameter_defaults** 中的 **CephAnsibleDisksConfig** 资源映射您的磁盘布局。此资源使用以下变量：

变量	必需？	默认值（如果未设置）	Description

变量	必需？	默认值（如果未设置）	Description
osd_scenario	是	lvm 注： 默认值为 **lvm**。	**lvm** 值允许 **ceph-ansible** 使用 **ceph-volume** 来配置 OSD、**block.db** 和 BlueStore WAL 设备。
devices	是	NONE.必须设置变量。	要用于节点上 OSD 的块设备列表。
dedicated_devices	是（仅在 **osd_scenario** 为 **非并置**）	devices	将 **devices** 参数中的每个条目映射到专用日志块设备的列表。您只能在 **osd_scenario=non-collocated** 时使用此变量。
dmcrypt	否	false	设置 OSD 上存储的数据是加密的(**true**)还是未加密的(**false**)。
osd_objectstore	否	bluestore 注： 默认值为 **bluestore**。	设置 Ceph 使用的存储后端。 注意：虽然值默认为 **bluestore**，但您可以在 collated 或 non-collated 场景中将 **osd_scenario** 设置为 **filestore**。您可以在非协调场景中将值设置为 **filestore**，其中 **dedicated_devices** 标识日志记录磁盘。您可以在联合场景中将值设置为 **filestore**，您可以在其中对设备中定义的磁盘进行分区，并将 OSD 数据和日志数据存储在同一设备上。

5.3.1. 使用 BlueStore

流程

1. 要指定您要用作 Ceph OSD 的块设备，请使用以下片段的变体：

```
parameter_defaults:
```

```
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n1
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. 因为 **/dev/nvme0n1** 位于执行的设备类中，所以 example **parameter_defaults** 会生成三个在 **/dev/sdb**、**/dev/sdc** 和 **/dev/sdd** 上运行的 OSD。三个 OSD 使用 **/dev/nvme0n1** 作为 **block.db** 和 BlueStore WAL 设备。**ceph-volume** 工具使用 **batch** 子命令执行此操作。每个 Ceph Storage 节点都会重复相同的配置，并假设硬件统一。如果 **block.db** 和 BlueStore WAL 数据位于与 OSD 相同的磁盘上，则使用以下方式更改参数默认值：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

5.3.2. 使用持久名称引用设备

流程

1. 在某些节点中，磁盘路径（如 **/dev/sdb** 和 **/dev/sdc** ）可能无法在重启期间指向同一块设备。如果您的 Ceph Storage 节点是这种情况，请使用 **/dev/disk/by-path/** 符号链接为每个磁盘指定，以确保在部署过程中一致的块设备映射：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0

    dedicated_devices:
      - /dev/nvme0n1
      - /dev/nvme0n1
```

2. 可选： 因为您必须在 overcloud 部署前设置 OSD 设备列表，因此可能无法识别和设置磁盘设备的 PCI 路径。在这种情况下，在内省期间为块设备收集 **/dev/disk/by-path/symlink** 数据。在以下示例中，运行第一个命令，从服务器 **b08-h03-r620-hci**.json 的 undercloud Object Storage 服务(swift)下载内省数据，并将数据保存到名为 **b08-h03-r620-hci.json** 的文件中。运行第二个命令来对"by-path"使用 grep。此命令的输出包含可用于识别磁盘的唯一 **/dev/disk/by-path** 值。

```
(undercloud) [stack@b08-h02-r620 ironic]$ openstack baremetal introspection data save
b08-h03-r620-hci | jq . > b08-h03-r620-hci.json
(undercloud) [stack@b08-h02-r620 ironic]$ grep by-path b08-h03-r620-hci.json
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
        "by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:1:0",
```

```
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:3:0",
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:4:0",
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:5:0",
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:6:0",
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:7:0",
"by_path": "/dev/disk/by-path/pci-0000:02:00.0-scsi-0:2:0:0",
```

有关存储设备的命名规则的更多信息，请参阅 *管理存储设备指南*中的持久性命名属性概述。

5.3.3. 在高级场景中配置 OSD

在环境文件中，您将列出要在 **CephAnsibleDisksConfig** 资源的 **devices** 变量中用于 OSD 的块设备。

当您在没有其他设备配置参数的情况下使用 **devices** 变量时，**ceph-volume lvm batch** 通过将更高的性能设备作为较慢的设备的 **block.db** 来自动优化 OSD 配置。

您可以使用以下步骤配置 设备 以避免在 **ceph-volume lvm batch** 模式下运行。

5.3.3.1. 使用 block.db 提高性能

使用 **block.db** 可以通过增加吞吐量并缩短响应时间来提高 Ceph Storage 集群的性能。 **block.db** 是一个数据库，它由数据片段和 BlueStore write-ahead 日志(WAL)组成。

流程

1. 在环境文件中添加以下内容：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
      - /dev/nvme0n1
      - /dev/sdc
      - /dev/sdd
      - /dev/nvme0n2
    osd_scenario: lvm
    osd_objectstore: bluestore
```

这将配置四个 OSD： **sda**、**sdb**、**sdc** 和 **sdd**。每个对都有自己的数据库：nv **em0n1** 和 **nvme0n2**。

注意

设备列表中的设备顺序非常重要。**列出驱动器，后跟 block.db 和 BlueStore WAL (DB-WAL)设备。在这个示例中，nvme0n1 是 sda 和 sdb 的 DB-WAL，nvme0n2 是 sdc 和 sdd 的 DB-WAL。如需更多信息，请参阅使用 BlueStore。**

2. **在部署 overcloud 时，使用 -e 选项包括部署命令中包含新内容的环境文件。**

5.3.3.2. 使用专用 write-ahead 日志(WAL)设备

您可以指定专用的 write-ahead 日志(WAL)设备。使用 devices、dedicated_devices 和 bluestore_wal_devices 意味着您可以将 OSD 的所有组件隔离到单独的设备，从而提高性能。

在以下示例中，另一个额外的字典 bluestore_wal_devices 隔离 NVMe devices nvme0n1 和 nvme0n2 上的 write-ahead 日志。

流程

1. 在环境文件中添加以下内容：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sda
      - /dev/sdb
    dedicated_devices:
      - /dev/sdx
      - /dev/sdy
    bluestore_wal_devices:
      - /dev/nvme0n1
      - /dev/nvme0n2
```

2. 在部署 overcloud 时，使用 -e 选项包括部署命令中包含新内容的环境文件。

5.3.3.3. 使用预先创建的 LVM 来提高控制

在以前的高级场景中，ceph-volume 使用不同类型的设备列表来为 OSD 创建逻辑卷。您还可以在 ceph-volume 运行前创建逻辑卷，然后将 ceph-volume 传递这些逻辑卷的 lvm_volumes 列表。虽然这需要您提前创建逻辑卷，但这意味着您有更精确的控制。由于 director 还负责硬件置备，因此您必须使用第一次引导脚本提前创建这些 LVM。

流程

1. 创建一个环境文件 /home/stack/templates/firstboot.yaml，它将您的 heat 模板注册为 OS::TripleO::NodeUserData 资源类型，并包含以下内容：

```
resource_registry:
  OS::TripleO::NodeUserData: /home/stack/templates/ceph-lvm.yaml
```

2. 创建一个环境文件 /home/stack/templates/ceph-lvm.yaml。添加类似以下示例的列表，其

中包含三个物理卷。如果您的设备列表较长，请根据您的要求扩展示例。

```
heat_template_version: 2014-10-16

description: >
  Extra hostname configuration

resources:
  userdata:
    type: OS::Heat::MultipartMime
    properties:
      parts:
      - config: {get_resource: ceph_lvm_config}

  ceph_lvm_config:
    type: OS::Heat::SoftwareConfig
    properties:
      config: |
        #!/bin/bash -x
        pvcreate /dev/sda
        vgcreate ceph_vg_hdd /dev/sda
        pvcreate /dev/sdb
        vgcreate ceph_vg_ssd /dev/sdb
        pvcreate /dev/nvme0n1
        vgcreate ceph_vg_nvme /dev/nvme0n1
        lvcreate -n ceph_lv_wal1 -L 50G ceph_vg_nvme
        lvcreate -n ceph_lv_db1 -L 500G ceph_vg_ssd
        lvcreate -n ceph_lv_data1 -L 5T ceph_vg_hdd
        lvs

outputs:
  OS::stack_id:
    value: {get_resource: userdata}
```

3. 通过以下方式使用 **lvm_volumes** 参数而不是 **devices** 列表。这假设已经创建了卷组和逻辑卷。在这种情况下，一个典型的用例是 **WAL** 和 **DB LV** 位于 **SSD** 上，数据 **LV** 位于 **HDD** 上：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    osd_objectstore: bluestore
    osd_scenario: lvm
    lvm_volumes:
      - data: ceph_lv_data1
        data_vg: ceph_vg_hdd
        db: ceph_lv_db1
        db_vg: ceph_vg_ssd
        wal: ceph_lv_wal1
        wal_vg: ceph_vg_nvme
```

4. 在部署 **overcloud** 时，使用 **-e** 选项包括部署命令中包含新内容的环境文件。

备注

只有在 WAL 设备位于性能优于 DB 设备的硬件上，才需要指定单独的 WAL 设备。通常，创建单独的 DB 设备就足够了，然后相同的分区用于 WAL 功能。

5.4. 为不同的 CEPH 池分配自定义属性

默认情况下，使用 director 创建的 Ceph Storage 池具有相同的放置组数量(pg_num 和 pgp_num)和大小。您可以使用 *第 5 章 自定义 Ceph Storage 集群* 中的任一方法全局覆盖这些设置。这样做会将相同的值应用到所有池。

使用 CephPools 参数，将不同的属性应用到每个 Ceph Storage 池或创建新的自定义池。

流程

1. 将 POOL 替换为您要配置的池的名称：

```
parameter_defaults:
  CephPools:
    - name: POOL
```

2. 通过执行以下操作之一配置放置组：

- 要手动覆盖默认设置，将 pg_num 设置为放置组数量：

```
parameter_defaults:
  CephPools:
    - name: POOL
      pg_num: 128
      application: rbd
```

- 或者，要自动扩展放置组，将 pg_autoscale_mode 设置为 True，并将 target_size_ratio 设置为相对于预期的 Ceph Storage 要求的百分比：

```
parameter_defaults:
  CephPools:
    - name: POOL
      pg_autoscale_mode: True
      target_size_ratio: PERCENTAGE
      application: rbd
```

用十进制替换 PERCENTAGE。例如，0.5 等于 50%。总计百分比必须等于 1.0 或 100 百分比。

例如，以下值只包括：

```
paramter_defaults:
  CephPools:
    - {"name": backups, "target_size_ratio": 0.1, "pg_autoscale_mode": True, "application": rbd}
    - {"name": volumes, "target_size_ratio": 0.5, "pg_autoscale_mode": True, "application": rbd}
    - {"name": vms,     "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application": rbd}
    - {"name": images,  "target_size_ratio": 0.2, "pg_autoscale_mode": True, "application": rbd}
```

有关更多信息，请参阅 *Red Hat Ceph Storage 安装指南*中的放置组自动缩放器。

3. 指定应用程序类型。

Compute、Block Storage 和 Image Storage 的应用类型是'rbd'。但是，根据您使用池的内容，您可以指定不同的应用程序类型。

例如，gnocchi 指标池的应用类型是 openstack_gnocchi。如需更多信息，*请参阅存储策略指南中的*启用应用程序。

注意

如果不使用 CephPools 参数，则 director 会自动设置适当的应用类型，但仅适用于默认的池列表。

4. 可选：添加名为 custompool 的池来创建一个自定义池，并设置特定于您的环境需求的参数：

```
parameter_defaults:
  CephPools:
    - name: custompool
      pg_num: 128
      application: rbd
```

提示

有关常见 Ceph 用例的典型池配置，请参阅 Ceph 放置组(PG)每个池计算器。此计算器通常用于生成手动配置 Ceph 池的命令。在本部署中，director 根据您的规格配置池。

警告

Red Hat Ceph Storage 3 (Luminous)对 OSD 可以具有的最大 PG 数量引入了一个硬性限制，默认为 200。不要覆盖超过 200 个参数。如果因为 Ceph PG 数量超过最大值，请调整每个池的 pg_num 来解决这个问题，而不是 mon_max_pg_per_osd。

5.5. 覆盖用于忽略 CEPH STORAGE 节点的参数

所有节点具有托管 Ceph OSD 的角色，如 CephStorage 或 ComputeHCI，使用 第 5.3 节 “映射 Ceph Storage 节点磁盘布局” 中创建的全局 devices 和 dedicated_devices 列表。这些列表假定所有这些服务器具有相同的硬件。如果有具有此硬件的服务器不相同，则必须使用特定于节点的磁盘配置更新不同设备和 dedicated_ devices 列表的详细信息。

注意

托管 Ceph OSD 的角色在 roles_data.yaml 文件中包括 OS::TripleO::Services::CephOSD 服务。

没有与其他节点相同的硬件的 Ceph Storage 节点可能会导致性能问题。在 Red Hat OpenStack Platform (RHOSP)环境中，标准节点和带有特定于节点覆盖的节点之间存在更多差异，这是可能的性能损失。

5.5.1. 特定于节点的磁盘配置

必须为没有相同硬件的服务配置 director。这称为特定于节点的磁盘配置。

您可以使用以下方法之一创建特定于节点的磁盘配置：

- 自动 ：您可以生成 JSON heat 环境文件，以自动创建特定于节点的磁盘配置。

- 手动：您可以更改节点磁盘布局，以创建特定于节点的磁盘配置。

5.5.1.1. 为 Ceph 设备生成 JSON heat 环境文件

您可以使用 /usr/share/openstack-tripleo-heat-templates/tools/make_ceph_disk_list.py 脚本从裸机置备服务(ironic)的内省数据自动创建有效的 JSON heat 环境文件。使用此 JSON 文件将特定于节点的磁盘配置传递给 director。

流程

1. 为您要部署的 Ceph 节点导出来自裸机置备服务的内省数据：

```
openstack baremetal introspection data save oc0-ceph-0 > ceph0.json
openstack baremetal introspection data save oc0-ceph-1 > ceph1.json
...
```

2. 将 实用程序复制到 undercloud 上 stack 用户的主目录，并使用它来生成 node_data_lookup.json 文件。

```
./make_ceph_disk_list.py -i ceph*.json -o node_data_lookup.json -k by_path
```

3. 将托管 Ceph OSD 的所有节点的 openstack baremetal introspection data save 命令的内省数据文件传递给 utility，因为您在部署过程中只能定义 NodeDataLookup。-i 选项可以采用类似 *.json 的表达式，或者将文件列表作为输入。

使用 -k 选项定义您要用来识别 OSD 磁盘的裸机置备磁盘数据结构的密钥。不要使用 name，因为它会生成类似 /dev/sdd 的设备文件，这在重启过程中可能并不总是指向同一设备。反之，请使用 by_path。如果没有指定 -k，则这是默认设置。

裸机置备服务保留系统中的其中一个可用磁盘作为根磁盘。实用程序始终从生成的设备列表中删除根磁盘。

4. 可选： 您可以使用 ./make_ceph_disk_list.py -help 查看其他可用选项。

5. 在部署 overcloud 时，将 node_data_lookup.json 文件包含与您环境相关的任何其他环境

文件：

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e node_data_lookup.json \
…
```

5.5.1.2. 更改 Ceph Storage 节点中的磁盘布局

重要

非同构 Ceph Storage 节点可能会导致性能问题。在 Red Hat OpenStack Platform (RHOSP)环境中，标准节点和带有特定于节点覆盖的节点之间存在更多差异，这是可能的性能损失。

要将特定于节点的磁盘配置传递给 director，您必须将 heat 环境文件（如 node-spec-overrides.yaml ）传递给 openstack overcloud deploy 命令，文件内容必须通过机器唯一 UUID 和本地变量的列表来标识各个服务器，以覆盖全局变量。

您可以为每个独立的服务器或裸机置备服务(ironic)数据库提取机器唯一的 UUID。

备注

在以下步骤中，您可以创建包含嵌入式有效 JSON 的有效 YAML 环境文件。您还可以使用 make_ceph_disk_list.py 生成完整的 JSON 文件，并将它传递到部署命令，就像它是 YAML 一样。如需更多信息，请参阅为 Ceph 设备生成 JSON heat 环境文件。

流程

1. **要找到单个服务器的 UUID，请登录到服务器并输入以下命令：**

```
$ dmidecode -s system-uuid
```

2. **要从裸机置备服务数据库中提取 UUID，请在 undercloud 上输入以下命令：**

```
$ openstack baremetal introspection data save NODE-ID | jq .extra.system.product.uuid
```

警告

如果 undercloud.conf 在 undercloud 安装或升级和内省之前没有 inspection_extras = true，则 machine-unique UUID 不在裸机置备服务数据库中。

重要

machine-unique UUID 不是裸机置备服务 UUID。

有效的 node-spec-overrides.yaml 文件可能类似如下：

```
parameter_defaults:
  NodeDataLookup: {"32E87B4C-C4A7-418E-865B-191684A6883B": {"devices":
["/dev/sdc"]}}
```

3. **前两行后面的所有行都必须是有效的 JSON。使用 jq 命令验证 JSON 是否有效。**

   a. **从文件中删除前两行(parameter_defaults: 和 NodeDataLookup:)。**

   b. **运行 cat node-spec-overrides.yaml | jq .**

4. **随着 node-spec-overrides.yaml 文件增长，您还可以使用 jq 命令来确保嵌入的 JSON 有效。例如，由于 devices 和 dedicated_devices 列表必须有相同的长度，因此在开始部署前，使用以下命令来验证它们的长度是否相同。在以下示例中，node-spec-c05-h17-h21-h25-6048r.yaml 在机架 c05 中有三个服务器，其中插槽 h17、h21 和 h25 缺失磁盘。**

```
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.devices | length'
33
30
33
(undercloud) [stack@b08-h02-r620 tht]$ cat node-spec-c05-h17-h21-h25-6048r.yaml | jq '.[] |
.dedicated_devices | length'
33
```

```
30
33
(undercloud) [stack@b08-h02-r620 tht]$
```

5. **在验证了 JSON 被验证后，这两个行使其成为有效的环境 YAML 文件 (parameter_defaults: 和 NodeDataLookup:)，并在部署中包括带有 -e 的行。在以下示例中，更新的 heat 环境文件对 Ceph 部署使用 NodeDataLookup。所有服务器都有一个设备列表，带有 35 磁盘的设备列表，但其中之一缺少磁盘。此环境文件只覆盖那个单一节点的默认设备列表，并为其提供必须使用的 34 磁盘列表，而不是全局列表。**

6. **验证 JSON 后，再添加这两行使其成为有效的环境 YAML 文件(parameter_defaults: 和 NodeDataLookup:)，并在部署命令中包括它并带有 -e。**

在以下示例中，更新的 heat 环境文件对 Ceph 部署使用 NodeDataLookup。所有服务器都有一个设备列表，带有 35 磁盘的设备列表，但其中之一缺少磁盘。此环境文件只覆盖那个单一节点的默认设备列表，并为该节点提供必须使用的 34 磁盘列表，而不是全局列表。

```
parameter_defaults:
  # c05-h01-6048r is missing scsi-0:2:35:0 (00000000-0000-0000-0000-0CC47A6EFD0C)
  NodeDataLookup: {
    "00000000-0000-0000-0000-0CC47A6EFD0C": {
      "devices": [
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:1:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:32:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:2:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:3:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:4:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:5:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:6:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:33:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:7:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:8:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:34:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:9:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:10:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:11:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:12:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:13:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:14:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:15:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:16:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:17:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:18:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:19:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:20:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:21:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:22:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:23:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:24:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:25:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:26:0",
```

```
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:27:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:28:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:29:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:30:0",
    "/dev/disk/by-path/pci-0000:03:00.0-scsi-0:2:31:0"
      ],
    "dedicated_devices": [
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:81:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1",
    "/dev/disk/by-path/pci-0000:84:00.0-nvme-1"
      ]
    }
}
```

5.5.2. 更改 BlueStore block.db 大小

BlueStore block.db 是数据片段和 BlueStore write-ahead 日志(WAL)的数据库。可以通过两种方法更改数据库大小。选择这些方法之一以更改大小。

5.5.2.1. 使用 ceph-volume 时更改 BlueStore block.db 大小

使用 ceph-volume 时，使用以下步骤覆盖 block.db 大小。当 osd_scenario: lvm. ceph-volume 会自动设置 block.db 大小。但是，您可以在高级场景中覆盖 block.db 大小。

以下示例使用 ceph-ansible 主机变量，而不是 Ceph 配置文件覆盖，因此使用的 block_db_size 传递给 ceph-volume 调用。

流程

1. 创建一个 JSON 环境文件，其内容类似以下示例，但根据您的要求替换值：

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "block_db_size": 3221225472
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "block_db_size": 3221225472
      }
    }
  }
}
```

2. 在部署 overcloud 时，包括与环境相关的任何其他环境文件的 JSON 文件：

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.5.2.2. 使用 ceph-disk时更改 BlueStore block.db 大小

使用 ceph-disk 时，使用以下步骤覆盖 block.db 大小。当 osd_scenario: non-collocated 或 osd_scenario: collocated 时使用 ceph-disk。

以下示例对特定节点使用 Ceph 配置覆盖来设置 blustore_block_db_size。使用 ceph-volume 时会忽略此 Ceph 配置选项，但 ceph-disk 使用此配置选项。

流程

1. 创建一个 JSON 环境文件，其内容类似以下示例，但根据您的要求替换值：

```
{
  "parameter_defaults": {
    "NodeDataLookup": {
      "32e87b4c-c4a7-41be-865b-191684a6883b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      },
      "ea6a84d6-cf89-4fe2-b7bd-869b3fe4dd6b": {
        "ceph_conf_overrides": {
          "osd": {
            "bluestore_block_db_size": 3221225472
          }
        }
      }
    }
  }
}
```

2. 在部署 overcloud 时，包括与环境相关的任何其他环境文件的 JSON 文件：

```
$ openstack overcloud deploy \
--templates \
…
-e <existing_overcloud_environment_files> \
-e <json_environment_file>  \
…
```

5.6. 为大型 CEPH 集群增加重启延迟

在部署期间，OSD 和 monitor 等 Ceph 服务已重启，部署也不会继续，直到服务再次运行为止。Ansible 等待 15 秒（延迟），并检查服务启动（重试）的 5 次。如果服务没有重启，部署将停止，以便操作员可以干预。

根据 Ceph 集群的大小，您可能需要增加重试或延迟值。这些参数的确切名称及其默认值如下：

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

流程

1. 更新 CephAnsibleExtraConfig 参数，以更改默认延迟和重试值：

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
    health_mon_check_delay: 20
    health_mon_check_retries: 10
```

本例使集群检查 30 次，在每次检查 Ceph OSD 的每个检查之间等待 40 秒，再检查 Ceph MON 的每个检查之间等待 10 秒。

2. 要纳入这些更改，请使用 -e 使用 openstack overcloud deploy 传递更新的 yaml 文件。

5.7. 覆盖 ANSIBLE 环境变量

Red Hat OpenStack Platform Workflow 服务(mistral)使用 Ansible 来配置 Ceph 存储，但您可以使用 Ansible 环境变量自定义 Ansible 环境。

流程

要覆盖 ANSIBLE_* 环境变量，请使用 CephAnsibleEnvironmentVariables heat 模板参数。

这个示例配置会增加 fork 的数量和 SSH 重试次数：

```
parameter_defaults:
  CephAnsibleEnvironmentVariables:
    ANSIBLE_SSH_RETRIES: '6'
    DEFAULT_FORKS: '35'
```

如需有关 Ansible 环境变量的更多信息，请参阅 Ansible 配置设置。

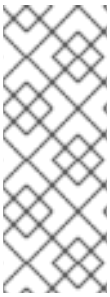
有关如何自定义 Ceph Storage 集群的更多信息，请参阅自定义 Ceph Storage 集群。

5.8. 启用 CEPH ON-WIRE 加密

从 Red Hat Ceph Storage 4 及更高版本开始，您可以通过引入 messenger version 2 协议，为网络上的所有 Ceph 流量启用加密。me senger v2 的安全模式设置加密 Ceph 守护进程和 Ceph 客户端之间

的通信，从而为您提供端到端加密。

该功能在此发行版本中作为*技术预览*提供，因此不享有红帽的全面支持。它只应用于测试，不应部署在生产环境中。有关技术预览功能的更多信息，请参阅覆盖范围详细信息。



注意

此功能虽然目前是一个技术预览，但用于 Red Hat OpenStack Platform (RHOSP)版本 16.1 及更新的版本。使用外部 Red Hat Ceph Storage 版本 4 的 RHOSP 版本 13 部署中不支持它。有关更多信息，请参阅 ***Red Hat Ceph Storage 架构指南***中的 Ceph 在线加密。

要在 RHOSP 上启用 Ceph on-wire 加密，请在环境覆盖文件中设置以下参数：

```
parameter_defaults:
  CephConfigOverrides:
    global:
      ms_cluster_mode: secure
      ms_service_mode: secure
      ms_client_mode: secure
```

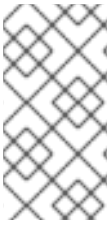
有关 Ceph 在线加密的更多信息，请参阅 架构指南中的 Ceph 在线加密。

第 6 章 使用 DIRECTOR 在 CEPH STORAGE 集群中定义不同工作负载的性能层

您可以使用 Red Hat OpenStack Platform (RHOSP) director 部署不同的 Red Hat Ceph Storage 性能层。您可以组合 Ceph CRUSH 规则和 CephPools director 参数，以使用设备类功能和构建不同的层来容纳具有不同性能要求的工作负载。例如，您可以为普通工作负载定义 HDD 类，以及一个仅通过 SSD 分发数据的 SSD 类，以实现高性能负载。在这种情况下，当您创建新的块存储卷时，您可以选择性能层，可以是 HDD 或 SSD。

WARNING

在现有环境中定义性能层可能会导致 Ceph 集群中大量数据移动。Ceph-ansible (director 在堆栈更新期间触发)没有逻辑来检查集群中是否已定义池，以及它是否包含数据。这意味着，在现有环境中定义性能层可能会危险，因为更改与池关联的默认 CRUSH 规则会导致数据移动。如果您需要帮助或建议添加或删除节点，请联系红帽支持。



注意

Ceph 自动检测磁盘类型，并根据 Linux 内核公开的硬件属性将其分配给对应的设备类 HDD、SSD 或 NVMe。但是，您还可以根据自己的需要自定义类别。

前提条件

- 对于新部署，Red Hat Ceph Storage (RHCS)版本 4.1 或更高版本。
- 对于现有部署，Red Hat Ceph Storage (RHCS)版本 4.2 或更高版本。

要部署不同的 Red Hat Ceph Storage 性能层，创建一个包含 CRUSH map 详细信息的新环境文件，然后在部署命令中包括它。

在以下步骤中，每个 Ceph Storage 节点包含三个 OSD，sdb 和 sdc 是旋转的磁盘，sdc 是 SSD。Ceph 会自动检测正确的磁盘类型。然后，您可以配置两个 CRUSH 规则 HDD 和 SSD，以映射到两个对应的设备类。HDD 规则是默认，适用于所有池，除非您使用不同的规则配置池。

最后，您创建一个名为 fastpool 的额外池，并将其映射到 SSD 规则。此池最终通过 Block Storage (cinder)后端公开。任何消耗此块存储后端的工作负载都由 SSD 支持，以获得快速性能。您可以使用它来进行数据或从卷引导。

6.1. 配置性能层

WARNING

在现有环境中定义性能层可能会导致 Ceph 集群中大量数据移动。Ceph-ansible (director 在堆栈更新期间触发)没有逻辑来检查集群中是否已定义池，以及它是否包含数据。这意味着，在现有环境中定义性能层可能会危险，因为更改与池关联的默认 CRUSH 规则会导致数据移动。如果您需要帮助或建议添加或删除节点，请联系红帽支持。

director 不会公开特定的参数来覆盖此功能，但您可以通过完成以下步骤来生成 ceph-ansible 预期的变量。

流程

1. 以 stack 用户身份登录 undercloud 节点。

2. 创建一个环境文件，如 /home/stack/templates/ceph-config.yaml，使其包含 Ceph 配置参数和设备类变量。或者，您还可以将以下配置添加到现有环境文件中。

3. 在环境文件中，使用 CephAnsibleDisksConfig 参数列出您要用作 Ceph OSD 的块设备：

```
CephAnsibleDisksConfig:
  devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
  osd_scenario: lvm
  osd_objectstore: bluestore
```

4. 可选：Ceph 会自动检测磁盘类型，并将其分配给对应的设备类。但是，您还可以使用 crush_device_class 属性来强制特定设备属于特定的类或创建自己的自定义类。以下示例包含具有指定类的相同 OSD 列表：

```
CephAnsibleDisksConfig:
  lvm_volumes:
    - data: '/dev/sdb'
      crush_device_class: 'hdd'
    - data: '/dev/sdc'
      crush_device_class: 'hdd'
    - data: '/dev/sdd'
      crush_device_class: 'ssd'
  osd_scenario: lvm
  osd_objectstore: bluestore
```

5. 添加 CephAnsibleExtraVars 参数。crush_rules 参数必须包含您定义或 Ceph 检测到的每

个类的规则。在创建新池时，如果没有指定规则，则会选择 Ceph 用作默认值的规则。

```
CephAnsibleExtraConfig:
  crush_rule_config: true
  create_crush_tree: true
  crush_rules:
   - name: HDD
     root: default
     type: host
     class: hdd
     default: true
   - name: SSD
     root: default
     type: host
     class: ssd
     default: false
```

6. 添加 CephPools 参数：

   - 使用 rule_name 参数指定不使用默认规则的每个池的层。在以下示例中，fastpool 池使用配置为快速层的 SSD 设备类来管理块存储卷。

   - 将 <appropriate_PG_num > 替换为适当的放置组数量(PG)。或者，使用放置组 auto-scaler 计算 Ceph 池的 PG 数量。

     有关更多信息，请参阅将自定义属性分配到不同的 Ceph 池。

   - 使用 CinderRbdExtraPools 参数将 fastpool 配置为块存储后端。

```
CephPools:
  - name: fastpool
    pg_num: <appropraiate_PG_num>
    rule_name: SSD
    application: rbd
CinderRbdExtraPools: fastpool
```

7. 使用以下示例来确保环境文件包含正确的值：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - '/dev/sdb'
      - '/dev/sdc'
```

```
        - '/dev/sdd'
    osd_scenario: lvm
    osd_objectstore: bluestore
CephAnsibleExtraConfig:
    crush_rule_config: true
    create_crush_tree: true
    crush_rules:
      - name: HDD
        root: default
        type: host
        class: hdd
        default: true
      - name: SSD
        root: default
        type: host
        class: ssd
        default: false
CinderRbdExtraPools: fastpool
CephPools:
    - name: fastpool
      pg_num: <appropriate_PG_num>
      rule_name: SSD
      application: rbd
```

8. ***在 openstack overcloud deploy 命令中包含新的环境文件。***

```
$ openstack overcloud deploy \
--templates \
…
-e <other_overcloud_environment_files> \
-e /home/stack/templates/ceph-config.yaml  \
…
```

将 <other_overcloud_environment_files > 替换为属于该部署的环境文件列表。

重要

如果将环境文件应用到现有的 Ceph 集群，则预先存在的 Ceph 池不会使用新规则进行更新。因此，您必须在部署完成后输入以下命令，将规则设置为指定的池。

```
$ ceph osd pool set <pool> crush_rule <rule>
```

- 将 <pool> 替换为您要将新规则应用到的池的名称。
- 将 <rule > 替换为您使用 crush_rules 参数指定的规则名称之一。
- 将 <appropriate_PG_num > 替换为适当的放置组数量或 target_size_ratio，并将 pg_autoscale_mode 设置为 true。

对于您使用这个命令更改的每个规则，更新现有条目，或者在现有模板中的 CephPools 参数中添加新条目：

```
CephPools:
  - name: <pool>
    pg_num: <appropriate_PG_num>
    rule_name: <rule>
    application: rbd
```

6.2. 将 BLOCK STORAGE (CINDER)类型映射到您的新 CEPH 池

WARNING

在现有环境中定义性能层可能会导致 Ceph 集群中大量数据移动。Ceph-ansible (director 在堆栈更新期间触发)没有逻辑来检查集群中是否已定义池，以及它是否包含数据。这意味着，在现有环境中定义性能层可能会危险，因为更改与池关联的默认 CRUSH 规则会导致数据移动。如果您需要帮助或建议添加或删除节点，请联系红帽支持。

完成配置步骤后，使用 Block Storage (cinder)创建映射到您创建的 fastpool 层的类型，使性能层可供 RHOSP 租户使用。

流程

1. 以 stack 用户身份登录 undercloud 节点。

2. 获取 overcloudrc 文件：

```
$ source overcloudrc
```

3. 检查 Block Storage 卷现有的类型：

```
$ cinder type-list
```

4. 创建新的块存储卷 fast_tier：

```
$ cinder type-create fast_tier
```

5. 检查是否创建了 Block Storage 类型：

```
$ cinder type-list
```

6. 当 fast_tier Block Storage 类型可用时，将您创建的新层的 fastpool 设置为块存储卷后端：

```
$ cinder type-key fast_tier set volume_backend_name=tripleo_ceph_fastpool
```

7. 使用新层来创建新卷：

```
$ cinder create 1 --volume-type fast_tier --name fastdisk
```

6.3. 验证 CRUSH 规则已创建，并且您的池已设置为正确的 CRUSH 规则

WARNING

在现有环境中定义性能层可能会导致 Ceph 集群中大量数据移动。Ceph-ansible (director 在堆栈更新期间触发)没有逻辑来检查集群中是否已定义池，以及它是否包含数据。这意味着，在现有环境中定义性能层可能会危险，因为更改与池关联的默认 CRUSH 规则会导致数据移动。如果您需要帮助或建议添加或删除节点，请联系红帽支持。

流程

1. 以 heat-admin 用户身份登录 overcloud Controller 节点。

2. 要验证您的 OSD 层是否已成功设置，请输入以下命令。将 <controller_hostname > 替换为主机控制器节点的名称。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd tree
```

3. 在生成的树视图中，验证 CLASS 列是否显示您设置的每个 OSD 的正确设备类。

4. 另外，使用以下命令验证 OSD 是否已正确分配给设备类。将 <controller_hostname > 替换为主机控制器节点的名称。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd crush tree --show-shadow
```

5. 将生成的层次结构与以下命令的结果进行比较，以确保每个规则都应用相同的值。

   - 将 <controller_hostname > 替换为主机控制器节点的名称。
   - 将 <rule_name > 替换为您要检查的规则的名称。

```
$ sudo podman exec <controller_hostname> ceph osd crush rule dump <rule_name>
```

6. 根据部署期间使用的 crush_rules 参数，验证您创建的规则名称和 ID 是否正确。将 <controller_hostname > 替换为主机控制器节点的名称。

```
$ sudo podman exec -it ceph-mon-<controller_hostname>  ceph osd crush rule dump | grep -E "rule_(id|name)"
```

7. 验证 Ceph 池已绑定到在第 3 步中获得的正确 CRUSH 规则 ID。将 <controller_hostname > 替换为主机控制器节点的名称。

```
$ sudo podman exec -it ceph-mon-<controller_hostname> ceph osd dump | grep pool
```

8. 对于每个池，确保规则 ID 与您所期望的规则名称匹配。

第 7 章 创建 OVERCLOUD

当自定义环境文件就绪时，您可以指定每个角色使用的类别和节点，然后执行部署。以下小节更详细地说明了这两个步骤。

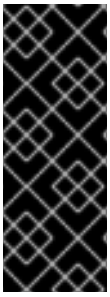
7.1. 为角色分配节点和类别

规划 overcloud 部署涉及指定要分配给各个角色的节点数量和哪些类型。与所有 Heat 模板参数一样，这些角色规格在环境文件的 parameter_defaults 部分中声明（本例中为 ~/templates/storage-config.yaml）。

为此，请使用以下参数：

表 7.1. Overcloud 节点的角色和类别

Heat 模板参数	Description
ControllerCount	扩展的 Controller 节点数量
OvercloudControlFlavor	Controller 节点使用的 flavor (**控制**)
ComputeCount	扩展的 Compute 节点数量
OvercloudComputeFlavor	Compute 节点使用的 flavor (**计算**)
CephStorageCount	扩展的 Ceph 存储(OSD)节点数量
OvercloudCephStorageFlavor	用于 Ceph Storage (OSD)节点(**ceph-storage**)的类别。
CephMonCount	扩展的专用 Ceph MON 节点数量
OvercloudCephMonFlavor	用于专用 Ceph MON 节点(**ceph-mon**)的 flavor
CephMdsCount	扩展的专用 Ceph MDS 节点数量
OvercloudCephMdsFlavor	用于专用 Ceph MDS 节点(**ceph-mds**)的类别。



重要

CephMonCount、CephMdsCount、OvercloudCephMonFlavor 和 OvercloudCephMdsFlavor 参数（以及 ceph-mon 和 ceph-mds 类别）只有在您创建了自定义 CephMON 和 CephMds 角色时，才有效，如 第 3 章 在专用节点上部署 Ceph 服务 所述。

例如，要将 overcloud 配置为为每个角色(Controller、Compute、Ceph-Storage 和 CephMon)部署三个节点，请将以下内容添加到您的 parameter_defaults 中：

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
```

注意

如需更完整的 Heat 模板参数列表，请参阅 Director 安装和使用指南中的使用 CLI 工具创建 Overcloud。

7.2. 启动 OVERCLOUD 部署

注意

在 undercloud 安装过程中，在 undercloud.conf 文件中设置 generate_service_certificate=false。否则，在部署 overcloud 时您必须注入信任定位符，如 高级 Overcloud 自定义指南中的 Overcloud Public Endpoints 上启用 SSL/TLS 所述。

备注

如果要在 overcloud 部署期间添加 Ceph 控制面板，请参阅 第 8 章 将 Red Hat Ceph Storage Dashboard 添加到 overcloud 部署中。

创建 overcloud 需要 openstack overcloud deploy 命令的额外参数。例如：

```
$ openstack overcloud deploy --templates -r /home/stack/templates/roles_data_custom.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml \
  -e /home/stack/templates/storage-config.yaml \
  -e /home/stack/templates/ceph-config.yaml \
  --ntp-server pool.ntp.org
```

以上命令使用以下选项：

- *--templates - 从默认的 Heat 模板集合创建 Overcloud （即 /usr/share/openstack-tripleo-heat-templates/）。*
- *-r /home/stack/templates/roles_data_custom.yaml - 指定来自 第 3 章 在专用节点上部署 Ceph 服务 的自定义角色，它为 Ceph MON 或 Ceph MDS 服务添加自定义角色。这些角色允许在专用节点上安装任一服务。*
- *-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml - 设置 director 以创建 Ceph 集群。特别是，此环境文件将部署具有 容器化 Ceph Storage 节点的 Ceph 集群。*
- *-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-rgw.yaml - 启用 Ceph 对象网关，如 第 4.2 节 “启用 Ceph 对象网关” 所述。*
- *-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-mds.yaml - 启用 Ceph 元数据服务器，如 第 4.1 节 “启用 Ceph 元数据服务器” 所述。*
- *-e /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml - 启用块存储备份服务(cinder-backup)，如 第 4.4 节 “将备份服务配置为使用 Ceph” 所述。*
- *-e /home/stack/templates/storage-config.yaml - 添加包含自定义 Ceph Storage 配置的环境文件。*
- *-e /home/stack/templates/ceph-config.yaml - 添加包含自定义 Ceph 集群设置的环境文件，如 第 5 章 自定义 Ceph Storage 集群 所述。*

- **--ntp-server pool.ntp.org** - 设置 NTP 服务器。

提示

您还可以使用 回答文件来 调用所有模板和环境文件。例如，您可以使用以下命令部署相同的 **overcloud**：

```
$ openstack overcloud deploy -r /home/stack/templates/roles_data_custom.yaml \
  --answers-file /home/stack/templates/answers.yaml --ntp-server pool.ntp.org
```

在这种情况下，回答文件 **/home/stack/templates/answers.yaml** 包含：

```
templates: /usr/share/openstack-tripleo-heat-templates/
environments:
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-rgw.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/ceph-mds.yaml
  - /usr/share/openstack-tripleo-heat-templates/environments/cinder-backup.yaml
  - /home/stack/templates/storage-config.yaml
  - /home/stack/templates/ceph-config.yaml
```

如需了解更多详细信息，请参阅 **overcloud 部署中包含环境文件**。

如需完整的选项列表，请输入：

```
$ openstack help overcloud deploy
```

如需更多信息，请参阅 **Director 安装和使用指南中的使用 CLI 工具配置基本 overcloud**。

overcloud 创建过程开始，**director** 置备节点。这个过程需要一些时间来完成。要查看 **overcloud** 创建的状态，请以 **stack** 用户身份打开一个单独的终端并输入以下命令：

```
$ source ~/stackrc
$ openstack stack list --nested
```

7.2.1. 限制运行 ceph-ansible 的节点

您可以通过限制 **ceph-ansible** 运行的节点来减少部署更新时间。当 **Red Hat OpenStack Platform**

(RHOSP)使用 config-download 配置 Ceph 时，您可以使用 --limit 选项指定节点列表，而不是在整个部署中运行 config-download 和 ceph-ansible。例如，作为扩展 overcloud 或替换失败的磁盘的一部分，此功能很有用。在这些情况下，部署只能在您添加到环境中的新节点上运行。

在故障磁盘替换中使用 --limit 的示例

在以下示例中，Ceph 存储节点 oc0-cephstorage-0 的磁盘故障，以便它收到新的工厂干净磁盘。Ansible 需要在 oc0-cephstorage-0 节点上运行，以便新磁盘可以用作 OSD，但不需要在所有其他 Ceph 存储节点上运行。将示例环境文件和节点名称替换为适合您的环境。

流程

1. *以 stack 用户身份登录 undercloud 节点，并提供 stackrc 凭证文件：*

```
# source stackrc
```

2. *完成以下步骤之一，以便使用新磁盘来启动缺少的 OSD。*

- *运行堆栈更新并包含 --limit 选项，以指定您希望 ceph-ansible 运行的节点：*

```
$ openstack overcloud deploy --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e ~/my-ceph-settings.yaml \
  -e <other-environment_files> \
  --limit oc0-controller-0:oc0-controller-2:oc0-controller-1:oc0-cephstorage-0:undercloud
```

在本例中，包含 Controller，因为 Ceph mons 需要 Ansible 更改其 OSD 定义。

- *如果 config-download 生成一个 ansible-playbook-command.sh 脚本，您也可以使用 --limit 选项运行脚本，以将指定节点传递给 ceph-ansible ：*

```
./ansible-playbook-command.sh --limit oc0-controller-0:oc0-controller-2:oc0-controller-
1:oc0-cephstorage-0:undercloud
```

警告

您必须始终将 undercloud 包含在限制列表中，否则在使用 --limit 时无法执行 ceph-ansible。这是必要的，因为 ceph-ansible 执行通过

external_deploy_steps_tasks playbook 进行，该 playbook 仅在 undercloud 上运行。

第 8 章 将 RED HAT CEPH STORAGE DASHBOARD 添加到 OVERCLOUD 部署中

Red Hat Ceph Storage Dashboard 默认是禁用的，但您可以使用 Red Hat OpenStack Platform director 在 overcloud 中启用它。Ceph 控制面板是基于 Web 的内置 Ceph 管理和监控应用，用于管理集群中各种方面和对象。Red Hat Ceph Storage Dashboard 包括以下组件：

- Ceph Dashboard manager 模块提供用户界面并嵌入平台前端 Grafana。
- Prometheus，监控插件。
- Alertmanager 将警报发送到仪表板。
- 节点导出器将集群数据导出到仪表板。

备注

Ceph Storage 4.1 或更高版本支持此功能。有关如何确定系统上安装的 Ceph Storage 版本的更多信息，请参阅 Red Hat Ceph Storage 发行版本以及对应的 Ceph 软件包版本。

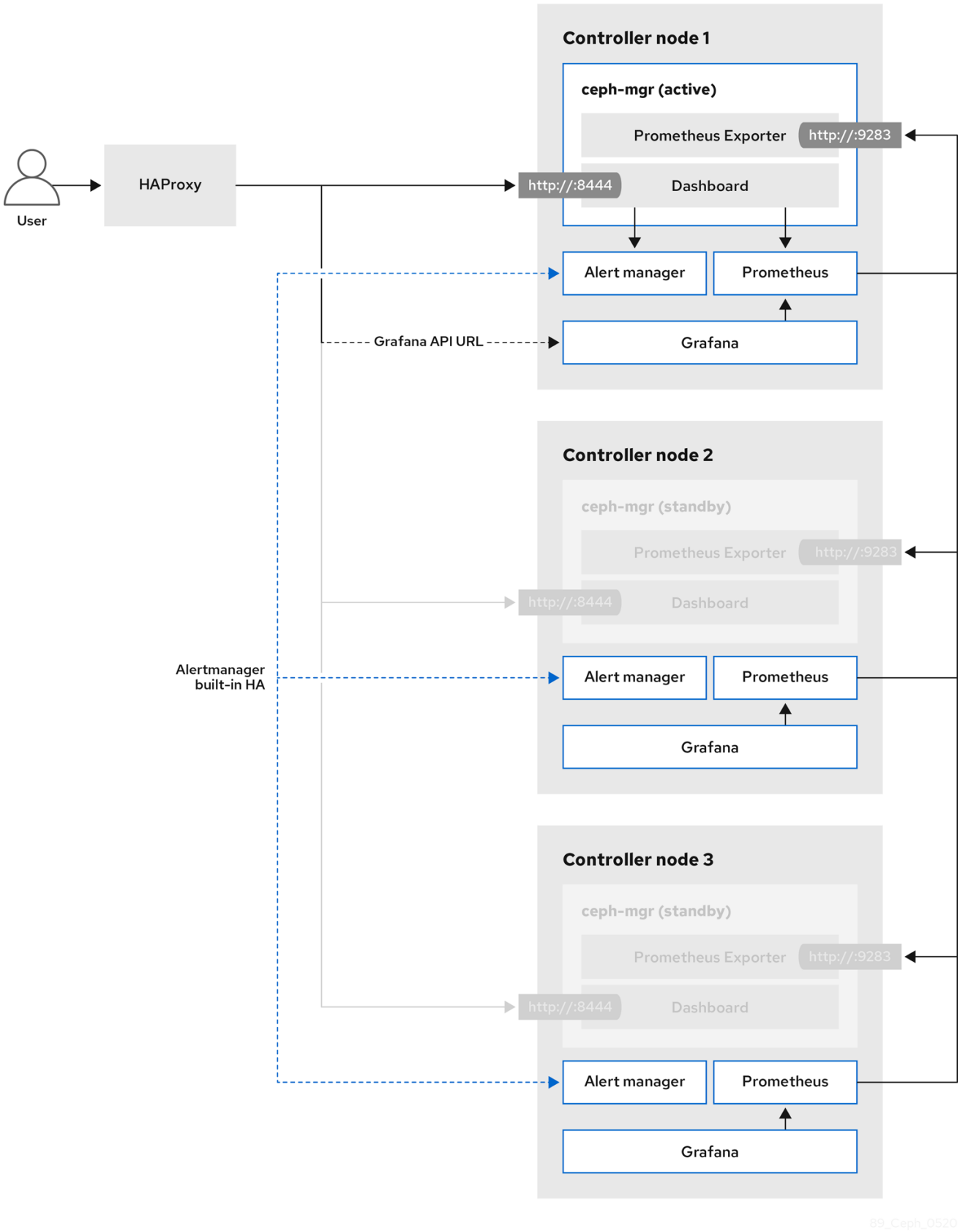
备注

Red Hat Ceph Storage Dashboard 始终与其他 Ceph 管理器组件位于同一个节点上。

备注

如果要在初始 overcloud 部署过程中添加 Ceph 仪表板，请在 第 7.2 节 “启动 overcloud 部署” 中部署初始 overcloud 前完成本章中的步骤。

下图显示了 Red Hat OpenStack Platform 上 Ceph 控制面板的架构：



有关控制面板及其功能及限制的更多信息，请参阅 Red Hat Ceph Storage Dashboard 指南中的控制面板功能。

TLS 在任何地方使用 Ceph 仪表板

Dashboard 前端与 TLS 完全集成。您可以在任何地方启用 TLS，为您提供所需的环境文件，并将其包含在 overcloud deploy 命令中。这会在 overcloud 部署期间触发 Grafana 和 Ceph 控制面板和生成的证书和密钥文件 的证书请求。有关如何为控制面板和其他 openstack 服务启用 TLS 的说明和更多信息，请参阅高级 Overcloud 自定义指南中的 以下位置：

- 在 Overcloud 公共端点 上启用 SSL/TLS.
- 使用身份管理在内部和公共端点中启用 SSL/TLS。

备注

访问 Ceph 控制面板的端口在 TLS-everywhere 上下文中保持不变。

8.1. 为 CEPH 仪表板包含所需的容器

在将 Ceph 控制面板模板添加到 overcloud 之前，您必须使用 containers-prepare-parameter.yaml 文件包括必要的容器。要生成 containers-prepare-parameter.yaml 文件以准备您的容器镜像，请完成以下步骤：

流程

1. 以 stack 用户身份登录 undercloud 主机。
2. 生成默认的容器镜像准备文件：

```
$ sudo openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

3. 编辑 containers-prepare-parameter.yaml 文件并进行修改以符合您的要求。以下 containers-prepare-parameter.yaml 文件示例包含与 Dashboard 服务相关的镜像位置和标签，如 Grafana、Prometheus、Alertmanager 和 Node Exporter。根据您的具体情况编辑值：

```
parameter_defaults:
    ContainerImagePrepare:
    - push_destination: true
        set:
            ceph_alertmanager_image: ose-prometheus-alertmanager
            ceph_alertmanager_namespace: registry.redhat.io/openshift4
```

```
    ceph_alertmanager_tag: v4.1
    ceph_grafana_image: rhceph-4-dashboard-rhel8
    ceph_grafana_namespace: registry.redhat.io/rhceph
    ceph_grafana_tag: 4
    ceph_image: rhceph-4-rhel8
    ceph_namespace: registry.redhat.io/rhceph
    ceph_node_exporter_image: ose-prometheus-node-exporter
    ceph_node_exporter_namespace: registry.redhat.io/openshift4
    ceph_node_exporter_tag: v4.1
    ceph_prometheus_image: ose-prometheus
    ceph_prometheus_namespace: registry.redhat.io/openshift4
    ceph_prometheus_tag: v4.1
    ceph_tag: latest
```

有关使用 containers-prepare-parameter.yaml 文件的 registry 和镜像配置的更多信息，请参阅过渡到 Containerized Services 指南中的容器镜像准备参数。

8.2. 部署 CEPH 仪表板

备注

如果要使用可组合网络部署 Ceph 仪表板，请参阅 第 8.3 节 “使用可组合网络部署 Ceph 仪表板”

备注

Ceph 控制面板 admin 用户角色默认设置为只读模式。要更改 Ceph 控制面板 admin 默认模式，请参阅 第 8.4 节 “更改默认权限”。

流程

1. 以 stack 用户身份登录 undercloud 节点。

2. 在 openstack overcloud deploy 命令中包含以下环境文件，以及属于部署的所有环境文件：

```
$ openstack overcloud deploy \
 --templates \
 -e <overcloud_environment_files> \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
 -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-dashboard.yaml
```

将 <overcloud_environment_files > 替换为属于该部署的环境文件列表。

结果

生成的部署包含 grafana、prometheus、alertmanager 和 node-exporter 容器的外部堆栈。Ceph Dashboard manager 模块是此堆栈的后端，它会嵌入 grafana 布局，以便为最终用户提供 ceph 集群特定的指标。

8.3. 使用可组合网络部署 CEPH 仪表板

您可以在可组合网络中部署 Ceph 控制面板，而不是在默认的 Provisioning 网络中部署。这消除了在 Provisioning 网络上公开 Ceph 控制面板服务的需求。在可组合网络中部署仪表板时，您还可以实施单独的授权配置集。

您必须在部署前选择要使用的网络，因为只能在首次部署 overcloud 时，才会将仪表板应用到新网络。您不能将仪表板应用到现有的外部网络，或重复使用 Provisioning 网络以外的一个现有网络。在部署前，使用以下步骤选择可组合网络。

流程

1. 以 stack 用户身份登录 undercloud。

2. 生成特定于 Controller 的角色，使其包含 Dashboard 可组合网络：

```
$ openstack overcloud roles generate -o /home/stack/roles_data_dashboard.yaml
ControllerStorageDashboard Compute BlockStorage ObjectStorage CephStorage
```

结果

- 在定义为命令输出的 roles_data.yaml 中生成新的 ControllerStorageDashboard 角色。在使用 overcloud deploy 命令时，您必须将此文件包含在模板列表中。

  注： ControllerStorageDashboard 角色不包含 CephNFS 或 network_data_dashboard.yaml。

- director 提供了一个网络环境文件，它定义了可组合网络。此文件的默认位置为 /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml。在使用 overcloud deploy 命令时，您必须将此文件包含在 overcloud 模板列表中。

3. 在 openstack overcloud deploy 命令中包含以下环境文件，以及属于部署的所有环境文件：

```
$ openstack overcloud deploy \
  --templates \
  -r /home/stack/roles_data.yaml \
  -n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
  -e <overcloud_environment_files> \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
dashboard.yaml
```

将 <overcloud_environment_files > 替换为属于该部署的环境文件列表。

结果

生成的部署包含 grafana、prometheus、alertmanager 和 node-exporter 容器的外部堆栈。Ceph 控制面板管理器模块是此堆栈的后端，它嵌入了 grafana 布局，以便为最终用户提供 Ceph 集群特定的指标。

8.4. 更改默认权限

Ceph 控制面板 admin 用户角色默认设置为只读模式，以安全监控 Ceph 集群。要允许管理员用户提升了特权，以便使用 CephDashboardAdminRO 参数更改 Ceph 集群的元素，您可以使用 CephDashboardAdminRO 参数来更改默认的 admin 权限。

警告

具有完整权限的用户可能会改变 director 配置的集群的元素。这可能会导致运行堆栈更新时与 director 配置的选项冲突。为避免这个问题，请不要使用 Ceph 控制面板更改 director 配置的选项，如 Ceph OSP 池属性。

流程

1. **以 stack 用户的身份登录 undercloud。**

2. **创建以下 ceph_dashboard_admin.yaml 环境文件：**

```
parameter_defaults:
  CephDashboardAdminRO: false
```

3. **运行 overcloud deploy 命令以更新现有堆栈，并包括您使用现有部署一部分的所有其他环境文件创建的环境文件：**

```
$ openstack overcloud deploy \
--templates \
-e <existing_overcloud_environment_files> \
-e ceph_dashboard_admin.yml
```

使用 **<existing_overcloud_environment_files>** 属于您现有部署的环境文件列表替换。

8.5. 访问 CEPH 仪表板

要测试 **Ceph** 仪表板是否正确运行，请完成以下验证步骤来访问它，并检查它从 **Ceph** 集群显示的数据是否正确。

流程

1. 以 **stack** 用户身份登录 **undercloud** 节点。

2. 检索仪表板 **admin** 登录凭证：

```
[stack@undercloud ~]$ grep dashboard_admin_password /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

3. 检索 **VIP** 地址以访问 **Ceph** 仪表板：

```
[stack@undercloud-0 ~]$ grep dashboard_frontend_vip /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml
```

4. 使用 **Web** 浏览器指向前端 **VIP** 并访问控制面板。**director** 在 **provisioning** 网络上配置并公开控制面板，因此您可以使用您检索到的 **VIP** 在 **TCP** 端口 **8444** 上直接访问仪表板。确保满足以下条件：

   - **Web** 客户端主机连接到 **provisioning** 网络的第 **2** 层。
   - 调配网络已正确路由或代理，可以从 **Web** 客户端主机访问。如果没有满足这些条件，您仍然可以打开 **SSH** 隧道来访问 **overcloud** 上的 **Dashboard VIP**：

```
client_host$ ssh -L 8444:<dashboard_vip>:8444 stack@<your undercloud>
```

将 <dashboard_vip> 替换为您检索到的 control plane VIP 的 IP 地址。

5. 要访问仪表板，使用浏览器访问 http://localhost:8444，并使用以下详情进行登录：

- ceph-ansible 创建的默认用户： admin。
- /var/lib/mistral/overcloud/ceph-ansible/group_vars/all.yml 中的密码。

结果

- 您可以访问 Ceph 控制面板。
- 控制面板显示的数字和图形反映了 CLI 命令 ceph -s 返回的同一集群状态。

有关 Red Hat Ceph Storage 仪表板的更多信息，请参阅 Red Hat Ceph Storage 管理指南

第 9 章 POST-DEPLOYMENT

以下小节描述了几个用于管理 Ceph 集群的部署后操作。

9.1. 访问 OVERCLOUD

director 会生成脚本来配置和帮助认证 undercloud 与 overcloud 的交互。director 将此文件 (overcloudrc)保存到 stack 用户的主目录中。运行以下命令来使用此文件：

```
$ source ~/overcloudrc
```

这会加载必要的环境变量，以便从 undercloud CLI 与 overcloud 交互。要返回与 undercloud 进行交互的状态，请运行以下命令：

```
$ source ~/stackrc
```

9.2. 监控 CEPH STORAGE 节点

创建 overcloud 后，检查 Ceph Storage 集群的状态，以确保它正常工作。

流程

1. 以 heat-admin 用户身份登录 Controller 节点：

```
$ nova list
$ ssh heat-admin@192.168.0.25
```

2. 检查集群的健康状况：

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph health
```

如果集群没有问题，该命令将报告回 HEALTH_OK。这意味着集群可以安全地使用。

3. 登录运行 Ceph 监控服务的 overcloud 节点，并检查集群中的所有 OSD 的状态：

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd tree
```

4. *检查 **Ceph Monitor 仲裁的状态：***

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph quorum_status
```

这显示了参与仲裁的监控器，以及哪个是领导的。

5. *验证所有 **Ceph OSD 都在运行：***

```
$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd stat
```

如需有关监控 Ceph Storage 集群的更多信息，请参阅 Red Hat Ceph Storage Administration Guide 中的 Monitoring。

第 10 章 重新引导环境

在需要重新引导环境时，可能会出现一个情况。例如，当您可能需要修改物理服务器时，或者可能需要从电源中断中恢复。在这种情况下，务必要确保 Ceph Storage 节点正确引导。

确保按照以下顺序引导节点：

- 首先引导所有 Ceph 监控节点 - 这样可确保 Ceph Monitor 服务在高可用性集群中处于活跃状态。默认情况下，Ceph Monitor 服务安装在 Controller 节点上。如果 Ceph Monitor 与自定义角色中的 Controller 分开，请确保此自定义 Ceph monitor 角色处于活动状态。
- 引导所有 Ceph Storage 节点 - 这样可确保 Ceph OSD 集群可以连接到 Controller 节点上的活跃 Ceph monitor 集群。

10.1. 重新引导 CEPH STORAGE (OSD) 集群

完成以下步骤以重新引导 Ceph Storage (OSD) 节点集群。

先决条件

- 在运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点上，检查 Red Hat Ceph Storage 集群是否正常运行，并且 pg 状态是 active+clean ：

```
$ sudo podman exec -it ceph-mon-controller-0 ceph -s
```

如果 Ceph 集群处于健康状态，它将返回 HEALTH_OK 状态。

如果 Ceph 集群状态不健康，它将返回 HEALTH_WARN 或 HEALTH_ERR 状态。有关故障排除指南，请参阅 Red Hat Ceph Storage 4 故障排除指南。

步骤

1. 登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并临时禁用 Ceph Storage 集群重新平衡：

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
```

注意

如果您有多堆栈或分布式计算节点(DCN)架构，则必须在设置 noout 和 norebalance 标志时指定集群名称。例如： sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster <cluster_name>

2. 选择第一个要重新引导的 Ceph Storage 节点并登录到该节点。

3. 重新引导节点：

```
$ sudo reboot
```

4. 稍等片刻，直到节点启动。

5. 登录到节点，并检查集群的状态：

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

确认 pgmap 报告的所有 pgs 的状态是否都正常 (active+clean)。

6. 注销节点，重新引导下一个节点，并检查其状态。重复此流程，直到您已重新引导所有 Ceph 存储节点。

7. 完成后，登录到运行 ceph-mon 服务的 Ceph Monitor 或 Controller 节点，并重新启用集群重新平衡：

```
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
$ sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
```

注意

如果您有多堆栈或分布式计算节点(DCN)架构，您必须在取消设置 noout 和 norebalance 标志时指定集群名称。例如： sudo podman exec -it ceph-mon-controller-0 ceph osd set noout --cluster <cluster_name>

8. 执行最后的状态检查，确认集群报告 HEALTH_OK：

```
$ sudo podman exec -it ceph-mon-controller-0 ceph status
```

如果所有 overcloud 节点同时引导时都启动，Ceph OSD 服务可能无法在 Ceph Storage 节点上正确启动。在这种情况下，重启 Ceph Storage OSD，以便它们能够连接到 Ceph Monitor 服务。

使用以下命令验证 Ceph Storage 节点集群的 HEALTH_OK 状态：

```
$ sudo ceph status
```

第 11 章 扩展 CEPH STORAGE 集群

11.1. 扩展 CEPH STORAGE 集群

您可以使用您需要的 Ceph Storage 节点数量重新运行 overcloud 中的 Ceph Storage 节点数量，从而扩展 overcloud 中的 Ceph Storage 节点数量。

在执行此操作前，请确保有足够的节点进行更新的部署。这些节点必须注册到 director，并相应地标记。

注册新的 Ceph Storage 节点

要使用 director 注册新的 Ceph 存储节点，请完成以下步骤。

流程

1. 以 stack 用户身份登录 undercloud，并初始化 director 配置：

```
$ source ~/stackrc
```

2. 在新节点定义模板中定义新节点的硬件和电源管理详情，例如 instackenv-scale.json。

3. 将此文件导入到 director：

```
$ openstack overcloud node import ~/instackenv-scale.json
```

导入节点定义模板会将此处定义的每个节点注册到 director。

4. 将内核和 ramdisk 镜像分配给所有节点：

```
$ openstack overcloud node configure
```

注意

有关注册新节点的详情，请参考 第 2.2 节 “注册节点”。

手动标记新节点

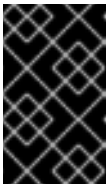
注册每个节点后，您必须检查硬件并将节点标记到特定的配置集中。使用 profile 标签将节点与类别匹配，然后将类别分配给部署角色。

流程

1. *触发硬件内省以检索每个节点的硬件属性：*

```
$ openstack overcloud node introspect --all-manageable --provide
```

- *--all-manageable 选项仅内省处于受管状态的节点。在此示例中，所有节点都处于受管理状态。*
- *--provide 选项会在内省后将所有节点重置 为活动状态。*



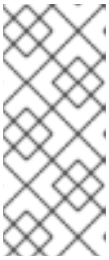
重要

确保此过程成功完成。它可能需要 15 分钟来检查这些裸机节点。

2. *检索节点列表来识别它们的 UUID：*

```
$ openstack baremetal node list
```

3. *在每个节点的 properties/capabilities 参数中添加 profile 选项，来手动将节点标记到特定的配置集。添加 profile 选项会将节点标记为相关的配置集。*



注意

作为手动标记的替代选择，请使用 Automated Health Check (AHC)工具根据基准测试数据自动标记更多节点。例如，以下命令使用 ceph-storage 配置集标记三个额外的节点：

```
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
551d81f5-4df2-4e0f-93da-6c5de0b868f7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
```

```
5e735154-bd6b-42dd-9cc2-b6195c4196d7
$ openstack baremetal node set --property capabilities='profile:baremetal,boot_option:local'
1a2b090c-299d-4c20-a25d-57dd21a7085b
```

提示

如果您标记并注册的节点使用多个磁盘，您可以将 director 设置为在每个节点上使用特定的根磁盘。更多信息请参阅 第 2.5 节 “为多磁盘集群定义根磁盘”。

使用额外的 Ceph Storage 节点重新部署 overcloud

在注册并标记新节点后，您可以通过重新部署 overcloud 来扩展 Ceph Storage 节点的数量。

流程

1. 在重新部署 overcloud 之前，在环境文件的 parameter_defaults 中设置 CephStorageCount 参数，本例中为 ~/templates/storage-config.yaml。在 第 7.1 节 “为角色分配节点和类别” 中，overcloud 配置为部署具有三个 Ceph Storage 节点。以下示例将 overcloud 扩展到 6 个节点：

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 6
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```

2. 重新部署 overcloud。overcloud 现在有六个 Ceph Storage 节点，而不是三个。

11.2. 缩减并替换 CEPH STORAGE 节点

在某些情况下，您可能需要缩减 Ceph 集群，甚至替换 Ceph Storage 节点，例如，如果 Ceph Storage 节点有故障。在这两种情况下，您必须禁用并重新平衡您要从 overcloud 中删除的任何 Ceph Storage 节点，以避免数据丢失。

注意

此流程使用 Red Hat Ceph Storage 管理指南中的步骤来手动删除 Ceph Storage 节点。有关手动删除 Ceph Storage 节点的更多信息，请参阅 启动、停止和重启容器中运行的 Ceph 守护进程，以及使用命令行界面删除 Ceph OSD。

流程

1. 以 heat-admin 用户身份登录 Controller 节点。director stack 用户具有访问 heat-admin 用户的 SSH 密钥。

2. 列出 OSD 树，并查找节点的 OSD。例如，您要删除的节点可能包含以下 OSD：

```
-2 0.09998     host overcloud-cephstorage-0
0 0.04999        osd.0                 up  1.00000          1.00000
1 0.04999        osd.1                 up  1.00000          1.00000
```

3. 禁用 Ceph Storage 节点上的 OSD。在本例中，OSD ID 为 0 和 1。

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd out 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph osd out 1
```

4. Ceph Storage 集群开始重新平衡。等待此过程完成。使用以下命令跟踪状态：

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph -w
```

5. 在 Ceph 集群完成重新平衡后，以 heat-admin 用户身份登录到您要删除的 Ceph Storage 节点，本例中为 overcloud-cephstorage-0，并停止并禁用该节点。

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl disable ceph-osd@1
```

6. 停止 OSD。

```
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@0
[heat-admin@overcloud-cephstorage-0 ~]$ sudo systemctl stop ceph-osd@1
```

7. **在登录到 Controller 节点时，从 CRUSH map 中删除 OSD，以便它们不再接收数据。**

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd crush remove osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd crush remove osd.1
```

8. **移除 OSD 身份验证密钥。**

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
auth del osd.0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
auth del osd.1
```

9. **从集群中移除该 OSD。**

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd rm 0
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd rm 1
```

10. **从 CRUSH 映射中删除存储节点：**

```
[heat-admin@overcloud-controller-0 ~]$ sudo docker exec ceph-mon-<HOSTNAME> ceph
osd crush rm <NODE>
[heat-admin@overcloud-controller-0 ~]$ sudo ceph osd crush remove <NODE>
```

您可以通过搜索 CRUSH 树，确认 CRUSH map 中定义的 <NODE> 名称：

```
[heat-admin@overcloud-controller-0 ~]$ sudo podman exec ceph-mon-<HOSTNAME> ceph
osd crush tree | grep overcloud-osd-compute-3 -A 4
            "name": "overcloud-osd-compute-3",
            "type": "host",
            "type_id": 1,
            "items": []
        },
[heat-admin@overcloud-controller-0 ~]$
```

在 CRUSH 树中，确保 items 列表为空。如果列表不为空，请重新访问第 7 步。

11. 保留节点，并以 stack 用户身份返回到 undercloud。

```
[heat-admin@overcloud-controller-0 ~]$ exit
[stack@director ~]$
```

12. 禁用 Ceph Storage 节点，以便 director 不会重新置备它。

```
[stack@director ~]$ openstack baremetal node list
[stack@director ~]$ openstack baremetal node maintenance set UUID
```

13. 移除 Ceph Storage 节点需要使用本地模板文件更新 director 中的 overcloud 堆栈。首先识别 overcloud 堆栈的 UUID：

```
$ openstack stack list
```

14. 识别您要删除的 Ceph Storage 节点的 UUID：

```
$ openstack server list
```

15. 从堆栈中删除节点并相应地更新计划：

重要

如果您在创建 overcloud 时传递任何额外的环境文件，请使用 -e 选项再次传递它们，以避免对 overcloud 进行不必要的更改。有关更多信息，请参阅 Director 安装和使用指南中的修改 overcloud 环境。

```
$ openstack overcloud node delete /
--stack <stack-name> /
--templates /
-e <other-environment-files> /
<node_UUID>
```

16. 等待堆栈完成更新。使用 heat stack-list --show-nested 命令来监控堆栈更新。

17. 将新节点添加到 director 节点池中，并将它们部署为 Ceph Storage 节点。使用环境文件的 parameter_defaults 中的 CephStorageCount 参数，在本例中为 ~/templates/storage-

config.yaml 来定义 overcloud 中 Ceph Storage 节点的总数量。

```
parameter_defaults:
  ControllerCount: 3
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
```

注意

有关如何为每个角色定义节点数量的更多信息，请参阅 第 7.1 节 “为角色分配节点和类别”。

18. 更新环境文件后，重新部署 overcloud：

```
$ openstack overcloud deploy --templates -e <ENVIRONMENT_FILE>
```

director 置备新节点，并使用新节点的详细信息更新整个堆栈。

19. 以 heat-admin 用户身份登录 Controller 节点，并检查 Ceph Storage 节点的状态：

```
[heat-admin@overcloud-controller-0 ~]$ sudo ceph status
```

20. 确认 osdmap 部分中的值与您想要的集群中的节点数量匹配。您移除的 Ceph Storage 节点会替换为新节点。

11.3. 将 OSD 添加到 CEPH STORAGE 节点

此流程演示了如何将 OSD 添加到节点。有关 Ceph OSD 的更多信息，请参阅 Red Hat Ceph Storage Operations 指南中的 Ceph OSD。

流程

1. 请注意，以下 heat 模板将使用三个 OSD 设备部署 Ceph Storage：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

2. 要添加 OSD，请更新节点磁盘布局，如 第 5.3 节 “映射 Ceph Storage 节点磁盘布局” 所述。在本例中，将 /dev/sde 添加到模板中：

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
    osd_scenario: lvm
    osd_objectstore: bluestore
```

3. 运行 openstack overcloud deploy 以更新 overcloud。

注意

在本例中，所有具有 OSD 的主机都有一个新设备，称为 /dev/sde。如果您不希望所有节点具有新设备，请更新 heat 模板。有关如何使用不同 devices 列表定义主机的详情，请参考 第 5.5 节 “覆盖用于忽略 Ceph Storage 节点的参数” 和 第 5.5.1.2 节 “更改 Ceph Storage 节点中的磁盘布局”。

11.4. 从 CEPH STORAGE 节点移除 OSD

此流程演示了如何从节点中删除 OSD。它假定以下与环境有关：

- 服务器(ceph-storage0)有一个在 /dev/sde 上运行的 OSD (ceph-osd@4)。
- Ceph 监控服务(ceph-mon)在 controller0 上运行。

- 有足够的 OSD 来确保存储集群不达到其 near-full 比率。

有关 Ceph OSD 的更多信息，请参阅 Red Hat Ceph Storage Operations 指南中的 Ceph OSD。

流程

1. SSH 连接到 ceph-storage0，并以 root 身份登录。
2. 禁用和停止 OSD 服务：

```
[root@ceph-storage0 ~]# systemctl disable ceph-osd@4
[root@ceph-stoarge0 ~]# systemctl stop ceph-osd@4
```

3. 从 ceph-storage0 断开连接。
4. SSH 连接到 controller0，然后以 root 身份登录。
5. 识别 Ceph 监控容器的名称：

```
[root@controller0 ~]# podman ps | grep ceph-mon
ceph-mon-controller0
[root@controller0 ~]#
```

6. 启用 Ceph 监控容器，将不需要的 OSD 标记为 out ：

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd out 4
```

注意

此命令会导致 Ceph 重新平衡存储集群，并将数据复制到集群中的其他 OSD。在重新平衡完成前，集群会临时保留 active+clean 状态。

7. *运行以下命令并等待存储集群状态变为* ***active+clean*** *：*

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph -w
```

8. ***从 CRUSH map 中删除 OSD，使其不再接收数据：***

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd crush remove osd.4
```

9. ***删除 OSD 身份验证密钥：***

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph auth del osd.4
```

10. ***删除 OSD：***

```
[root@controller0 ~]# podman exec ceph-mon-controller0 ceph osd rm 4
```

11. ***从 controller0 断开连接。***

12. ***以 stack 用户身份通过 SSH 连接到 undercloud，再找到您定义 CephAnsibleDisksConfig 参数的 heat 环境文件。***

13. ***注意 heat 模板包含四个 OSD：***

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
      - /dev/sde
    osd_scenario: lvm
    osd_objectstore: bluestore
```

14. ***修改模板以删除 /dev/sde。***

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
```

```
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
  osd_scenario: lvm
  osd_objectstore: bluestore
```

15. *运行 **openstack overcloud deploy** 以更新 overcloud。*

注意

在本例中，您将从具有 OSD 的所有主机中删除 /dev/sde 设备。如果您没有从所有节点中删除同一设备，请更新 heat 模板。有关如何使用不同 devices 列表定义主机的详情，请参考 第 5.5 节 “覆盖用于忽略 Ceph Storage 节点的参数”。

第 12 章 替换失败的磁盘

如果 Ceph 集群中有一个磁盘失败，请完成以下步骤以替换它：

1. 确定设备名称是否有变化，请参阅 第 12.1 节 “确定是否存在设备名称更改”。
2. 确保 OSD 已关闭并销毁，请参阅 第 12.2 节 “确保 OSD 已关闭并销毁”。
3. 从系统中删除旧磁盘并安装替换磁盘，请参阅 第 12.3 节 “从系统中删除旧磁盘并安装替换磁盘”。
4. 验证磁盘替换是否成功，请参阅 第 12.4 节 “验证磁盘替换是否成功”。

12.1. 确定是否存在设备名称更改

在替换磁盘前，请确定替换 OSD 的替换磁盘是否在与您要替换的设备不同的操作系统中。如果替换磁盘具有不同的名称，您必须为 devices 列表更新 Ansible 参数，以便后续运行 ceph-ansible 时（包括 director 运行 ceph-ansible ）不会因为更改而失败。有关使用 director 时您必须更改的设备列表示例，请参阅 第 5.3 节 “映射 Ceph Storage 节点磁盘布局”。

警告

如果设备名称有变化，且您使用以下步骤在 ceph-ansible 或 director 之外更新您的系统，则配置管理工具与它们管理的系统不同步，直到您更新系统定义文件且配置在没有错误的情况下被重新分配。

存储设备的持久性命名

sd 驱动程序管理的存储设备可能在重启后可能始终具有相同的名称。例如，通常由 /dev/sdc 标识的磁盘可能命名为 /dev/sdb。即使您希望将一个磁盘作为 /dev/sdc 的替换磁盘，对于替换磁盘 /dev/sdc 在操作系统中也可以显示为 /dev/sdd。要解决这个问题，请使用持久的名称并匹配以下模式：/dev/disk/by-*。如需更多信息，请参阅 Red Hat Enterprise Linux (RHEL) 7 存储管理指南中的 持久性命名。

根据您用于部署 Ceph 的命名方法，您可能需要在替换 OSD 后更新 devices 列表。使用以下命名方法列表来确定是否必须更改设备列表：

主号码和次号范围方法

如果您使用 sd 并想继续使用它，请在安装新磁盘后检查名称是否已改变。如果名称没有改变，例如，如果与 /dev/sdd 正确显示相同的名称，则不需要在完成磁盘替换步骤后更改名称。

重要

不建议此命名方法，因为仍存在名称随时间不一致的风险。如需更多信息，请参阅 RHEL 7 存储管理指南 中的 持久性命名。

by-path 方法

如果您使用这个方法，且在同一插槽中添加替换磁盘，则路径一致，且不需要更改。

重要

虽然这种命名方法最好使用主号和次号范围方法，但要小心谨慎，以确保目标号不会改变。例如，如果主机适配器被移到不同的 PCI 插槽，请使用持久性绑定和更新名称。另外，如果以不同顺序载入驱动程序，或者系统上安装了新的 HBA，则 SCSI 主机号可能会改变，如果 HBA 无法探测。by-path 命名方法在 RHEL7 和 RHEL8 之间也有所不同。如需更多信息，请参阅：

- 文章 [在 RHEL8 和 RHEL7 中创建的"路径"链接之间有什么区别？] https://access.redhat.com/solutions/5171991
- RHEL 8 管理文件系统 指南中的 持久性命名属性概述。

by-uuid 方法

如果使用此方法，您可以使用 blkid 实用程序将新磁盘设置为与旧磁盘具有相同的 UUID。如需更多信息，请参阅 RHEL 7 存储管理指南 中的 持久性命名。

by-id 方法

如果使用这个方法，您必须更改 devices 列表，因为此标识符是设备的属性，且设备已被替换。

当您向系统添加新磁盘时，如果可以根据 RHEL7 Storage Administrator 指南 修改持久性命名属性，请参阅 Persistent Naming，以便设备名称保持不变，则不需要更新 devices 列表并重新运行 ceph-ansible，或者触发 director 重新运行 ceph-ansible，以重新运行 ceph-ansible，您可以继续执行磁盘替换过程。但是，您可以重新运行 ceph-ansible 以确保更改不会导致任何不一致。

12.2. 确保 OSD 已关闭并销毁

在托管 Ceph Monitor 的服务器上，使用正在运行的 monitor 容器中的 ceph 命令，以确保您要替换的 OSD 为 down，然后销毁它。

流程

1. 识别正在运行的 Ceph 监控容器的名称，并将其存储在名为 MON 的环境变量中：

```
MON=$(podman ps | grep ceph-mon | awk {'print $1'})
```

2. 对 ceph 命令的别名，使其在运行的 Ceph 监控容器内执行：

```
alias ceph="podman exec $MON ceph"
```

3. 使用新别名验证您要替换的 OSD 的状态是否为 down ：

```
[root@overcloud-controller-0 ~]# ceph osd tree | grep 27
27   hdd 0.04790         osd.27                 down  1.00000 1.00000
```

4. 销毁 OSD。以下示例命令销毁 OSD 27 ：

```
[root@overcloud-controller-0 ~]# ceph osd destroy 27 --yes-i-really-mean-it
destroyed osd.27
```

12.3. 从系统中删除旧磁盘并安装替换磁盘

在容器主机上，使用您要替换的 OSD，从系统中删除旧磁盘并安装替换磁盘。

先决条件：

- 验证设备 ID 是否已更改：更多信息请参阅 第 12.1 节 “确定是否存在设备名称更改”。

ceph-volume 命令存在于 Ceph 容器中，但没有安装到 overcloud 节点上。创建一个别名，使 ceph-volume 命令能够在 Ceph 容器内运行 ceph-volume 二进制文件。然后，使用 ceph-volume 命令清理新磁盘，并将它添加为 OSD。

流程

1. 确保失败的 OSD 没有运行：

```
systemctl stop ceph-osd@27
```

2. 识别 ceph 容器镜像的镜像 ID，并将其存储在名为 IMG 的环境变量中：

```
IMG=$(podman images | grep ceph | awk {'print $3'})
```

3. 为 ceph-volume 命令别名，使其在 $IMG Ceph 容器内运行，使用 ceph-volume 入口点和相关目录：

```
alias ceph-volume="podman run --rm --privileged --net=host --ipc=host -v
/run/lock/lvm:/run/lock/lvm:z -v /var/run/udev/:/var/run/udev/:z -v /dev:/dev -v
/etc/ceph:/etc/ceph:z -v /var/lib/ceph/:/var/lib/ceph/:z -v /var/log/ceph/:/var/log/ceph/:z --
entrypoint=ceph-volume $IMG --cluster ceph"
```

4. 验证 aliased 命令是否已成功运行：

```
ceph-volume lvm list
```

5. 检查您的新 OSD 设备是否还没有作为 LVM 的一部分。使用 pvdisplay 命令检查设备，并确保 VG Name 字段为空。将 <NEW_DEVICE > 替换为新 OSD 设备的 /dev sections 路径：

```
[root@overcloud-computehci-2 ~]# pvdisplay <NEW_DEVICE>
  --- Physical volume ---
  PV Name               /dev/sdj
  VG Name               ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
  PV Size               50.00 GiB / not usable 1.00 GiB
  Allocatable           yes (but full)
  PE Size               1.00 GiB
  Total PE              49
  Free PE               0
```

```
  Allocated PE          49
  PV UUID               kOO0If-ge2F-UH44-6S1z-9tAv-7ypT-7by4cp
[root@overcloud-computehci-2 ~]#
```

如果 VG Name 字段不为空，则设备将属于您必须删除的卷组。

6. 如果设备属于卷组，请使用 lvdisplay 命令检查卷组中是否存在逻辑卷。将 <VOLUME_GROUP > 替换为您从 pvdisplay 命令检索到的 VG Name 字段的值：

```
[root@overcloud-computehci-2 ~]# lvdisplay | grep <VOLUME_GROUP>
  LV Path                /dev/ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2/osd-data-a0810722-
7673-43c7-8511-2fd9db1dbbc6
  VG Name                ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2
[root@overcloud-computehci-2 ~]#
```

如果 LV Path 字段不为空，则该设备会包含您必须删除的逻辑卷。

7. 如果新设备是逻辑卷或卷组的一部分，请删除逻辑卷、卷组和逻辑卷以及设备关联作为 LVM 系统中的物理卷。

- 将 <LV_PATH > 替换为 LV Path 字段的值。
- 将 <VOLUME_GROUP > 替换为 VG Name 字段的值。
- 将 <NEW_DEVICE > 替换为新 OSD 设备的 /dev sections 路径。

```
[root@overcloud-computehci-2 ~]# lvremove --force <LV_PATH>
  Logical volume "osd-data-a0810722-7673-43c7-8511-2fd9db1dbbc6" successfully
removed
```

```
[root@overcloud-computehci-2 ~]# vgremove --force <VOLUME_GROUP>
  Volume group "ceph-0fb0de13-fc8e-44c8-99ea-911e343191d2" successfully removed
```

```
[root@overcloud-computehci-2 ~]# pvremove <NEW_DEVICE>
  Labels on physical volume "/dev/sdj" successfully wiped.
```

8. 确保新 OSD 设备清理干净。在以下示例中，该设备为 /dev/sdj ：

```
[root@overcloud-computehci-2 ~]# ceph-volume lvm zap /dev/sdj
```

```
--> Zapping: /dev/sdj
--> --destroy was not specified, but zapping a whole device will remove the partition table
Running command: /usr/sbin/wipefs --all /dev/sdj
Running command: /bin/dd if=/dev/zero of=/dev/sdj bs=1M count=10
 stderr: 10+0 records in
10+0 records out
10485760 bytes (10 MB, 10 MiB) copied, 0.010618 s, 988 MB/s
--> Zapping successful for: <Raw Device: /dev/sdj>
[root@overcloud-computehci-2 ~]#
```

9. 使用新设备创建具有现有 OSD ID 的新 OSD，但传递 **--no-systemd**，以便 ceph-volume 不会尝试启动 OSD。这无法从容器中实现：

```
ceph-volume lvm create --osd-id 27 --data /dev/sdj --no-systemd
```

10. 启动容器外的 OSD：

```
systemctl start ceph-osd@27
```

12.4. 验证磁盘替换是否成功

要检查您的磁盘替换是否成功，在 undercloud 中完成以下步骤。

流程

1. 检查设备名称是否已改变，根据您用于部署 Ceph 的命名方法更新 devices 列表。更多信息请参阅 第 12.1 节 “确定是否存在设备名称更改”。

2. 为确保更改没有引入任何不一致的情况，请重新运行 **overcloud deploy** 命令来执行堆栈更新。

3. 如果您有具有不同设备列表的主机，您可能需要定义例外。例如，您可以使用以下示例 **heat** 环境文件来部署具有三个 **OSD** 设备的节点。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
    osd_objectstore: bluestore
```

CephAnsibleDisksConfig 参数应用到托管 OSD 的所有节点，因此您无法使用新设备列表更新 devices 参数。反之，您必须为具有不同设备列表的新主机定义例外。有关定义例外的更多信息，请参阅 第 5.5 节 “覆盖用于忽略 Ceph Storage 节点的参数” 和 第 5.5.1.2 节 “更改 Ceph Storage 节点中的磁盘布局”。

附录 A. 示例环境文件：创建 CEPH STORAGE 集群

以下自定义环境文件使用整个 第 2 章 为 overcloud 部署准备 Ceph Storage 节点 描述的许多选项。此示例不包括任何注释选项。有关环境文件的概述，请参阅环境文件 (包括在高级 Overcloud 自定义指南中）。

/home/stack/templates/storage-config.yaml

```
parameter_defaults: (1)
  CinderBackupBackend: ceph (2)
  CephAnsibleDisksConfig: (3)
    osd_scenario: lvm
    osd_objectstore: bluestore
    dmcrypt: true
    devices:
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:10:0
      - /dev/disk/by-path/pci-0000:03:00.0-scsi-0:0:11:0
      - /dev/nvme0n1
  ControllerCount: 3 (4)
  OvercloudControlFlavor: control
  ComputeCount: 3
  OvercloudComputeFlavor: compute
  CephStorageCount: 3
  OvercloudCephStorageFlavor: ceph-storage
  CephMonCount: 3
  OvercloudCephMonFlavor: ceph-mon
  CephMdsCount: 3
  OvercloudCephMdsFlavor: ceph-mds
  NeutronNetworkType: vxlan (5)
```

(1)

parameter_defaults 部分修改所有模板中的参数的默认值。此处列出的大多数条目在 第 4 章 自定义存储服务 中进行了描述。

(2)

如果要部署 Ceph 对象网关，您可以使用 Ceph Object Storage (ceph-rgw)作为备份目标。要配置此功能，请将 CinderBackupBackend 设置为 swift。详情请查看 第 4.2 节 “启用 Ceph 对象网关”。

(3)

4

对于每个角色，*Count 参数分配多个节点，而 Overcloud*Flavor 参数分配类别。例如，Controller Count: 3 将 3 个节点分配给 Controller 角色，而 OvercloudControlFlavor: 控制将每个角色设置为使用 control 类型。详情请查看 第 7.1 节 “为角色分配节点和类别”。

注意

CephMonCount、CephMdsCount、OvercloudCephMonFlavor 和 OvercloudCephMdsFlavor 参数（以及 ceph-mon 和 ceph-mds 类别）只有在您创建了自定义 CephMON 和 CephMds 角色时，才有效，如 第 3 章 在专用节点上部署 Ceph 服务 所述。

5

NeutronNetworkType：设置 neutron 服务应使用的网络类型（本例中为 vxlan）。

附录 B. 自定义接口模板示例：多个绑定接口

以下模板是 /usr/share/openstack-tripleo-heat-templates/network/config/bond-with-vlans/ceph-storage.yaml 的自定义版本。它具有多个绑定接口来隔离后端和前端存储网络流量，以及两个连接的冗余，如 第 4.5 节 “为 Ceph 节点配置多个绑定接口” 所述。

它还使用自定义绑定选项 'mode=4 lacp_rate=1'，如 第 4.5.1 节 “配置绑定模块指令” 所述。

/usr/share/openstack-tripleo-heat-templates/network/bond-with-vlans/ceph-storage.yaml (custom)

```
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with VLANs attached for the ceph storage role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
    type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
    type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
    type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
    type: string
  ManagementIpSubnet: # Only populated when including environments/network-
management.yaml
    default: ''
    description: IP address/subnet on the management network
    type: string
  BondInterfaceOvsOptions:
    default: 'mode=4 lacp_rate=1'
```

```
    description: The bonding_options string for the bond interface. Set
                 things like lacp=active and/or bond_mode=balance-slb
                 using this option.
    type: string
    constraints:
      - allowed_pattern: "^((?!balance.tcp).)*$"
        description: |
          The balance-tcp bond mode is known to cause packet loss and
          should not be used in BondInterfaceOvsOptions.
  ExternalNetworkVlanID:
    default: 10
    description: Vlan ID for the external network traffic.
    type: number
  InternalApiNetworkVlanID:
    default: 20
    description: Vlan ID for the internal_api network traffic.
    type: number
  StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
    type: number
  StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
    type: number
  TenantNetworkVlanID:
    default: 50
    description: Vlan ID for the tenant network traffic.
    type: number
  ManagementNetworkVlanID:
    default: 60
    description: Vlan ID for the management network traffic.
    type: number
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
  ExternalInterfaceDefaultRoute: # Not used by default in this template
    default: '10.0.0.1'
    description: The default route of the external network.
    type: string
  ManagementInterfaceDefaultRoute: # Commented out by default in this template
    default: unset
    description: The default route of the management network.
    type: string
  DnsServers: # Override this via parameter_defaults
    default: []
    description: A list of DNS servers (2 max for some implementations) that will be added to
resolv.conf.
    type: comma_delimited_list
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - - {get_param: ControlPlaneIp}
                        - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
            -
              type: ovs_bridge
              name: br-bond
              members:
                -
                  type: linux_bond
                  name: bond1
                  bonding_options: {get_param: BondInterfaceOvsOptions}
                  members:
                    -
                      type: interface
                      name: nic2
                      primary: true
                    -
                      type: interface
                      name: nic3
                -
                  type: vlan
                  device: bond1
                  vlan_id: {get_param: StorageNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: StorageIpSubnet}
            -
              type: ovs_bridge
              name: br-bond2
              members:
                -
```

```
            type: linux_bond
            name: bond2
            bonding_options: {get_param: BondInterfaceOvsOptions}
            members:
              -
                type: interface
                name: nic4
                primary: true
              -
                type: interface
                name: nic5
          -
            type: vlan
            device: bond1
            vlan_id: {get_param: StorageMgmtNetworkVlanID}
            addresses:
              -
                ip_netmask: {get_param: StorageMgmtIpSubnet}
outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value: {get_resource: OsNetConfigImpl}
```