Red Hat OpenStack Platform 16.2

分布式计算节点和存储部署

部署 Red Hat OpenStack Platform 分布式计算节点技术

Last Updated: 2024-05-21

Red Hat OpenStack Platform 16.2 分布式计算节点和存储部署

部署 Red Hat OpenStack Platform 分布式计算节点技术

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

您可以使用分布式计算节点(DCN)架构部署 Red Hat OpenStack Platform (RHOSP)架构，以便使用 heat 堆栈分离的边缘站点操作性。每个站点都可以具有自己的用于镜像服务(glance)多存储的 Ceph 存储后端。

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。

在 JIRA 中提供文档反馈

使用 Create Issue 表单对文档提供反馈。JIRA 问题将在 Red Hat OpenStack Platform Jira 项目中创建，您可以在其中跟踪您的反馈进度。

1. 确保您已登录到 JIRA。如果您没有 JIRA 帐户，请创建一个帐户来提交反馈。
2. 点击以下链接打开 **Create Issue** 页面： Create Issue
3. 完成 **Summary** 和 **Description** 字段。在 **Description** 字段中，包含文档 URL、章节或章节号以及问题的详细描述。不要修改表单中的任何其他字段。
4. 点 **Create**。

第 1 章 了解 DCN

注意

分布式 Compute 节点(DCN)部署不支持从 Red Hat OpenStack Platform (RHOSP) 16.2 升级到 RHOSP 17.1。

分布式计算节点(DCN)架构适用于边缘用例，允许在共享常见的中央化 control plane 时远程部署远程计算节点。DCN 架构允许您定位更接近操作需求的工作负载，以提高性能。

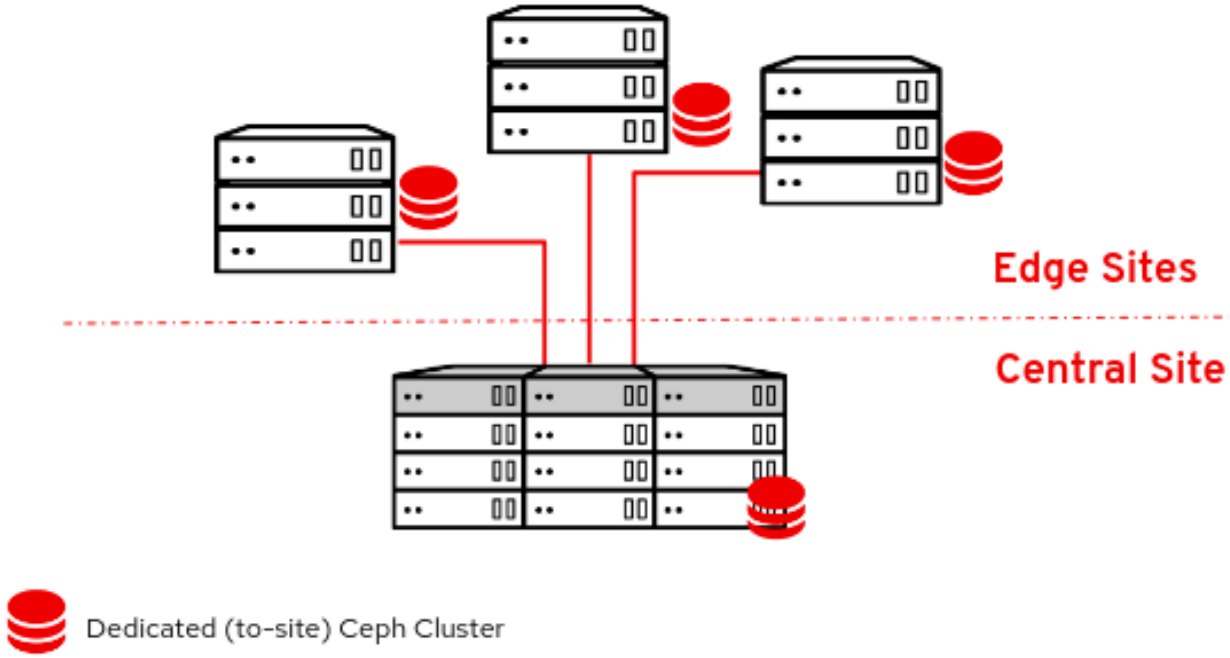
中央位置可由任何角色组成，但至少需要三个控制器。Compute 节点可以存在于边缘，也可存在于中央位置。

DCN 架构是一个 hub 和 spoke 路由网络部署。DCN 与使用 Red Hat OpenStack Platform director 路由置备和 control plane 网络相关的 spine 和 leaf 部署类似。

- hub 是包含核心路由器和数据中心网关(DC-GW)的中央网站。
- spoke 是远程边缘或页（leaf）。

边缘位置没有控制器，使其架构与 Red Hat OpenStack Platform 的传统部署不同：

- control plane 服务在中央位置远程运行。
- 没有安装 Pacemaker。
- 块存储服务(cinder)以主动/主动模式运行。
- etcd 部署为分布式锁定管理器(DLM)。



1.1. 分布式计算节点架构所需的软件

下表显示了在分布式计算节点(DCN)架构中部署 Red Hat OpenStack Platform 所需的软件和最小版本：

平台	版本	选填
Red Hat Enterprise Linux	8	否
Red Hat OpenStack Platform	16.1	否
Red Hat Ceph Storage	4	是

1.2. 多堆栈设计

当您使用 DCN 设计部署 Red Hat OpenStack Platform (RHOSP)时，您可以使用 Red Hat director 的功能进行多个堆栈部署和管理，将每个站点部署为不同的堆栈。

不支持将 DCN 架构作为单一堆栈进行管理，除非部署是从 Red Hat OpenStack Platform 13 的升级。不支持分割现有堆栈的方法，但您可以将堆栈添加到预先存在的部署中。更多信息请参阅 第 A.3 节 “迁移到多堆栈部署”。

中央位置是 RHOSP 的传统堆栈部署，但您不需要使用中央堆栈部署 Compute 节点或 Red Hat Ceph 存储。

使用 DCN 时，您可以将每个位置部署为不同的可用区(AZ)。

1.3. DCN 存储

您可以在超融合节点上部署每个边缘站点，可以是没有存储，也可以使用 Ceph 部署。您部署的存储专用于您部署到的站点。

DCN 架构使用 Glance 多存储。对于在没有存储的情况下部署的边缘站点，提供了额外的工具，以便您可以在计算服务(nova)缓存中缓存和存储镜像。nova 中的缓存 glance 镜像通过避免在 WAN 链接中下载镜像的过程，为实例提供更快的引导时间。更多信息请参阅 第 10 章 *将 glance 镜像预缓存到 nova*。

1.4. DCN EDGE

使用分布式 Compute 节点架构时，中央位置使用管理边缘位置的控制节点进行部署。当您部署边缘位置时，您将仅部署计算节点，使边缘站点架构与 Red Hat OpenStack Platform 的传统部署不同。在边缘位置：

- control plane 服务在中央位置远程运行。
- Pacemaker 不在 DCN 站点中运行。
- 块存储服务(cinder)以主动/主动模式运行。
- etcd 部署为分布式锁定管理器(DLM)。

第 2 章 规划分布式 COMPUTE 节点(DCN)部署

在规划 DCN 架构时，请检查您需要的技术是否可用并支持。

2.1. DCN 架构存储注意事项

DCN 架构目前不支持以下功能：

- 从 Red Hat OpenStack Platform 13 都 16，在分布式计算节点架构中快进更新 (FFU)。
- 在边缘站点间复制卷快照。您可以通过从卷创建镜像并使用 glance 复制镜像来解决这个问题。复制镜像后，您可以从其中创建卷。
- 边缘的 Ceph Rados 网关(RGW)
- CephFS 位于边缘。
- 边缘站点上的实例高可用性(HA)。
- 站点之间的 RBD 镜像。
- 实例迁移、实时或冷站点，也可从中央位置到边缘站点。您仍然可以在站点边界内迁移实例。要在站点之间移动镜像，您必须对镜像进行快照，并使用 **glance image-import**。如需更多信息，请参阅 确认可以在站点之间创建和复制镜像快照。

另外，您必须考虑以下几点：

- 您必须将镜像上传到中央位置，然后才能将镜像复制到边缘站点；每个镜像的副本必须存在于中央位置的镜像服务(glance)中。
- 在边缘站点创建实例前，您必须在该边缘站点具有镜像的本地副本。
- 您必须将 RBD 存储驱动程序用于 Image、Compute 和 Block Storage 服务。
- 对于每个站点，分配一个唯一可用区，并将相同的值用于 NovaComputeAvailabilityZone 和 CinderStorageAvailabilityZone 参数。
- 您可以将离线卷从边缘站点迁移到中央位置，反之亦然。您无法直接在边缘站点之间迁移卷。

2.2. DCN 架构上的网络注意事项

DCN 架构目前不支持以下功能：

- Octavia
- DPDK 节点上的 DHCP
- conntrack 用于 TC Flower Hardware Offload

TC Flower Hardware Offload 的 conntrack 作为技术预览在 DCN 上提供，因此红帽不会完全支持使用这些解决方案。此功能应该只用于 DCN 进行测试，不应在生产环境中部署。如需有关技术预览功能的更多信息，请参阅覆盖范围详情。

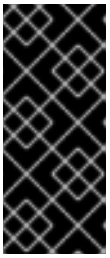
完全支持以下 ML2/OVS 技术：

- DPDK 节点上没有 DHCP 的 OVS-DPDK

- SR-IOV
- TC 流硬件卸载，没有 conntrack
- 边缘使用网络器节点的 Neutron 可用区(AZ)，每个站点一个 AZ
- 路由供应商网络

完全支持以下 ML2/OVN 网络技术：

- DPDK 节点上没有 DHCP 的 OVS-DPDK
- SR-IOV （没有 DHCP）
- TC 流硬件卸载，没有 conntrack
- 路由供应商网络
- 支持 Neutron AZ 的 OVN GW （网络节点）



重要

通过设置 **OVNCMSOptions: 'enable-chassis-as-gw'**，并通过为 **OVNAvailabilityZone** 参数提供一个或多个 AZ 值来确保所有路由器网关端口驻留在 OpenStack Controller 节点上。执行此操作可防止路由器将所有机箱调度到路由器网关端口的潜在主机。如需更多信息，请参阅网络指南中的 使用 ML2/OVN 配置网络服务可用域。

另外，您必须考虑以下几点：

- *网络延迟：根据往返时间(RTT)来测量延迟，以及预期的并发 API 操作数来保持可接受的性能。最大 TCP/IP 吞吐量与 RTT 相反。您可以通过调整内核 TCP 参数来降低带有高带宽的高延迟连接的一些问题。如果跨站点通信超过 100 ms，则红帽支持红帽支持。*
- *Network drop outs：如果边缘站点暂时丢失与中央站点的连接，则在停机期间无法在受影响的边缘站点内执行 OpenStack control plane API 或 CLI 操作。例如，边缘站点上的 Compute 节点无法创建实例的快照、发布身份验证令牌或删除镜像。常规 OpenStack control plane API 和 CLI 操作在此中断过程中仍然可以正常工作，并可继续为具有工作连接的任何其他边缘站点提供服务。*
  镜像类型：在使用 Ceph 存储部署 DCN 架构时，您必须使用 raw 镜像。
- *镜像大小：*
  - *Overcloud 节点镜像 - 从中央 undercloud 节点下载 overcloud 节点镜像。这些镜像可能是大型文件，在调配期间，所有必要网络将从中央站点传输到边缘站点。*
  - *实例镜像：如果没有边缘的块存储，则镜像服务镜像会在第一次使用过程中遍历 WAN。镜像会在本地复制到目标边缘节点，以便随后使用。glance 镜像没有大小限制。传输时间因可用带宽和网络延迟而异。*
    如果边缘存在块存储，则镜像将通过 WAN 异步复制，以便更快地在边缘引导时复制。
- *提供商网络：这是 DCN 部署的建议网络方法。如果您在远程站点中使用提供商网络，那么您必须考虑网络服务(neutron)不会放置任何限制或检查您可以附加可用网络的位置。例如，如果您只在边缘站点 A 中使用提供商网络，您必须确保不要尝试附加到边缘站点 B 中的提供商网络。这是因为，将其绑定到 Compute 节点时，供应商网络中没有验证检查。*
- *特定于站点的网络：如果您使用特定于特定站点的网络，则 DCN 网络中会出现一个限制：当您使用 Compute 节点部署集中式 neutron 控制器时，neutron 中没有触发器将特定的 Compute 节点*

识别为远程节点。因此，计算节点接收其他 Compute 节点列表，并在彼此之间自动形成隧道；隧道从边缘到边缘，通过中央站点从边缘到边缘。如果您使用 VXLAN 或 Geneve，每个站点中的每个 Compute 节点都会与所有其他 Compute 节点和 Controller 节点组成一个隧道，无论它们是本地还是远程节点。如果您在任何位置使用相同的 neutron 网络，则这不是问题。使用 VLAN 时，neutron 期望所有 Compute 节点具有相同的网桥映射，并且所有 VLAN 都位于每个站点。

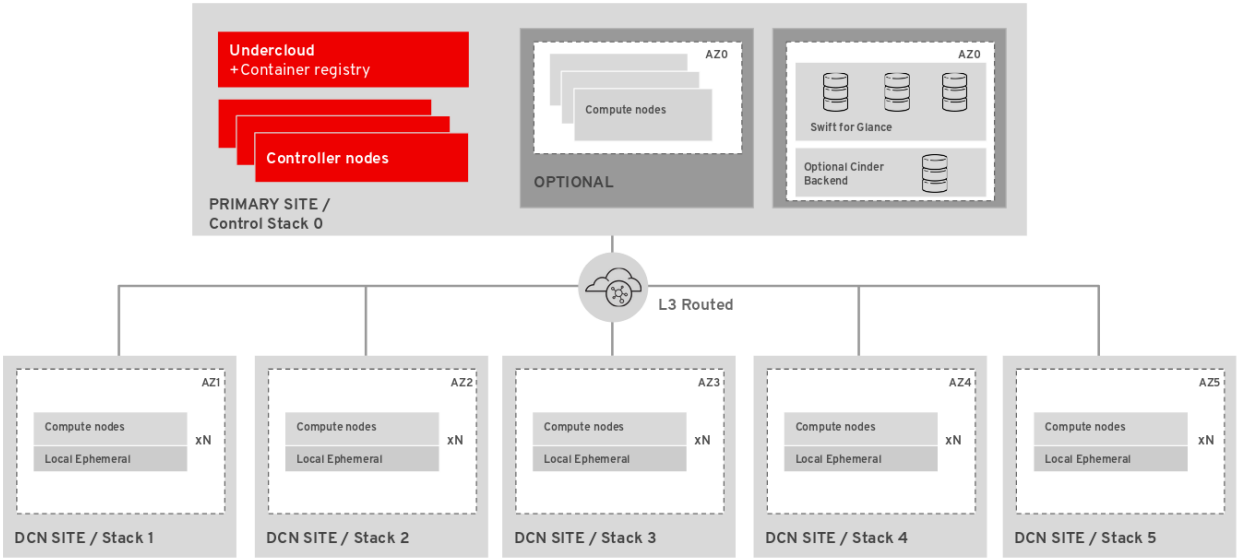
- 其他站点：如果您需要从中央站点扩展到额外的远程站点，您可以使用 Red Hat OpenStack Platform director 上的 openstack CLI 来添加新的网络段和子网。
- 如果边缘服务器没有被预置备，您必须配置 DHCP 转发，以便在路由片段上内省和置备。
- 路由必须在云或将每个边缘站点连接到 hub 的网络基础架构中配置。您应该实施网络设计，为每个 Red Hat OpenStack Platform 集群网络（外部、内部 API 等）分配 L3 子网，每个站点都是唯一的。

2.3. 边缘的存储拓扑和角色

当您使用分布式计算节点架构部署 Red Hat OpenStack 平台时，您必须决定是否需要边缘存储。根据存储和性能需求，您可以使用三种配置之一部署每个站点。并非所有边缘站点都必须具有相同的配置。

没有存储的 DCN

要部署此架构，请使用 **Compute** 角色。



边缘没有块存储：

- control plane 上的 Object Storage (swift)服务用作镜像(glance)后端服务。
- 多后端服务服务不可用。
  - 在 Nova 中，镜像在边缘站点本地进行缓存。如需更多信息，请参阅 第 10 章 将 glance 镜像预缓存到 nova。
- 实例存储在本地 Compute 节点上。
- 边缘站点不提供 Block Storage (cinder)等卷服务。

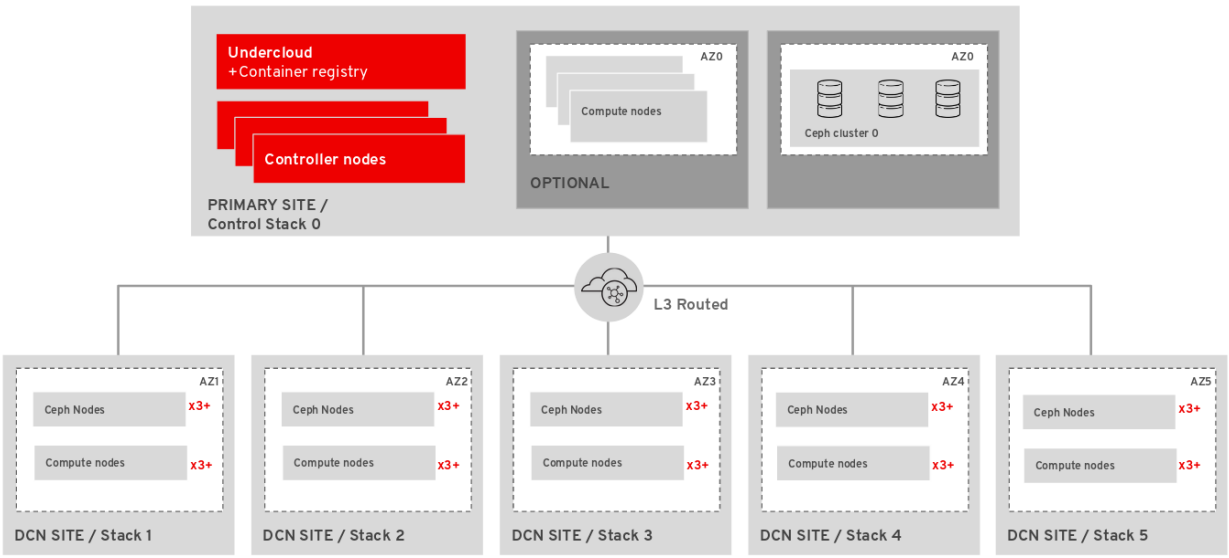
重要

如果没有使用 Red Hat Ceph Storage 部署中央位置，则不能在以后部署带有存储的边缘站点。

有关在边缘部署没有块存储的详情，请参考 第 6.1 节 "在没有存储的情况下部署边缘节点"。

带有存储的 DCN

*要使用存储部署 DCN，还必须在中央位置部署 Red Hat Ceph Storage。您需要使用 **dcn-storage.yaml** 和 **ceph-ansible.yaml** 环境文件。对于包含非超融合 Red Hat Ceph Storage 节点的边缘站点，请使用 **DistributedCompute**、distributed**ComputeScaleOut**、CephAll、**CephAll** 和 **CephStorage** 角色。*

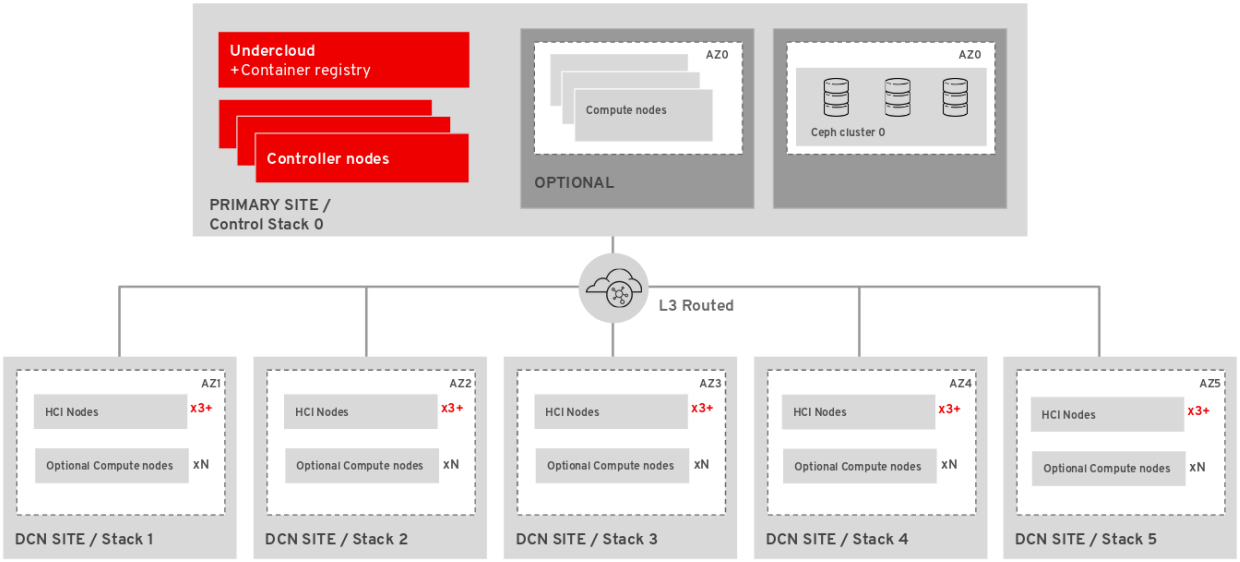


在边缘使用块存储：

- *Red Hat Ceph 块设备(RBD)用作镜像(glance)服务后端。*
- *多后端服务服务(glance)可用，以便在中央和 DCN 站点之间复制镜像。*
- *Block Storage (cinder)服务在所有站点上都可用，可使用 Red Hat Ceph Block Devices (RBD) 驱动程序访问。*
- *Block Storage (cinder)服务在 Compute 节点上运行，Red Hat Ceph Storage 在专用存储节点上运行。*
- *Nova 临时存储由 Ceph (RBD)支持。*
  更多信息请参阅 第 5.2 节 "使用存储部署中央站点"。

带有超融合存储的 DCN

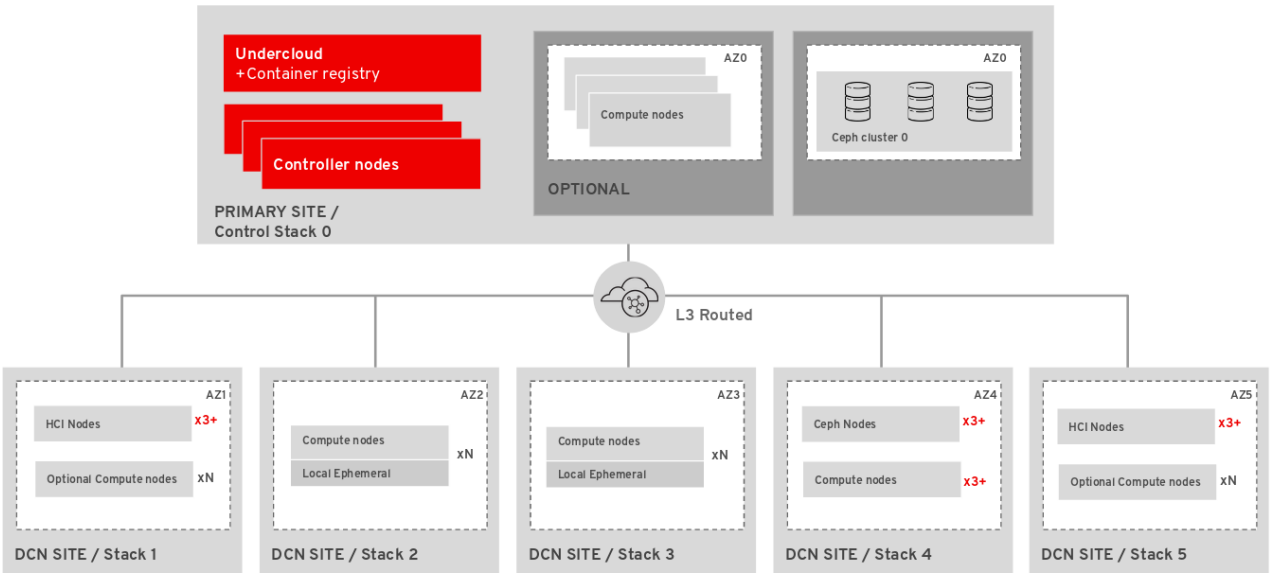
*若要部署此配置，还必须在中央位置部署 Red Hat Ceph Storage。您需要配置 **dcn-storage.yaml** 和 **ceph-ansible.yaml** 环境文件。使用 **DistributedComputeHCI**，以及 **DistributedComputeHCIScaleOut** 角色。您还可以使用 **DistributedComputeScaleOut** 角色来添加不参与提供 Red Hat Ceph Storage 服务的 Compute 节点。*



使用边缘的超融合存储：

- *Red Hat Ceph 块设备(RBD)用作镜像(glance)服务后端。*
- *多后端服务服务(glance)可用，以便在中央和 DCN 站点之间复制镜像。*
- *Block Storage (cinder)服务在所有站点上都可用，可使用 Red Hat Ceph Block Devices (RBD) 驱动程序访问。*
- *Block Storage 服务和 Red Hat Ceph Storage 在 Compute 节点上运行。*
  更多信息请参阅 第 7.1 节 "使用存储部署边缘站点"。

当您在分布式计算架构中部署 Red Hat OpenStack Platform 时，您可以选择部署多个存储拓扑，在每个站点都有唯一的配置。您必须使用 Red Hat Ceph Storage 部署中央位置，以使用存储部署任何边缘站点。



2.3.1. *边缘部署的角色*

以下角色可用于边缘部署。根据您的所选配置，为您的环境选择适当的角色。

Compute

***Compute** 角色用于没有存储的边缘部署。*

DistributedCompute

***DistributedCompute** 角色用于没有超融合节点的存储部署。**DistributedCompute** 角色包含 **GlanceApiEdge** 服务，它确保镜像服务在本地边缘站点而不是位于中央 hub 位置。您可以使用 **DistributedCompute** 角色部署最多三个节点。对于任何其他节点，请使用 **DistributedComputeScaleOut** 角色。*

DistributedComputeScaleOut

***DistributedComputeScaleOut** 角色包含 **HAproxyEdge** 服务，它允许在 DistributedComputeScaleOut 角色中创建的实例将对镜像服务的请求代理到边缘站点提供该服务的节点。使用 **DistributedCompute** 角色部署三个节点后，您可以使用 DistributedComputeScaleOut 角色来扩展计算资源。使用 **DistrubutedComputeScaleOut** 角色部署没有最少的主机数量。此角色用于没有超融合节点的存储部署。*

DistributedComputeHCI

***DistributedComputeHCI** 角色通过包括 Ceph 管理和 OSD 服务，在边缘启用超融合部署。使用 DistributedComputeHCI 角色时，您必须使用三个节点。此角色用于具有完全聚合节点的存储部署。*

DistributedComputeHCIScaleOut

***DistributedComputeHCIScaleOut** 角色包含 **Ceph OSD** 服务，允许在将更多节点添加到边缘时通过计算扩展存储容量。此角色还包括 **HAproxyEdge** 服务，用于将镜像下载请求重定向到边缘站点的 **GlanceAPIEdge** 节点。此角色在边缘启用超线程部署。使用 **DistributedComputeHCI** 角色时，您必须使用三个节点。此角色用于带有超融合节点的存储部署。*

CephAll

***CephAll** 角色包括 Ceph OSD、Ceph mon 和 Ceph Mgr 服务。此角色用于没有超融合节点的存储部署。您可以使用 CephAll 角色部署最多三个节点。对于任何其他存储容量，请使用 CephStorage 角色。*

CephStorage

***CephStorage** 角色包含 Ceph OSD 服务。此角色用于没有超融合节点的存储部署。如果三个 CephAll 节点没有足够的存储容量，请根据需要添加任意数量的 CephStorage 节点。*

第 3 章 在 UNDERCLOUD 中配置路由 SPINE-LEAF

本节论述了如何配置 undercloud 以容纳与可组合网络相关的路由热插拔的用例。

3.1. 配置 SPINE LEAF PROVISIONING 网络

要为 spine leaf 基础架构配置 provisioning 网络，请编辑 **undercloud.conf** 文件并设置以下流程中包含的相关参数。

流程

1. 以 **stack** 用户的身份登录 undercloud。
2. 如果您还没有 **undercloud.conf** 文件，请复制示例模板文件：

```
[stack@director ~]$ cp /usr/share/python-tripleoclient/undercloud.conf.sample
~/undercloud.conf
```

3. 编辑 **undercloud.conf** 文件。
4. 在 **[DEFAULT]** 部分中设置以下值：
   a. 将 **local_ip** 设置为 **leaf0** 上的 undercloud IP：

```
local_ip = 192.168.10.1/24
```

   b. 将 **undercloud_public_host** 设置为 undercloud 的面向外部的 IP 地址：

```
undercloud_public_host = 10.1.1.1
```

   c. 将 **undercloud_admin_host** 设置为 undercloud 的管理 IP 地址。这个 IP 地址通常位于 leaf0 上：

```
undercloud_admin_host = 192.168.10.2
```

   d. 将 **local_interface** 设置为本地网络的桥接接口：

```
local_interface = eth1
```

   e. 将 **enable_routed_networks** 设置为 **true** ：

```
enable_routed_networks = true
```

   f. 使用 **subnets** 参数定义子网列表。在路由 spine 和 leaf 中为每个 L2 片段定义一个子网：

```
subnets = leaf0,leaf1,leaf2
```

   g. 使用 **local_subnet** 参数指定与 undercloud 本地的物理 L2 段关联的子网：

```
local_subnet = leaf0
```

   h. 设置 **undercloud_nameservers** 的值。

```
undercloud_nameservers = 10.11.5.19,10.11.5.20
```

提示

您可以通过查看 /etc/resolv.conf 来查找用于 undercloud 名称服务器的 DNS 服务器的当前 IP 地址。

5. *为您在 **subnets** 参数中定义的每个子网创建一个新部分：*

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False
```

6. *保存 **undercloud.conf** 文件。*

7. *运行 undercloud 安装命令：*

```
[stack@director ~]$ openstack undercloud install
```

此配置在 provisioning 网络或 control plane 上创建三个子网。overcloud 使用每个网络来置备每个对应叶中的系统。

为确保正确将 DHCP 请求转发到 undercloud，您可能需要配置 DHCP 转发。

3.2. 配置 DHCP 转发

您可以在连接到要转发请求的远程网络段的交换机、路由器或服务器上运行 DHCP 转发服务。

注意

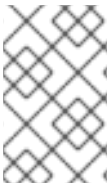
不要在 undercloud 上运行 DHCP 转发服务。

undercloud 在 provisioning 网络中使用两个 DHCP 服务器：

- 内省 DHCP 服务器。
- 置备 DHCP 服务器。

您必须将 DHCP 转发配置为将 DHCP 请求转发到 undercloud 上的两个 DHCP 服务器。

您可以将 UDP 广播与支持它的设备一起使用，将 DHCP 请求转发到 undercloud 置备网络的 L2 网络段。或者，您可以使用 UDP 单播，将 DHCP 请求转发到特定的 IP 地址。



注意

在特定设备类型中配置 DHCP 转发超出了本文档的范围。作为参考，本文档提供了使用 ISC DHCP 软件中的实现的 DHCP 转发配置示例。如需更多信息，请参阅手册页 dhcrelay (8)。

重要

某些中继需要 DHCP 选项 79，特别是提供 DHCPv6 地址的转发，以及不通过原始 MAC 地址的中继。如需更多信息，请参阅 RFC6939。

广播 DHCP 转发

此方法使用 UDP 广播流量将 DHCP 请求转发到 DHCP 服务器或服务器所在的 L2 网络段。网络段上的所有设备都会接收广播流量。使用 UDP 广播时，undercloud 上的两个 DHCP 服务器都会接收中继的 DHCP 请求。根据实现，您可以通过指定接口或 IP 网络地址来配置它：

Interface

指定连接到 DHCP 请求中继的 L2 网络段的接口。

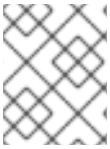
IP 网络地址

指定 DHCP 请求转发的 IP 网络的网络地址。

单播 DHCP 转发

此方法使用 UDP 单播流量将 DHCP 请求转发到特定的 DHCP 服务器。当使用 UDP 单播时，您需要配置一个设备，这个设备为转发 DHCP 请求进行转发到在 undercloud 中进行内省的接口所分配的 IP 地址，以及 OpenStack Networking (neutron) 服务创建用于为 **ctlplane** 网络托管 DHCP 服务的网络命名空间的 IP 地址。

用于内省的接口是在 **undercloud.conf** 文件中定义为 **inspection_interface** 的接口。如果没有设置此参数，undercloud 的默认接口为 **br-ctlplane**。



注意

使用 **br-ctlplane** 接口进行内省很常见。您在 **undercloud.conf** 文件中定义为 **local_ip** 的 IP 地址位于 **br-ctlplane** 接口上。

分配给 Neutron DHCP 命名空间的 IP 地址是您为 **undercloud.conf** 文件中的 **local_subnet** 配置的 IP 范围中的第一个地址。IP 范围中的第一个地址是您在配置中定义为 **dhcp_start** 的地址。例如，如果您使用以下配置，则 **192.168.10.10** 是 IP 地址：

```
[DEFAULT]
local_subnet = leaf0
subnets = leaf0,leaf1,leaf2
```

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False
```

警告

DHCP 命名空间的 IP 地址会被自动分配。在大多数情况下，这个地址是 IP 范围中的第一个地址。要验证是否是这种情况，请在 undercloud 上运行以下命令：

```
$ openstack port list --device-owner network:dhcp -c "Fixed IP Addresses"
+-----------------------------------------------------------------------------+
| Fixed IP Addresses                                                          |
+-----------------------------------------------------------------------------+
| ip_address='192.168.10.10', subnet_id='7526fbe3-f52a-4b39-a828-
ec59f4ed12b2' |
+-----------------------------------------------------------------------------+
$ openstack subnet show 7526fbe3-f52a-4b39-a828-ec59f4ed12b2 -c name
+-------+--------+
| Field | Value  |
+-------+--------+
| name  | leaf0  |
+-------+--------+
```

dhcrelay 配置示例

在以下示例中，**dhcp** 软件包中的 **dhcrelay** 命令使用以下配置：

- 转发传入的 DHCP 请求的接口：**eth1**、**eth2** 和 **eth3**。
- 网络接口网络段上的 undercloud DHCP 服务器连接到 **eth0**。
- 用于内省的 DHCP 服务器正在侦听 IP 地址 ：**192.168.1** 680.1。
- 用于置备的 DHCP 服务器侦听 IP 地址 **192.168.10.10**。

这会生成以下 **dhcrelay** 命令：

- **dhcrelay** 版本 4.2.x:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -i eth0 -i eth1 -i eth2 -i eth3
```

- **dhcrelay** 版本 4.3.x 及更新的版本：

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -iu eth0 -id eth1 -id eth2 -id eth3
```

Cisco IOS 路由交换机配置示例

这个示例使用以下 Cisco IOS 配置来执行以下任务：

- 配置 VLAN 以用于 provisioning 网络。
- 添加叶的 IP 地址。
- 将 UDP 和 BOOTP 请求转发到侦听 IP 地址的内省 DHCP 服务器：**192.168.10.1**。
- 将 UDP 和 BOOTP 请求转发到侦听 IP 地址 **192.168.10.10** 的调配 DHCP 服务器。

```
interface vlan 2
ip address 192.168.24.254 255.255.255.0
ip helper-address 192.168.10.1
ip helper-address 192.168.10.10
!
```

现在，您已配置了 provisioning 网络，您可以配置剩余的 overcloud leaf 网络。

3.3. 为叶网络创建类别和标记节点

每个叶网络中的每个角色都需要一个类别和角色分配，以便您可以将节点标记为对应的叶。完成以下步骤以创建各个类别并将其分配到角色。

流程

1. Source **stackrc** 文件：

```
[stack@director ~]$ source ~/stackrc
```

2. 为每个自定义角色创建类别：

```
$ ROLES="control compute_leaf0 compute_leaf1 compute_leaf2 ceph-storage_leaf0 ceph-storage_leaf1 ceph-storage_leaf2"
$ for ROLE in $ROLES; do openstack flavor create --id auto --ram <ram_size_mb> --disk <disk_size_gb> --vcpus <no_vcpus> $ROLE ; done
$ for ROLE in $ROLES; do openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local" --property resources:DISK_GB='0' --property resources:MEMORY_MB='0' --property resources:VCPU='0' $ROLE ; done
```

   - 将 **<ram_size_mb>** 替换为裸机节点的 RAM，以 MB 为单位。
   - 将 **<disk_size_gb>** 替换为裸机节点中的磁盘大小（以 GB 为单位）。
   - 将 **<no_vcpus>** 替换为裸机节点中的 CPU 数量。

3. 检索节点列表来识别它们的 UUID：

```
(undercloud)$ openstack baremetal node list
```

4. 使用自定义资源类将每个裸机节点标记为其叶网络和角色：

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.LEAF-ROLE <node>
```

将 **<node>** 替换为裸机节点的 ID。

例如，输入以下命令将带有 UUID **58c3d07e-24f2-48a7-bbb6-6843f0e8ee13** 的节点标记为 Leaf2 上的 Compute 角色：

```
(undercloud)$ openstack baremetal node set \
 --resource-class baremetal.COMPUTE-LEAF2 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13
```

5. 将每个叶网络角色类别与自定义资源类关联：

```
(undercloud)$ openstack flavor set \
 --property resources:CUSTOM_BAREMETAL_LEAF_ROLE=1 \
<custom_role>
```

要确定与裸机置备服务节点的资源类对应的自定义资源类的名称，请将资源类转换为大写，请将每个 punctuation 标记替换为下划线，并将前缀替换为 **CUSTOM_**。

注意

类别只能请求一个裸机资源类实例。

6. 在 **node-info.yaml** 文件中，指定要用于每个自定义叶角色的类别，以及为每个自定义叶角色分配的节点数量。例如，以下配置指定要使用的类别，以及为 **compute_leaf0**, **compute_leaf1**, **compute_leaf2**, **ceph-storage_leaf0**, **ceph-storage_leaf1**, 和 **ceph-storage_leaf2** 自定义叶角色分配的节点数量：

```
parameter_defaults:
  OvercloudControllerFlavor: control
  OvercloudComputeLeaf0Flavor: compute_leaf0
  OvercloudComputeLeaf1Flavor: compute_leaf1
  OvercloudComputeLeaf2Flavor: compute_leaf2
  OvercloudCephStorageLeaf0Flavor: ceph-storage_leaf0
  OvercloudCephStorageLeaf1Flavor: ceph-storage_leaf1
  OvercloudCephStorageLeaf2Flavor: ceph-storage_leaf2
  ControllerLeaf0Count: 3
  ComputeLeaf0Count: 3
  ComputeLeaf1Count: 3
  ComputeLeaf2Count: 3
  CephStorageLeaf0Count: 3
  CephStorageLeaf1Count: 3
  CephStorageLeaf2Count: 3
```

3.4. 将裸机节点端口映射到 CONTROL PLANE 网络片段

要在 L3 路由网络中启用部署，您必须在裸机端口上配置 **physical_network** 字段。每个裸机端口都与 OpenStack Bare Metal (ironic)服务中的裸机节点关联。物理网络名称是您在 undercloud 配置中的 **subnets** 选项中包含的名称。

注意

在 **undercloud.conf** 文件中指定为 **local_subnet** 子网的物理网络名称始终被命名为 **ctlplane**。

流程

1. Source **stackrc** 文件：

```
$ source ~/stackrc
```

2. 检查裸机节点：

```
$ openstack baremetal node list
```

3. 确保裸机节点处于 Register 或 **manageable** 状态。如果裸机节点不在其中一个状态中，在 baremetal 端口上设置 **physical_network** 属性的命令会失败。要将所有节点设置为 **manageable** 状态，请运行以下命令：

```
$ for node in $(openstack baremetal node list -f value -c Name); do openstack baremetal node manage $node --wait; done
```

4. 检查哪个 baremetal 端口与哪个 baremetal 节点关联：

```
$ openstack baremetal port list --node <node-uuid>
```

5. 为端口设置 **physical-network** 参数。在以下示例中，在配置中定义三个子网：**leaf0**, **leaf1**, 和 **leaf2**。local_subnet 是 **leaf0**。由于 **local_subnet** 的物理网络始终为 **ctlplane**，因此连接到 **leaf0** 的 baremetal 端口使用 ctlplane。剩余的端口使用其他 leaf 名称：

```
$ openstack baremetal port set --physical-network ctlplane <port-uuid>
$ openstack baremetal port set --physical-network leaf1 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
```

6. 在部署 overcloud 前内省节点。包含 **--all-manageable** 和 **--provide** 选项来设置可用于部署的节点：

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

3.5. 将一个新的叶添加到 SPINE-LEAF PROVISIONING 网络

在增加可添加新的物理站点的网络容量时，您可能需要向 Red Hat OpenStack Platform spine-leaf provisioning 网络添加新的叶和对应的子网。在 overcloud 上置备叶时，会使用对应的 undercloud leaf。

先决条件

- 您的 RHOSP 部署使用 spine-leaf 网络拓扑。

流程

1. 以 stack 用户身份登录 undercloud 主机。

2. *查找 undercloud 凭证文件：*

```
$ source ~/stackrc
```

3. *在 **/home/stack/undercloud.conf** 文件中执行以下操作：*

   a. *找到 **subnets** 参数，并为要添加的叶添加新子网。*
   子网代表路由的 spine 和 leaf 中的 L2 片段：

   示例

   *在本例中，为新叶(**leaf3**)添加新子网(**leaf3**)：*

   ```
   subnets = leaf0,leaf1,leaf2,leaf3
   ```

   b. *为您添加的子网创建一个部分。*

   示例

   *在本例中，为新子网添加了 **[leaf3]** 部分(**leaf3**)：*

   ```
   [leaf0]
   cidr = 192.168.10.0/24
   dhcp_start = 192.168.10.10
   dhcp_end = 192.168.10.90
   inspection_iprange = 192.168.10.100,192.168.10.190
   gateway = 192.168.10.1
   masquerade = False

   [leaf1]
   cidr = 192.168.11.0/24
   dhcp_start = 192.168.11.10
   dhcp_end = 192.168.11.90
   inspection_iprange = 192.168.11.100,192.168.11.190
   gateway = 192.168.11.1
   masquerade = False

   [leaf2]
   cidr = 192.168.12.0/24
   dhcp_start = 192.168.12.10
   dhcp_end = 192.168.12.90
   inspection_iprange = 192.168.12.100,192.168.12.190
   gateway = 192.168.12.1
   masquerade = False

   [leaf3]
   cidr = 192.168.13.0/24
   dhcp_start = 192.168.13.10
   dhcp_end = 192.168.13.90
   inspection_iprange = 192.168.13.100,192.168.13.190
   gateway = 192.168.13.1
   masquerade = False
   ```

4. *保存 **undercloud.conf** 文件。*

5. *重新安装 undercloud：*

```
$ openstack undercloud install
```

其他资源

- *在 spine-leaf 部署中添加新叶*

第 4 章 为 DCN 部署准备 OVERCLOUD 模板

4.1. 使用独立 HEAT 堆栈的先决条件

在使用单独的 heat 堆栈创建部署前，您的环境必须满足以下先决条件：

- 正常工作的 Red Hat OpenStack Platform 16 undercloud。
- 对于 Ceph Storage 用户：访问 Red Hat Ceph Storage 4。
- 对于中央位置：三个能够充当中央 Controller 节点的节点。所有这三个 Controller 节点都必须位于同一 heat 堆栈中。您不能将 Controller 节点或任何 control plane 服务分割到单独的 heat 堆栈。
- 如果您计划在边缘部署 Ceph 存储，Ceph 存储是中央位置的要求。
- 对于每个额外的 DCN 网站：三个 HCI 计算节点。
- 所有节点都必须预先置备，或者从中央部署网络进行 PXE 引导。您可以使用 DHCP 转发来为 DCN 启用这个连接。
- 所有节点都由 ironic 内省。
- 红帽建议将 <role>HostnameFormat 参数保留为默认值：%stackname%-<role>-%index%。如果没有包含 %stackname% 前缀，则您的 overcloud 在不同的堆栈中为分布式计算节点使用相同的主机名。确保分布式计算节点使用 %stackname% 前缀来区分节点与不同边缘站点的节点。例如，如果您部署两个名为 **dcn0** 和 **dcn1** 的边缘站点，则堆栈名称前缀可帮助您在 undercloud 上运行 **openstack server list** 命令时区分 dcn0-distributedcompute-0 和 dcn1-distributedcompute-0。
- 提供 **centralrc** 身份验证文件，以在边缘站点和中央位置调度工作负载。您不需要为边缘站点自动生成身份验证文件。

4.2. 独立 HEAT 堆栈部署示例的限制

本文档提供了在 Red Hat OpenStack Platform 上使用单独的 heat 堆栈的示例部署。这个示例环境有以下限制：

- spine/Leaf 网络 - 本指南中的示例不演示路由要求，这是分布式计算节点(DCN)部署中所需的路由要求。
- Ironic DHCP Relay - 本指南不包括如何使用 DHCP 转发配置 Ironic。

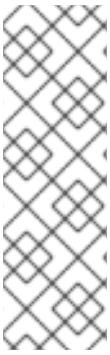
4.3. 设计单独的 HEAT 堆栈部署

要在单独的 heat 堆栈内分段部署，您必须首先使用 control plane 部署单个 overcloud。然后，您可以为分布式计算节点(DCN)站点创建单独的堆栈。以下示例显示了不同节点类型的独立堆栈：

- Controller 节点：名为 **central** 的独立 heat 堆栈，例如部署控制器。为 DCN 站点创建新的 heat 堆栈时，您必须使用 **中央** 堆栈中的数据创建它们。Controller 节点必须可用于任何实例管理任务。
- DCN 站点：您可以有单独的、唯一命名的 heat 堆栈，如 **dcn** 0、**dcn1** 等等。使用 DHCP 转发将 provisioning 网络扩展到远程站点。

注意

您必须为每个堆栈创建单独的可用区(AZ)。



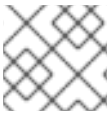
注意

*如果使用 spine/leaf 网络，则必须使用特定格式来定义 **Storage** 和 **StorageMgmt** 网络，以便 ceph-ansible 正确配置 Ceph 以使用这些网络。将 **Storage** 和 **StorageMgmt** 网络定义为覆盖值，并使用单引号括起值。在以下示例中，存储网络（称为 **public_network**）跨越两个子网，用逗号分开，并使用单引号括起来：*

```
CephAnsibleExtraConfig:
  public_network: '172.23.1.0/24,172.23.2.0/24'
```

4.4. 在多个堆栈中重复使用网络资源

*您可以将多个堆栈配置为使用相同的网络资源，如 VIP 和子网。您可以使用 **ManageNetworks** 设置或 **external_resource** the 字段在堆栈之间重复网络资源。*



注意

*如果您使用 **external_resource** 114 字段，请不要使用 **ManageNetworks** 设置。*

*如果您没有在堆栈间重复使用网络，在 **network_data.yaml** 中定义的每个网络都必须在所有部署的堆栈中具有唯一的名称。例如，除非打算在堆栈之间共享网络，否则网络名称 **internal_api** 无法重复使用。为网络指定不同的 name 和 **name_lower** 属性，如 **InternalApiCompute0** 和 **internal_api_compute_0**。*

4.5. 使用 MANAGENETWORKS 来重复利用网络资源

*使用 **ManageNetworks** 设置时，多个堆栈可以使用相同的 **network_data.yaml** 文件，并且该设置全局应用于所有网络资源。**network_data.yaml** 文件定义堆栈使用的网络资源：*

```
- name: StorageBackup
  vip: true
  name_lower: storage_backup
  ip_subnet: '172.21.1.0/24'
  allocation_pools: [{'start': '171.21.1.4', 'end': '172.21.1.250'}]
  gateway_ip: '172.21.1.1'
```

*将 ManageNetworks 设置为 false 时，节点将使用已在 **中央** 堆栈中创建的现有网络。*

使用以下序列，以便新堆栈不管理现有的网络资源。

流程

1. *使用 **ManageNetworks: true** 部署中央堆栈，或保留为未设置。*
2. *使用 **ManageNetworks: false** 部署额外的堆栈。*

*当您添加新网络资源时，例如，当您在 spine/leaf 部署中添加新保留时，您必须使用新的 **network_data.yaml** 更新中央堆栈。这是因为中央堆栈仍然拥有和管理网络资源。在中央堆栈中可用网络资源后，您可以部署额外的堆栈以使用它们。*

4.6. 使用 UUID 来重复利用网络资源

如果需要更多控制堆栈间重复使用哪些网络，您可以在 network_data.yaml 文件中将 **external_resource** bang 字段用于 **network_data.yaml** 文件中的资源，包括网络、子网、网段或 VIP。这些资源标记为外部管理，heat 不会对其执行任何创建、更新或删除操作。

在 **network_data.yaml** 文件中为每个所需的网络定义添加一个条目。然后，该资源可用于在单独的堆栈上部署：

```
external_resource_network_id: Existing Network UUID
external_resource_subnet_id: Existing Subnet UUID
external_resource_segment_id: Existing Segment UUID
external_resource_vip_id: Existing VIP UUID
```

本例重复使用单独的堆栈中的 control plane 堆栈的 **internal_api** 网络。

流程

1. 识别相关网络资源的 UUID：

```
$ openstack network show internal_api -c id -f value
$ openstack subnet show internal_api_subnet -c id -f value
$ openstack port show internal_api_virtual_ip -c id -f value
```

2. 保存以上命令的输出中显示的值，并将它们添加到单独堆栈的 **network_data.yaml** 文件中的 **internal_api** 网络的网络定义中：

```
- name: InternalApi
  external_resource_network_id: 93861871-7814-4dbc-9e6c-7f51496b43af
  external_resource_subnet_id: c85c8670-51c1-4b17-a580-1cfb4344de27
  external_resource_vip_id: 8bb9d96f-72bf-4964-a05c-5d3fed203eb7
  name_lower: internal_api
  vip: true
  ip_subnet: '172.16.2.0/24'
  allocation_pools: [{'start': '172.16.2.4', 'end': '172.16.2.250'}]
  ipv6_subnet: 'fd00:fd00:fd00:2000::/64'
  ipv6_allocation_pools: [{'start': 'fd00:fd00:fd00:2000::10', 'end':
'fd00:fd00:fd00:2000:ffff:ffff:ffff:fffe'}]
  mtu: 1400
```

4.7. 管理独立的 HEAT 堆栈

本指南中的步骤演示了如何为三个 heat 堆栈组织环境文件：**central**、**dcn0** 和 **dcn1**。红帽建议将每个 heat 堆栈的模板存储在一个单独的目录中，以保持与每个部署相关的信息。

流程

1. 定义 **中央** heat 堆栈：

```
$ mkdir central
$ touch central/overrides.yaml
```

2. 将 **中央** heat 堆栈中的数据提取到所有 DCN 站点的通用目录中：

```
$ mkdir dcn-common
$ touch dcn-common/overrides.yaml
$ touch dcn-common/central-export.yaml
```

central-export.yaml 文件由 **openstack overcloud export** 命令创建。它位于 **dcn-common** 目录中，因为本指南中的所有 DCN 部署都必须使用此文件。

3. 定义 **dcn0** 站点。

```
$ mkdir dcn0
$ touch dcn0/overrides.yaml
```

要部署更多 DCN 站点，请按数字创建额外的 **dcn** 目录。

注意

该触点用于提供文件组织示例。每个文件必须包含成功部署的适当内容。

4.8. 检索容器镜像

使用以下步骤及其示例文件内容来检索使用单独的 heat 堆栈部署所需的容器镜像。您必须通过运行带有边缘站点的环境文件的 **openstack container image prepare** 命令来确保包含可选或特定于边缘服务的容器镜像。

如需更多信息，请参阅 准备容器镜像。

流程

1. 将 Registry 服务帐户凭证添加到 **containers.yaml**。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ceph_namespace: registry.redhat.io/rhceph
      ceph_image: rhceph-4-rhel8
      ceph_tag: latest
      name_prefix: openstack-
      namespace: registry.redhat.io/rhosp16-rhel8
      tag: latest
  ContainerImageRegistryCredentials:
    # https://access.redhat.com/RegistryAuthentication
    registry.redhat.io:
      registry-service-account-username: registry-service-account-password
```

2. 将环境文件生成为 **images-env.yaml** ：

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file images-env.yaml
```

生成的 **images-env.yaml** 文件作为生成它的堆栈的 overcloud 部署流程的一部分。

4.9. 为边缘创建快速数据路径角色

要在边缘上使用快速数据路径服务，您必须创建一个自定义角色来定义快速数据路径和边缘服务。为部署创建角色文件时，您可以包括新创建的角色来定义分布式计算节点架构和快速数据路径服务（如 DPDK 或 SR-IOV）所需的服务。

例如，使用 DPDK 为 distributedCompute 创建自定义角色：

先决条件

成功安装 undercloud。如需更多信息，请参阅安装 undercloud。

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 复制 **默认角色** 目录：

```
cp -r /usr/share/openstack-tripleo-heat-templates/roles ~/.
```

3. 从 **DistributedCompute.yaml** 文件中创建一个名为 **DistributedComputeDpdk.yaml** 的新文件：

```
cp roles/DistributedCompute.yaml roles/DistributedComputeDpdk.yaml
```

4. 将 DPDK 服务添加到新的 **DistributedComputeDpdk.yaml** 文件中。您可以通过在 **ComputeOvsDpdk.yaml** 文件中识别没有存在于 **DistributedComputeDpdk.yaml** 文件中的的参数来识别需要添加的参数。

```
diff -u roles/DistributedComputeDpdk.yaml roles/ComputeOvsDpdk.yaml
```

在输出中，ComputeOvsDpdk.yaml 文件前面带有 **+** 的参数存在于 DistributedComputeDpdk.yaml 文件中。将这些参数包含在新的 **DistributedComputeDpdk.yaml** 文件中。

5. 使用 **DistributedComputeDpdk.yaml** 创建 **DistributedComputeDpdk** 角色文件：

```
openstack overcloud roles generate --roles-path ~/roles/ -o ~/roles/roles-custom.yaml
DistributedComputeDpdk
```

您可以使用同样的方法为 SR-IOV 创建快速数据路径角色，或 SR-IOV 和 DPDK 的组合来满足您的要求。

其它资源

- 创建自定义角色
- 支持的自定义角色

如果您计划在没有块存储的情况下部署边缘站点，请参阅以下内容：

- 第 5 章 安装中央位置
- 第 6.1 节 "在没有存储的情况下部署边缘节点"

如果您计划使用 Red Hat Ceph Storage 部署边缘站点，请参阅以下内容：

- *第 5 章 安装中央位置*
- *第 7.1 节 “使用存储部署边缘站点”*

第 5 章 安装中央位置

当您为分布式计算节点(DCN)架构部署中央位置时，您可以部署集群：

- 使用或不使用 Compute 节点
- 使用或不使用 Red Hat Ceph Storage

如果您在中央位置没有 Red Hat Ceph Storage 部署 Red Hat OpenStack Platform，则无法使用 Red Hat Ceph Storage 部署任何边缘站点。此外，您无法选择通过重新部署将 Red Hat Ceph Storage 添加到中央位置。

5.1. 在没有边缘存储的情况下部署中央控制器

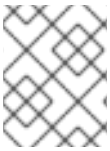
如果您使用 Object Storage 服务(swift)作为中央位置镜像服务(glance)的后端，您可以在边缘站点部署没有块存储的分布式计算节点集群。因为每个架构的不同角色和网络配置集，以后无法更新没有块存储的站点，使其具有块存储。

重要：以下步骤使用 lvm 作为 Cinder 的后端，在生产环境中不支持。您必须将经认证的块存储解决方案部署为 Cinder 的后端。

以类似于典型的 overcloud 部署的方式部署中央控制器集群。此集群不需要任何 Compute 节点，因此您可以将 Compute 数量设置为 **0** 来覆盖默认值 **1**。Central 控制器具有特定存储和 Oslo 配置要求。使用以下步骤满足这些要求。

流程

以下流程概述了中央位置初始部署的步骤。



注意

以下步骤详细介绍了与 DCN 部署关联的部署命令和环境文件，而无需 glance 多存储。这些步骤不包括不相关的、但在网络等配置的各个方面。

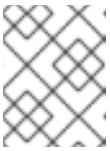
1. 在主目录中，为您计划部署的每个堆栈创建目录。

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1
```

2. 使用类似如下的设置，创建一个名为 **central/overrides.yaml** 的文件：

```
parameter_defaults:
  NtpServer:
    - 0.pool.ntp.org
    - 1.pool.ntp.org
  ControllerCount: 3
  ComputeCount: 0
  OvercloudControllerFlavor: baremetal
  OvercloudComputeFlavor: baremetal
  ControllerSchedulerHints:
    'capabilities:node': '0-controller-%index%'
  GlanceBackend: swift
```

- **ControllerCount：3** 指定将部署三个节点。这些会将 swift 用于 glance，lvm 用于 cinder，并为边缘计算节点托管 control-plane 服务。
- **ComputeCount: 0** 是一个可选参数，可防止使用中央 Controller 节点部署 Compute 节点。
- **GlanceBackend: swift** 使用 Object Storage (swift) 作为镜像服务 (glance) 后端。
  生成的配置通过以下方式与分布式计算节点 (DCN) 交互：
- DCN 上的镜像服务创建从中央对象存储后端接收的镜像的缓存副本。镜像服务使用 HTTP 将镜像从对象存储复制到本地磁盘缓存中。



注意

中央 Controller 节点必须能够连接到分布式计算节点 (DCN) 站点。中央 Controller 节点可以使用路由层 3 连接。

3. 使用适合您的环境的角色为中央位置生成角色：

```
openstack overcloud roles generate Controller \
-o ~/central/control_plane_roles.yaml
```

4. 生成环境文件 **~/central/central-images-env.yaml**：

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/central/central-images-env.yaml
```

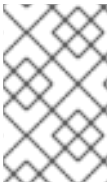
5. 在 **site-name.yaml** 环境文件中为站点配置命名约定。Nova 可用区 Cinder 存储可用区必须匹配：

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: central
EOF
```

6. 部署中央 Controller 节点。例如，您可以使用包含以下内容的 **deploy.sh** 文件：

```
#!/bin/bash

source ~/stackrc
time openstack overcloud deploy \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e ~/central/containers-env-file.yaml \
-e ~/central/overrides.yaml \
-e ~/central/site-name.yaml
```



注意

您必须在 **openstack overcloud deploy** 命令中包含用于配置网络配置的 heat 模板。为边缘架构设计需要 spine 和 leaf 网络。如需了解更多详细信息，请参阅 Spine Leaf Networking。

5.2. 使用存储部署中央站点

要使用多个存储部署镜像服务，并将 Ceph Storage 用作后端，请完成以下步骤：

先决条件

- 位于中央位置和每个可用区的 Ceph 集群的硬件，或者在需要存储服务的每个地理位置。
- 位于中央位置和每个可用区的三个镜像服务服务器的硬件，或者在需要存储服务的每个地理位置。

以下是部署两个或多个堆栈的示例：

- 一个堆栈位于中央位置，称为 **central**。
- 一个堆栈位于名为 **dcn0** 的边缘站点。
- 部署的其他堆栈与 **dcn0** 类似，如 **dcn1**、 **dcn2** 等。

流程

以下流程概述了中央位置初始部署的步骤。

注意

以下步骤详细介绍了与使用多个存储的镜像服务的 DCN 部署关联的部署命令和环境文件。这些步骤不包括不相关的、但在网络等配置的各个方面。

1. 在主目录中，为您计划部署的每个堆栈创建目录。

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1
```

2. 设置 Ceph 集群的名称，以及相对于可用硬件的配置参数。如需更多信息，请参阅使用自定义配置设置配置 Ceph ：

```
cat > /home/stack/central/ceph.yaml << EOF
parameter_defaults:
  CephClusterName: central
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
    devices:
      - /dev/sda
      - /dev/sdb
  CephPoolDefaultSize: 3
```

```
  CephPoolDefaultPgNum: 128

EOF
```

3. 使用适合您的环境的角色为中央位置生成角色：

```
openstack overcloud roles generate Compute Controller CephStorage \
-o ~/central/central_roles.yaml

cat > /home/stack/central/role-counts.yaml << EOF
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 2
  CephStorage: 3
EOF
```

4. 生成环境文件 ~/**central/central-images-env.yaml**

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/central/central-images-env.yaml
```

5. 在 **site-name.yaml** 环境文件中为站点配置命名约定。Nova 可用区和 Cinder 存储可用区必须匹配：

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
    NovaComputeAvailabilityZone: central
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: central
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: central
    GlanceBackendID: central
EOF
```

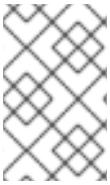
6. 使用类似如下的内容配置 **glance.yaml** 模板：

```
parameter_defaults:
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceBackend: rbd
    GlanceStoreDescription: 'central rbd glance store'
    GlanceBackendID: central
    CephClusterName: central
```

7. 准备所有其他模板后，部署 **中央** 堆栈：

```
openstack overcloud deploy \
      --stack central \
      --templates /usr/share/openstack-tripleo-heat-templates/ \
      -r ~/central/central_roles.yaml \
   ...
      -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
      -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
```

```
-e ~/central/central-images-env.yaml \
-e ~/central/role-counts.yaml \
-e ~/central/site-name.yaml \
-e ~/central/ceph.yaml \
-e ~/central/glance.yaml
```



注意

您必须在 **openstack overcloud deploy** 命令中包含用于配置网络配置的 heat 模板。为边缘架构设计需要 spine 和 leaf 网络。如需了解更多详细信息，请参阅 Spine Leaf Networking。

ceph-ansible.yaml 文件配置有以下参数：

- NovaEnableRbdBackend: true
- GlanceBackend: rbd

当您将这些设置一起使用时，glance.conf 参数 **image_import_plugins** 由 heat 配置成具有值 **image_conversion**，使用 **glance image-create-via-import --disk-format qcow2** 等命令自动化 QCOW2 镜像转换。

这是 Ceph RBD 的最佳选择。如果要禁用镜像转换，请使用 **GlanceImageImportPlugin** 参数：

```
parameter_defaults:
  GlanceImageImportPlugin: []
```

5.3. 集成外部 CEPH

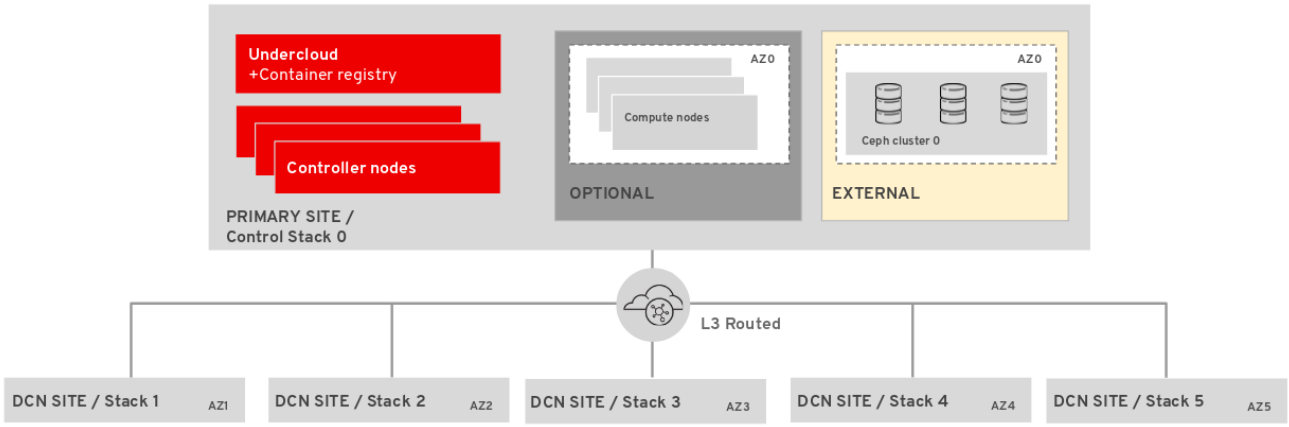
您可以部署分布式计算节点(DCN)架构的中心位置，并集成预部署的 Red Hat Ceph Storage 解决方案。

先决条件

- 位于中央位置和每个可用区的 Ceph 集群的硬件，或者在需要存储服务的每个地理位置。
- 位于中央位置和每个可用区的三个镜像服务服务器的硬件，或者在需要存储服务的每个地理位置。

以下是部署两个或多个堆栈的示例：

- 一个堆栈位于中央位置，称为 **central**。
- 一个堆栈位于名为 **dcn0** 的边缘站点。
- 部署的其他堆栈与 **dcn0** 类似，如 **dcn1**、 **dcn2** 等。



您可以按照将 Overcloud 与现有 Red Hat Ceph Cluster 集成的步骤，安装中央位置，使其与预先存在的 Red Hat Ceph Storage 解决方案集成。将 Red Hat Ceph Storage 与 DCN 部署的中央站点集成没有特殊要求，但在部署 overcloud 前，您仍需要完成 DCN 具体步骤：

1. *在主目录中，为您计划部署的每个堆栈创建目录。使用它来分隔针对其相应站点设计的模板。*

```
mkdir /home/stack/central
mkdir /home/stack/dcn0
mkdir /home/stack/dcn1
```

2. *使用 RHOSP director 管理的角色为中央位置生成角色。与外部 Ceph 集成时，请不要使用 Ceph 角色：*

```
cat > /home/stack/central/role-counts.yaml << EOF
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 2
EOF
```

3. *生成环境文件 ~/**central/central-images-env.yaml** ：*

```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/central/central-images-env.yaml
```

4. *在 **site-name.yaml** 环境文件中为站点配置命名约定。Compute (nova) 可用区和块存储(cinder) 可用区必须匹配：*

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
    NovaComputeAvailabilityZone: central
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: central
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: central
    GlanceBackendID: central
EOF
```

5. *使用类似如下的内容配置 **glance.yaml** 模板：*

```
parameter_defaults:
```

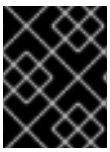
```
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceBackend: rbd
    GlanceStoreDescription: 'central rbd glance store'
    GlanceBackendID: central
    CephClusterName: central
```

6. *当在没有 Red Hat OpenStack Platform director 的情况下部署 Ceph 时，请不要使用* ***ceph-ansible.yaml*** *环境文件。改为使用* ***ceph-ansible-external.yaml*** *环境文件。*

```
openstack overcloud deploy \
      --stack central \
      --templates /usr/share/openstack-tripleo-heat-templates/ \
      -r ~/central/central_roles.yaml \
   ...
      -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-
external.yaml \
      -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
      -e ~/central/central-images-env.yaml \
      -e ~/central/role-counts.yaml \
      -e ~/central/site-name.yaml \
      -e ~/central/ceph.yaml \
      -e ~/central/glance.yaml
```

第 6 章 在没有存储的情况下部署边缘

如果您使用 Object Storage 服务(swift)作为中央位置镜像服务(glance)的后端，您可以在边缘站点部署没有块存储的分布式计算节点集群。因为每个架构的不同角色和网络配置集，以后无法更新没有块存储的站点，使其具有块存储。



重要

以下流程使用 lvm 作为块存储服务(cinder)的后端，在生产环境中不支持。您必须将经认证的块存储解决方案部署为块存储服务的后端。

6.1. 在没有存储的情况下部署边缘节点

您可以部署使用中央位置作为 control plane 的边缘计算节点。此流程演示了如何向部署添加新的 DCN 堆栈，并重复使用现有 heat 堆栈的配置来创建新环境文件。第一个 heat 堆栈在中央数据中心中部署 overcloud。创建额外的 heat 堆栈，将计算节点部署到远程位置。

6.1.1. 配置分布式计算节点环境文件

此流程创建一个新的 **central-export.yaml** 环境文件，并使用 overcloud 的 **plan-environment.yaml** 文件中的密码。**central-export.yaml** 文件包含敏感的安全数据。要提高安全性，您可以在不再需要该文件时删除该文件。

当您为 **--config-download-dir** 选项指定目录时，请使用 director 部署期间在 **/var/lib/mistral** 中创建的中央 hub Ansible 配置。不要使用您在 **openstack overcloud config download** 命令中使用手动生成的 Ansible 配置。手动生成的配置缺少了仅在部署操作期间创建的某些文件。

您必须将镜像上传到中央位置，然后才能将镜像复制到边缘站点；每个镜像的副本必须存在于中央位置的镜像服务(glance)中。

您必须将 RBD 存储驱动程序用于镜像、计算和块存储服务。

流程

1. 生成 DCN 站点所需的配置文件：

```
openstack overcloud export \
--config-download-dir /var/lib/mistral/central \
--stack central --output-file ~/dcn-common/central-export.yaml
```

2. 使用适合您的环境的角色为边缘位置生成角色：

```
openstack overcloud roles generate Compute -o ~/dcn0/dcn0_roles.yaml
```

注意

如果将 ML2/OVS 用于网络覆盖，您必须编辑您创建的角色文件，使其包含 **NeutronDhcpAgent** 和 **NeutronMetadataAgent** 角色：

```
...
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
+   - OS::TripleO::Services::NeutronDhcpAgent
+   - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaAZConfig
    - OS::TripleO::Services::NovaCompute

...
```

如需更多信息，请参阅准备路由的供应商网络。

6.1.2. 将 Compute 节点部署到 DCN 站点

此流程使用 **Compute** 角色将 Compute 节点部署到名为 **dcn0** 的可用区(AZ)中。在分布式计算节点(DC)上下文中，此角色用于没有存储的站点。

流程

1. 查看 dcn0/overrides.yaml 中分布式计算(DCN)站点的覆盖

```
parameter_defaults:
  ComputeCount: 3
  ComputeFlavor: baremetal
  ComputeSchedulerHints:
    'capabilities:node': '0-compute-%index%'
  NovaAZAttach: false
```

2. 在 ~/**dcn0** 目录中创建一个名为 **site-name.yaml** 的新文件，其内容如下：

```
resource_registry:
  OS::TripleO::Services::NovaAZConfig: /usr/share/openstack-tripleo-heat-
templates/deployment/nova/nova-az-config.yaml
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  RootStackName: dcn0
```

3. 检索 DCN 站点的容器镜像：

```
sudo openstack tripleo container image prepare \
--environment-directory dcn0 \
-r ~/dcn0/roles_data.yaml \
-e ~/dcn-common/central-export.yaml \
-e ~/containers-prepare-parameter.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

4. 为 dcn0 运行 deploy.sh 部署脚本：

```
#!/bin/bash
STACK=dcn0
source ~/stackrc
time openstack overcloud deploy \
    --stack $STACK \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
    -e ~/dcn-common/central-export.yaml \
    -e ~/dcn0/dcn0-images-env.yaml \
    -e ~/dcn0/site-name.yaml \
    -e ~/dcn0/overrides.yaml
```

*如果部署需要编辑 **network_data.yaml** 文件的额外边缘站点，您必须在中央位置执行堆栈更新。*

5. *在部署了边缘位置后，您必须确保在 nova API 数据库中创建 nova **cell_v2** 主机映射。在 undercloud 上运行以下命令：*

```
TRIPLEO_PLAN_NAME=central \
ansible -i /usr/bin/tripleo-ansible-inventory \
nova_api[0] -b -a \
"{{ container_cli }} exec -it nova_api \
nova-manage cell_v2 discover_hosts --by-service --verbose"
```

如果扩展边缘站点，您必须再次运行该命令。

注意

*您必须在 **openstack overcloud deploy** 命令中包含用于配置网络配置的 heat 模板。为边缘架构设计需要 spine 和 leaf 网络。如需了解更多详细信息，请参阅 Spine Leaf Networking。*

6.2. 边缘排除特定镜像类型

*默认情况下，Compute 节点公告它们支持的所有镜像格式。如果您的 Compute 节点没有使用 Ceph 存储，您可以从镜像格式公告中排除 RAW 镜像。RAW 镜像格式会消耗比 QCOW2 镜像更多的网络带宽和本地存储，并在没有 Ceph 存储的边缘站点中使用效率低。使用 **NovaImageTypeExcludeList** 参数排除特定的镜像格式：*

重要

不要在边缘站点与 Ceph 搭配使用这个参数，因为 Ceph 需要 RAW 镜像。

注意

没有公告 RAW 镜像的计算节点无法托管从 RAW 镜像创建的实例。这可能会影响 snapshot-redeploy 和 shelving。

先决条件

- *安装了 Red Hat OpenStack Platform director*
- *已安装中央位置*
- *Compute 节点可用于 DCN 部署*

流程

1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** 凭证文件：

```
$ source ~/stackrc
```

3. 在自定义模板中包含 **NovaImageTypeExcludeList** 参数：

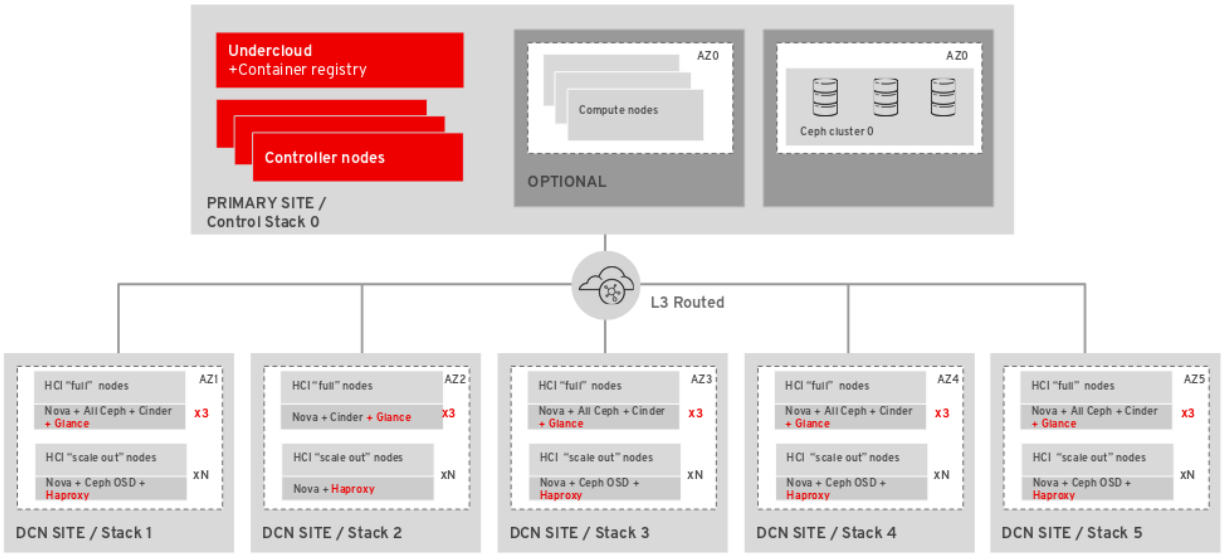
```
parameter_defaults:
  NovaImageTypeExcludeList:
      - raw
```

4. 在 overcloud 部署命令中包含 **NovaImageTypeExcludeList** 参数的环境文件，以及与部署相关的任何其他环境文件：

```
openstack overcloud deploy --templates \
-n network_data.yaml \
-r roles_data.yaml \
-e <environment_files> \
-e <new_environment_file>
```

第 7 章 在边缘部署存储

您可以使用 Red Hat OpenStack Platform director 扩展分布式计算节点部署，以使用 Red Hat OpenStack Platform 和 Ceph Storage 的优点在边缘包含分布式镜像和持久性存储。



7.1. 使用存储部署边缘站点

部署中心站点后，构建边缘站点并确保每个边缘位置主要连接到自己的存储后端，以及中央位置的存储后端。

*spine 和 leaf network configuration 应当包含在这一配置中，以及 ceph 需要的 **storage** 和 **storage_mgmt** 网络。如需更多信息，请参阅 Spine leaf 网络。*

您必须在中央位置和存储网络之间具有连接，以便您可以在站点之间移动 glance 镜像。

*确保中央位置可以在每个边缘站点与 **mons** 和 **osds** 通信。但是，您应该以站点位置边界终止存储管理网络，因为存储管理网络用于 OSD 重新平衡。*

流程

1. ***从中央** 堆栈导出堆栈信息。在运行此命令前，您必须部署 **中央** 堆栈：*

```
openstack overcloud export \
    --config-download-dir /var/lib/mistral/central/ \
    --stack central \
    --output-file ~/dcn-common/central-export.yaml
```

注意

***config-download-dir** 值默认为 **/var/lib/mistral/<stack>/**。*

2. *创建 **central_ceph_external.yaml** 文件。此环境文件将 DCN 站点连接到中央 hub Ceph 集群，因此信息特定于前面步骤中部署的 Ceph 集群。*

```
sudo -E openstack overcloud export ceph \
--stack central \
```

```
--config-download-dir /var/lib/mistral \
--output-file ~/dcn-common/central_ceph_external.yaml
```

当在没有 Red Hat OpenStack Platform director 的情况下部署 Ceph 时，无法运行 ***openstack overcloud export ceph*** *命令。手动创建* ***central_ceph_external.yaml*** *文件：*

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
          key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
          mode: "0600"
      dashboard_enabled: false
      ceph_conf_overrides:
        client:
          keyring: /etc/ceph/central.client.openstack.keyring
```

- ***fsid*** *参数是 Ceph Storage 集群的文件系统 ID：这个值在* ***[global]*** *部分的集群配置文件中指定：*

  ```
  [global]
  fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
  ...
  ```

- ***key*** *参数是 openstack 帐户的 ceph 客户端密钥：*

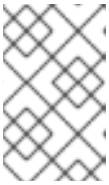
  ```
  [root@ceph ~]# ceph auth list
  ...
  [client.openstack]
      key = AQC+vYNXgDAgAhAAc8UoYt+OTz5uhV7ItLdwUw==
      caps mgr = "allow *"
      caps mon = "profile rbd"
      caps osd = "profile rbd pool=volumes, profile rbd pool=vms, profile rbd pool=images,
  profile rbd pool=backups, profile rbd pool=metrics"
  ...
  ```

  有关示例 ***central_ceph_external.yaml*** *文件中显示的参数的更多信息，请参阅 创建自定义环境文件。*

3. *为镜像服务配置覆盖创建* ***~/dcn0/glance.yaml*** *文件：*

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'dcn0 rbd glance store'
  GlanceBackendID: dcn0
  GlanceMultistoreConfig:
    central:
```

```
    GlanceBackend: rbd
    GlanceStoreDescription: 'central rbd glance store'
    CephClusterName: central
    GlanceRbdPoolName: images
    CephClientUserName: openstack
```



注意

如果您没有将 **GlanceRbdPoolName** 和 **CephClientUserName** 参数用于 glance 多存储配置，则这些值将从您用于配置中央位置的参数继承。这些值可能不同，可能会导致部署失败。

4. 使用与可用硬件相关的配置参数配置 **ceph.yaml** 文件。

```
cat > /home/stack/dcn0/ceph.yaml << EOF
parameter_defaults:
  CephClusterName: dcn0
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
    devices:
      - /dev/sda
      - /dev/sdb
  CephPoolDefaultSize: 3
  CephPoolDefaultPgNum: 128
EOF
```

有关更多信息，请参阅 映射 Ceph Storage 节点磁盘布局。

5. 使用包含根据您的环境要求调整的以下参数的文件实现系统性能优化：

```
cat > /home/stack/dcn0/tuning.yaml << EOF
parameter_defaults:
  CephAnsibleExtraConfig:
    is_hci: true
  CephConfigOverrides:
    osd_recovery_op_priority: 3
    osd_recovery_max_active: 3
    osd_max_backfills: 1
 ## Set relative to your hardware:
  # DistributedComputeHCIParameters:
  #   NovaReservedHostMemory: 181000
  # DistributedComputeHCIExtraConfig:
  #   nova::cpu_allocation_ratio: 8.2
EOF
```

- 有关为 **CephAnsibleExtraConfig** 设置值的更多信息，请参阅设置 ceph-ansible 组变量。
- 有关为 **CephConfigOverrides** 设置值的更多信息，请参阅自定义 Ceph Storage 集群。

6. 在 **site-name.yaml** 环境文件中为站点配置命名约定。Nova 可用区和 Cinder 存储可用区必须匹配。部署带有存储的边缘站点时，包括 **CinderVolumeCluster** 参数。当 cinder-volume 部署为 active/active （在边缘站点需要）时，使用此参数。作为最佳实践，将 Cinder 集群名称设置为与可用区匹配：

```
cat > /home/stack/central/site-name.yaml << EOF
parameter_defaults:
    ...
    NovaComputeAvailabilityZone: dcn0
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: dcn0
    CinderVolumeCluster: dcn0
```

7. 生成用于 dcn0 部署的 **roles.yaml** 文件，例如：

```
openstack overcloud roles generate DistributedComputeHCI
DistributedComputeHCIScaleOut -o ~/dcn0/roles_data.yaml
```

8. 通过为每个角色创建 **~/dcn0/roles-counts.yaml** 文件来设置每个角色中的数量系统。
+ 您必须分配三个节点来满足 GlanceApiEdge 服务的要求。将 DistributedComputeHCICount 参数用于超融合基础架构。对于其他架构，请使用 DistributedComputeCount 参数。

```
parameter_defaults:
  ControllerCount: 0
  ComputeCount: 0
  DistributedComputeHCICount: 3
  DistributedComputeHCIScaleOutCount: 1  # Optional
  DistributedComputeScaleOutCount: 1     # Optional
```

9. 检索边缘站点的容器镜像：

```
sudo openstack tripleo container image prepare \
--environment-directory dcn0 \
-r ~/dcn0/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
\
...
-e /home/stack/dcn-common/central-export.yaml \
-e /home/stack/containers-prepare-parameter.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

注意

您必须在 **openstack tripleo container image prepare** 命令中包含用于部署的所有环境文件。

10. 部署边缘站点：

```
openstack overcloud deploy \
    --stack dcn0 \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -r ~/dcn0/roles_data.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
    -e ~/dnc0/dcn0-images-env.yaml \
    ....
```

```
-e ~/dcn-common/central-export.yaml \
-e ~/dcn-common/central_ceph_external.yaml \
-e ~/dcn0/dcn_ceph_keys.yaml \
-e ~/dcn0/role-counts.yaml \
-e ~/dcn0/ceph.yaml \
-e ~/dcn0/site-name.yaml \
-e ~/dcn0/tuning.yaml \
-e ~/dcn0/glance.yaml
```

注意

您必须在 **openstack overcloud deploy** 命令中包含用于配置网络配置的 heat 模板。为边缘架构设计需要 spine 和 leaf 网络。如需了解更多详细信息，请参阅 Spine Leaf Networking。

11. 在部署了边缘位置后，您必须确保在 nova API 数据库中创建 nova **cell_v2** 主机映射。在 undercloud 上运行以下命令：

```
TRIPLEO_PLAN_NAME=central \
ansible -i /usr/bin/tripleo-ansible-inventory \
nova_api[0] -b -a \
"{{ container_cli }} exec -it nova_api \
nova-manage cell_v2 discover_hosts --by-service --verbose"
```

如果扩展边缘站点，您必须再次运行该命令。

7.2. 使用专用 CEPH 节点部署边缘站点

您可以使用 Red Hat OpenStack Platform director 部署专用 Ceph 节点。

流程

1. **从中央** 堆栈导出堆栈信息。在运行此命令前，您必须部署 **中央** 堆栈：

```
openstack overcloud export \
    --config-download-dir /var/lib/mistral/central/ \
    --stack central \
    --output-file ~/dcn-common/central-export.yaml
```

注意

config-download-dir 值默认为 **/var/lib/mistral/<stack>/**。

2. 创建 **central_ceph_external.yaml** 文件。此环境文件将 DCN 站点连接到中央 hub Ceph 集群，因此信息特定于前面步骤中部署的 Ceph 集群。

```
sudo -E openstack overcloud export ceph \
--stack central \
--config-download-dir /var/lib/mistral \
--output-file ~/dcn-common/central_ceph_external.yaml
```

3. 为 glance 配置覆盖创建 **~/dcn0/glance.yaml** 文件：

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'dcn0 rbd glance store'
  GlanceBackendID: dcn0
  GlanceMultistoreConfig:
    central:
      GlanceBackend: rbd
      GlanceStoreDescription: 'central rbd glance store'
      CephClientUserName: 'openstack'
      CephClusterName: central
```

4. 使用与可用硬件相关的配置参数配置 **ceph.yaml** 文件。

```
cat > /home/stack/dcn0/ceph.yaml << EOF
parameter_defaults:
  CephClusterName: dcn0
  CephAnsibleDisksConfig:
    osd_scenario: lvm
    osd_objectstore: bluestore
    devices:
      - /dev/sda
      - /dev/sdb
  CephPoolDefaultSize: 3
  CephPoolDefaultPgNum: 128
EOF
```

有关更多信息，请参阅 *映射 Ceph Storage 节点磁盘布局*。

5. 在 **site-name.yaml** 环境文件中为站点配置命名约定。Nova 可用区和 Cinder 存储可用区必须匹配。部署带有存储的边缘站点时，包括 **CinderVolumeCluster** 参数。当 cinder-volume 部署为 active/active （在边缘站点需要）时，使用此参数。作为最佳实践，将 Cinder 集群名称设置为与可用区匹配：

```
cat > /home/stack/dcn0/site-name.yaml << EOF
parameter_defaults:
    ...
    NovaComputeAvailabilityZone: dcn0
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: dcn0
    CinderVolumeCluster: dcn0
```

6. 生成用于 dcn0 部署的 **roles.yaml** 文件，例如：

```
openstack overcloud roles generate DistributedCompute DistributedComputeScaleOut
CephAll-o ~/dcn0/roles_data.yaml
```

7. 通过为每个角色创建 **~/dcn0/roles-counts.yaml** 文件来设置每个角色中的数量系统。
您必须为 DistributedCompute 角色分配三个节点，以满足 GlanceApiEdge 服务的要求，并为 **CephAll** 角色分配三个节点。

```
parameter_defaults:
  ControllerCount: 0
  ComputeCount: 0
```

```
DistributedComputeCount: 3
CephAll: 3
DistributedComputeScaleOutCount: 1  # Optional
```

8. *检索边缘站点的容器镜像：*

```
sudo openstack tripleo container image prepare \
--environment-directory dcn0 \
-r ~/dcn0/roles_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml
\
...
-e /home/stack/dcn-common/central-export.yaml \
-e /home/stack/containers-prepare-parameter.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

注意

您必须在 ***openstack tripleo container image prepare*** *命令中包含用于部署的所有环境文件。*

9. *部署边缘站点：*

```
openstack overcloud deploy \
   --stack dcn0 \
   --templates /usr/share/openstack-tripleo-heat-templates/ \
   -r ~/dcn0/roles_data.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
   -e ~/dnc0/dcn0-images-env.yaml \
   ....
   -e ~/dcn-common/central-export.yaml \
   -e ~/dcn-common/central_ceph_external.yaml \
   -e ~/dcn0/dcn_ceph_keys.yaml \
   -e ~/dcn0/role-counts.yaml \
   -e ~/dcn0/ceph.yaml \
   -e ~/dcn0/site-name.yaml \
   -e ~/dcn0/tuning.yaml \
   -e ~/dcn0/glance.yaml
```

注意

您必须在 ***openstack overcloud deploy*** *命令中包含用于配置网络配置的 heat 模板。为边缘架构设计需要 spine 和 leaf 网络。如需了解更多详细信息，请参阅 Spine Leaf Networking。*

10. *在部署了边缘位置后，您必须确保在 nova API 数据库中创建 nova* ***cell_v2*** *主机映射。在 undercloud 上运行以下命令：*

```
TRIPLEO_PLAN_NAME=central \
ansible -i /usr/bin/tripleo-ansible-inventory \
```

```
nova_api[0] -b -a \
"{{ container_cli }} exec -it nova_api \
nova-manage cell_v2 discover_hosts --by-service --verbose"
```

如果扩展边缘站点，您必须再次运行该命令。

7.3. 在边缘使用预安装的 RED HAT CEPH STORAGE 集群

您可以将 Red Hat OpenStack Platform 配置为使用已存在的 Ceph 集群。这称为外部 Ceph 部署。

先决条件

- 您必须有一个预安装的 Ceph 集群，它与 DCN 站点本地，以便不会超过延迟要求。

流程

1. 在 Ceph 集群中创建以下池。如果要在中央位置部署，请包含 **备份** 和 **指标** 池：

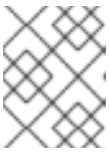
```
[root@ceph ~]# ceph osd pool create volumes <_PGnum_>
[root@ceph ~]# ceph osd pool create images <_PGnum_>
[root@ceph ~]# ceph osd pool create vms <_PGnum_>
[root@ceph ~]# ceph osd pool create backups <_PGnum_>
[root@ceph ~]# ceph osd pool create metrics <_PGnum_>
```

将 <_PGnum_> 替换为放置组数量。您可以使用 每个池的 Ceph Placement Groups (PG) 计算器 来确定合适的值。

2. 在 Ceph 中创建 OpenStack 客户端用户，以提供 Red Hat OpenStack Platform 环境对适当池的访问权限：

```
ceph auth add client.openstack mon 'allow r' osd 'allow class-read object_prefix rbd_children,
allow rwx pool=volumes, allow rwx pool=vms, allow rwx pool=images'
```

保存返回的 Ceph 客户端密钥。在配置 undercloud 时，使用此键作为 **CephClientKey** 参数的值。



注意

如果您在中央位置运行此命令，并计划使用 Cinder 备份或遥测服务，请在命令中添加 allow rwx pool=backups，在命令中允许 pool=metrics。

3. 保存 Ceph Storage 集群的文件系统 ID。Ceph 配置文件的 **[global]** 部分中的 **fsid** 参数的值是文件系统 ID：

```
[global]
fsid = 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
...
```

在配置 undercloud 时，使用这个值作为 **CephClusterFSID** 参数的值。

4. 在 undercloud 上，创建一个环境文件来配置节点以连接到非受管 Ceph 集群。使用可识别的命名惯例，如 ceph-external-<SITE>.yaml，其中 SITE 是部署的位置，如 ceph-external-central.yaml、ceph-external-dcn1.yaml 等。

```
parameter_defaults:
  # The cluster FSID
  CephClusterFSID: '4b5c8c0a-ff60-454b-a1b4-9747aa737d19'
  # The CephX user auth key
  CephClientKey: 'AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ=='
  # The list of IPs or hostnames of the Ceph monitors
  CephExternalMonHost: '172.16.1.7, 172.16.1.8, 172.16.1.9'
  # The desired name of the generated key and conf files
  CephClusterName: dcn1
```

a. 将之前保存的值用于 CephClusterFSID 和 CephClientKey 参数。

b. 使用 Ceph 监视器中以逗号分隔的 ip 地址列表，作为 CephExternalMonHost 参数的值。

c. 您必须在边缘站点间为 **CephClusterName** 参数选择一个唯一值。重新使用名称将导致配置文件被覆盖。

5. 如果您使用 Red Hat OpenStack Platform director 部署 Red Hat Ceph Storage，您可以将 ceph 配置导出到环境文件 **central_ceph_external.yaml**。此环境文件将 DCN 站点连接到中央 hub Ceph 集群，因此信息特定于上一步中部署的 Ceph 集群：

```
sudo -E openstack overcloud export ceph \
--stack central \
--config-download-dir /var/lib/mistral \
--output-file ~/dcn-common/central_ceph_external.yaml
```

如果中央位置从外部部署 Red Hat Ceph Storage，则无法使用 **openstack overcloud export ceph** 命令生成 **central_ceph_external.yaml** 文件。您必须手动创建 central_ceph_external.yaml 文件：

```
parameter_defaults:
  CephExternalMultiConfig:
    - cluster: "central"
      fsid: "3161a3b4-e5ff-42a0-9f53-860403b29a33"
      external_cluster_mon_ips: "172.16.11.84, 172.16.11.87, 172.16.11.92"
      keys:
        - name: "client.openstack"
          caps:
            mgr: "allow *"
            mon: "profile rbd"
            osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
          key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
          mode: "0600"
      dashboard_enabled: false
      ceph_conf_overrides:
        client:
          keyring: /etc/ceph/central.client.openstack.keyring
```

6. 创建一个环境文件，其中包含与每个站点的类似详细信息，其具有非受管 Red Hat Ceph Storage 集群用于中央位置。**openstack overcloud export ceph** 命令不适用于带有非受管 Red Hat Ceph Storage 集群的站点。当您更新中央位置时，此文件将允许存储集群位于边缘站点的中央位置作为二级位置

```
parameter_defaults:
  CephExternalMultiConfig:
```

```
cluster: dcn1
…
cluster: dcn2
…
```

7. 在部署 overcloud 时，使用 ceph-ansible-external.yaml、ceph-external-<SITE>.yaml 和 central_ceph_external.yaml 环境文件：

```
openstack overcloud deploy \
  --stack dcn1 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn1/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e ~/dnc1/ceph-external-dcn1.yaml  \
  ....
  -e ~/dcn-common/central-export.yaml \
  -e ~/dcn-common/central_ceph_external.yaml \
  -e ~/dcn1/dcn_ceph_keys.yaml \
  -e ~/dcn1/role-counts.yaml \
  -e ~/dcn1/ceph.yaml \
  -e ~/dcn1/site-name.yaml \
  -e ~/dcn1/tuning.yaml \
  -e ~/dcn1/glance.yaml
```

8. 部署所有边缘位置后重新部署中央位置。

7.4. 创建额外的分布式计算节点站点

新的分布式计算节点(DCN)站点在 undercloud 上具有自己的 YAML 文件的目录。更多信息请参阅 第 4.7 节 "管理独立的 heat 堆栈"。这个过程包含示例命令。

流程

1. 在 undercloud 上以 stack 用户身份，为 **dcn9** 创建一个新目录：

```
$ cd ~
$ mkdir dcn9
```

2. 将现有的 **dcn0** 模板复制到新目录中，并将 **dcn0** 字符串替换为 **dcn9** ：

```
$ cp dcn0/ceph.yaml dcn9/ceph.yaml
$ sed s/dcn0/dcn9/g -i dcn9/ceph.yaml
$ cp dcn0/overrides.yaml dcn9/overrides.yaml
$ sed s/dcn0/dcn9/g -i dcn9/overrides.yaml
$ sed s/"0-ceph-%index%"/"9-ceph-%index%"/g -i dcn9/overrides.yaml
$ cp dcn0/deploy.sh dcn9/deploy.sh
$ sed s/dcn0/dcn9/g -i dcn9/deploy.sh
```

3. 检查 **dcn9** 目录中的文件，以确认您的要求。

4. 编辑 undercloud.conf 以添加新的叶。在以下示例中，leaf9 添加到 undercloud.conf 中：

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

...
[leaf9]
cidr = 192.168.19.0/24
dhcp_start = 192.168.19.10
dhcp_end = 192.168.19.90
inspection_iprange = 192.168.19.100,192.168.19.190
gateway = 192.168.10.1
masquerade = False
```

5. 重新运行 openstack undercloud install 命令以更新环境配置。

6. 在 overcloud 模板中，将 **NetworkDeploymentActions** 参数的值从 **["CREATE"]** 的值更新为 **["CREATE", "UPDATE** "]。如果模板中目前没有包括此参数，请将其添加到其中一个环境文件中，或创建新环境文件。

```
cat > /home/stack/central/network-environment.yaml << EOF
parameter_defaults:
  NetworkDeploymentActions: ["CREATE", "UPDATE"]
EOF
```

7. 为中央位置运行部署脚本。包括您首次部署中央位置时使用的所有模板，以及新创建的 network-environment.yaml 文件：

```
openstack overcloud deploy \
    --stack central \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -r ~/central/roles_data.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
    -e ~/central/dcn9-images-env.yaml \
    ....
    -e ~/dcn-common/central-export.yaml \
    -e ~/dcn-common/central_ceph_external.yaml \
    -e ~/central/dcn_ceph_keys.yaml \
    -e ~/central/role-counts.yaml \
    -e ~/central/ceph.yaml \
    -e ~/central/site-name.yaml \
    -e ~/central/tuning.yaml \
    -e ~/central/glance.yaml
```

8. 验证您的节点是否可用，且处于 **Provisioning 状态** ：

```
$ openstack baremetal node list
```

9. 当节点可用时，使用所有适当的模板部署新的边缘站点：

```
openstack overcloud deploy \
  --stack dcn9 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn9/roles_data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/dcn-hci.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
  -e ~/dnc9/dcn9-images-env.yaml \
  ....
  -e ~/dcn-common/central-export.yaml \
  -e ~/dcn-common/central_ceph_external.yaml \
  -e ~/dcn9/dcn_ceph_keys.yaml \
  -e ~/dcn9/role-counts.yaml \
  -e ~/dcn9/ceph.yaml \
  -e ~/dcn9/site-name.yaml \
  -e ~/dcn9/tuning.yaml \
  -e ~/dcn9/glance.yaml
```

10. 如果您使用直接边缘到边缘通信部署位置，您必须重新部署每个边缘站点以更新路由并与新位置建立通信。

7.5. 更新中央位置

使用示例过程配置和部署所有边缘站点后，更新中央位置的配置，以便中央镜像服务可将镜像推送到边缘站点。

警告

此流程重启镜像服务(glance)，并中断任何长时间运行的镜像服务进程。例如，如果镜像从中央镜像服务服务器复制到 DCN 镜像服务服务器，该镜像副本将中断，您必须重启该镜像。如需更多信息，请参阅 在 镜像服务进程中断后清除遗留的数据。

流程

1. 创建一个类似如下的 **~/central/glance_update.yaml** 文件：这个示例包括两个边缘站点 dcn0 和 dcn1 的配置：

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
    dcn0:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn0 rbd glance store'
```

```
    CephClientUserName: 'openstack'
    CephClusterName: dcn0
    GlanceBackendID: dcn0
  dcn1:
    GlanceBackend: rbd
    GlanceStoreDescription: 'dcn1 rbd glance store'
    CephClientUserName: 'openstack'
    CephClusterName: dcn1
    GlanceBackendID: dcn1
```

2. 创建 **dcn_ceph.yaml** 文件。在以下示例中，此文件将中央站点上的 glance 服务配置为边缘站点 **dcn0** 和 **dcn1** 的 Ceph 集群的客户端。

```
sudo -E openstack overcloud export ceph \
--stack dcn0,dcn1 \
--config-download-dir /var/lib/mistral \
--output-file ~/central/dcn_ceph.yaml
```

3. 使用原始模板重新部署中央站点，并包含新创建的 **dcn_ceph.yaml** 和 **glance_update.yaml** 文件。

```
openstack overcloud deploy \
    --stack central \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    -r ~/central/central_roles.yaml \
  ...
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
    -e ~/central/central-images-env.yaml \
    -e ~/central/role-counts.yaml \
    -e ~/central/site-name.yaml
    -e ~/central/ceph.yaml \
    -e ~/central/ceph_keys.yaml \
    -e ~/central/glance.yaml \
    -e ~/central/dcn_ceph_external.yaml
```

4. 在中央位置的控制器上，重新启动 **cinder-volume** 服务。如果您使用 **cinder-backup** 服务部署中央位置，则重启 **cinder-backup** 服务：

```
ssh heat-admin@controller-0 sudo pcs resource restart openstack-cinder-volume
ssh heat-admin@controller-0 sudo pcs resource restart openstack-cinder-backup
```

7.5.1. 在镜像服务进程中断后清除数据

当您重启中央位置时，任何长时间运行的镜像服务(glance)进程都会中断。在重启这些进程前，您必须首先清理您重启的 Controller 节点上的数据，并在 Ceph 和镜像服务数据库中。

流程

1. 检查并清除重启的 Controller 节点中的遗留数据。将暂存存储的 **glance-api.conf** 文件中的文件与镜像服务数据库中的对应镜像进行比较，如 < **image_ID>.raw**。
    - 如果这些对应的镜像显示导入状态，您必须重新创建镜像。

- 如果镜像显示 active 状态，您必须从暂存中删除数据并重启副本导入。

2. 检查和清除 Ceph 存储中的遗留数据。从暂存区域清理的镜像必须具有包含镜像的 Ceph 存储的 **stores** 属性中的匹配记录。Ceph 中的镜像名称是镜像服务数据库中的镜像 ID。

3. 清除镜像服务数据库。清除从导入作业导入状态的任何镜像都会中断：

```
$ glance image-delete <image_id>
```

7.6. 在 DCN 上部署 RED HAT CEPH STORAGE DASHBOARD

流程

要将 Red Hat Ceph Storage 仪表板部署到中央位置，请参阅将 Red Hat Ceph Storage 仪表板添加到 overcloud 部署中。这些步骤应该在部署中央位置前完成。

要将 Red Hat Ceph Storage Dashboard 部署到边缘位置，请完成您为中央相同的步骤，但您必须完成以下操作：

- 在模板中，为部署边缘站点，确保 **ManageNetworks** 参数的值为 **false**。将 **ManageNetworks** 设置为 **false** 时，Edge 站点将使用已在中央堆栈中创建的现有网络：

```
parameter_defaults:
  ManageNetworks: false
```

- 您必须部署自己的解决方案才能进行负载平衡，才能创建高可用性虚拟 IP。边缘站点不部署 haproxy，也不部署 pacemaker。当您将 Red Hat Ceph Storage Dashboard 部署到边缘位置时，部署会在存储网络上公开。控制面板安装在三个具有不同 IP 地址的 DistributedComputeHCI 节点上，而无需负载均衡解决方案。

您可以创建额外网络来托管可以公开 Ceph 控制面板的虚拟 IP。您不能为多个堆栈重复使用网络资源。有关重复使用网络资源的更多信息，请参阅在多个堆栈中重复使用网络资源。

要创建此额外网络资源，请使用提供的 **network_data_dashboard.yaml** heat 模板。创建的网络的名称是 **StorageDashboard**。

流程

1. 以 **堆栈** 身份登录到 Red Hat OpenStack Platform Director。

2. 生成 **DistributedComputeHCIDashboard** 角色以及适用于您的环境的任何其他角色：

```
openstack overcloud roles generate DistributedComputeHCIDashboard -o ~/dnc0/roles.yaml
```

3. 在 overcloud deploy 命令中包含 **roles.yaml** 和 **network_data_dashboard.yaml** ：

```
$ openstack overcloud deploy --templates \
-r ~/<dcn>/<dcn_site_roles>.yaml \
-n /usr/share/openstack-tripleo-heat-templates/network_data_dashboard.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
```

-

注意

部署提供了在存储网络上启用仪表板的三个 ip 地址。

验证

若要确认控制面板在中央位置运行，并且它从 Ceph 集群显示的数据正确，请参阅 访问 Ceph 仪表板。

您可以通过类似的步骤确认控制面板正在边缘位置运行，但存在例外情况，因为边缘位置没有负载均衡器。

1. 从 **/var/lib/mistral/<stackname>/ceph-ansible/group_vars/all.yml**检索特定于所选堆栈的仪表板 admin 登录凭证

2. 在特定于所选堆栈的清单中，**/var/lib/mistral/<stackname>/ceph-ansible/inventory.yml**，找到 DistributedComputeHCI 角色主机列表并保存所有三个 **storage_ip** 值。在以下示例中，前两个仪表板 IP 是 172.16.11.84 和 172.16.11.87 ：

```
DistributedComputeHCI:
  hosts:
    dcn1-distributed-compute-hci-0:
      ansible_host: 192.168.24.16
...
storage_hostname: dcn1-distributed-compute-hci-0.storage.localdomain
storage_ip: 172.16.11.84
...
    dcn1-distributed-compute-hci-1:
ansible_host: 192.168.24.22
...
storage_hostname: dcn1-distributed-compute-hci-1.storage.localdomain
storage_ip: 172.16.11.87
```

3. 如果 Ceph 控制面板可以访问这些 IP 地址之一，您可以检查 Ceph 控制面板是否活跃。这些 IP 地址位于存储网络中，且不会被路由。如果这些 IP 地址不可用，您必须为从清单中获取的三个 IP 地址配置负载均衡器，以获取虚拟 IP 地址进行验证。

第 8 章 替换 DISTRIBUTEDCOMPUTEHCI 节点

在硬件维护期间，您可能需要在边缘站点缩减、扩展或替换 DistributedComputeHCI 节点。要替换 DistributedComputeHCI 节点，请从要替换的节点中删除服务，扩展节点数量，然后按照流程重新扩展这些节点。

8.1. 删除计算(NOVA)服务

禁用 nova-compute 服务并删除相关的网络代理。

流程

1. 从堆栈中删除的节点：

```
`openstack overcloud node delete --stack <dcn2> \
<computehci2-1>
```

2. 删除您要删除的节点上的网络代理：

```
(central) [stack@site-undercloud-0 ~]$ openstack network agent list | grep dcn2
…
| 17726d1a-e9d1-4e57-b40d-e742be5d073c | Open vSwitch agent | dcn2-computehci2-
1.redhat.local          | None              | XXX   | UP    | neutron-openvswitch-agent |
…
(central) [stack@site-undercloud-0 ~]$ openstack network agent delete 17726d1a-e9d1-
4e57-b40d-e742be5d073c
```

8.2. 删除 RED HAT CEPH STORAGE 服务

要删除 Red Hat Ceph 服务 **mon**、**mgr** 和 **osd**，您必须从您要删除的节点上的集群服务中禁用和移除 **ceph-osd**，然后停止并禁用 **mon**、**mgr** 和 **osd** 服务。

流程

1. 使用 SSH 连接到您要删除的 DistributedComputeHCI 节点，并以 root 用户身份登录。

```
$ ssh heat-admin@<dcn-computehci-node>
$ sudo su -
#
```

2. 识别与您要删除的 DistributedComputeHCI 节点关联的 OSD：

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dnc2-computehci2-1 ceph osd tree -
c /etc/ceph/dcn2.conf
…
-3       0.24399     host dcn2-computehci2-1
 1   hdd 0.04880         osd.1                    up  1.00000 1.00000
 7   hdd 0.04880         osd.7                    up  1.00000 1.00000
11   hdd 0.04880         osd.11                   up  1.00000 1.00000
15   hdd 0.04880         osd.15                   up  1.00000 1.00000
18   hdd 0.04880         osd.18                   up  1.00000 1.00000
…
```

3. *禁用相关 Ceph 节点上的 OSD：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd out 1
7 11 15 18 -c /etc/ceph/dcn2.conf
marked out osd.1. marked out osd.7. marked out osd.11. marked out osd.15. marked out
osd.18.
```

4. *等待 Ceph osd 重新平衡完成。使用以下命令监控进度：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph -w -c
/etc/ceph/dcn2.conf
…
mon.dcn2-computehci2-2 has auth_allow_insecure_global_id_reclaim set to true
```

当您看到 ***auth_allow_insecure_global_id_reclaim*** *设置为* ***true*** *时，重新平衡已完成。*

5. *停止并禁用 OSD：*

```
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-osd@1
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-osd@7
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-osd@11
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-osd@15
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-osd@18
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-osd@1
Removed /etc/systemd/system/multi-user.target.wants/ceph-osd@1.service.
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-osd@7
Removed /etc/systemd/system/multi-user.target.wants/ceph-osd@7.service.
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-osd@11
Removed /etc/systemd/system/multi-user.target.wants/ceph-osd@11.service.
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-osd@15
Removed /etc/systemd/system/multi-user.target.wants/ceph-osd@15.service.
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-osd@18
Removed /etc/systemd/system/multi-user.target.wants/ceph-osd@18.service.
```

6. *从 CRUSH map 中删除 OSD：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush remove osd.1 -c /etc/ceph/dcn2.conf
removed item id 1 name 'osd.1' from crush map
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush remove osd.7 -c /etc/ceph/dcn2.conf
removed item id 7 name 'osd.7' from crush map
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush remove osd.11 -c /etc/ceph/dcn2.conf
removed item id 11 name 'osd.11' from crush map
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush remove osd.15 -c /etc/ceph/dcn2.conf
removed item id 15 name 'osd.15' from crush map
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush remove osd.18 -c /etc/ceph/dcn2.conf
removed item id 18 name 'osd.18' from crush map
```

7. *删除 OSD 身份验证密钥：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph auth del
osd.1 -c /etc/ceph/dcn2.conf
updated
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph auth del
osd.7 -c /etc/ceph/dcn2.conf
updated
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph auth del
osd.11 -c /etc/ceph/dcn2.conf
updated
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph auth del
osd.15 -c /etc/ceph/dcn2.conf
updated
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph auth del
osd.18 -c /etc/ceph/dcn2.conf
updated
```

8. *从集群中移除 OSD：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd rm 1
-c /etc/ceph/dcn2.conf
removed osd.1
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd rm 7
-c /etc/ceph/dcn2.conf
removed osd.7
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd rm
11 -c /etc/ceph/dcn2.conf
removed osd.11
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd rm
15 -c /etc/ceph/dcn2.conf
removed osd.15
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd rm
18 -c /etc/ceph/dcn2.conf
removed osd.18
```

9. *从 CRUSH map 中删除 DistributedComputeHCI 节点：*

```
[root@dcn2-computehci2-1 ~]# podman exec ceph-mon-dcn2-computehci2-1 ceph osd
crush rm dcn2-computehci2-1 -c /etc/ceph/dcn2.conf
removed item id -3 name 'dcn2-computehci2-1' from crush map
```

10. *停止并禁用 **mon** 服务：*

```
[root@dcn2-computehci2-1 ~]# systemctl --type=service | grep ceph
  ceph-crash@dcn2-computehci2-1.service     loaded active     running       Ceph crash dump
collector
  ceph-mgr@dcn2-computehci2-1.service       loaded active     running       Ceph Manager
  ceph-mon@dcn2-computehci2-1.service       loaded active     running       Ceph Monitor

[root@dcn2-computehci2-1 ~]# systemctl stop ceph-mon@dcn2-computehci2-1

[root@dcn2-computehci2-1 ~]# systemctl disable ceph-mon@dcn2-computehci2-1
Removed /etc/systemd/system/multi-user.target.wants/ceph-mon@dcn2-computehci2-
1.service.
```

11. *使用 SSH 连接到同一集群中的另一节点，并从集群中移除该监控器。请注意输出中的 v1 和 v2 条目：*

```
[root@dcn2-computehci2-0 ~]# podman exec ceph-mon-dcn2-computehci2-0 ceph mon
remove dcn2-computehci2-1 -c /etc/ceph/dcn2.conf
removing mon.dcn2-computehci2-1 at [v2:172.23.3.153:3300/0,v1:172.23.3.153:6789/0],
there will be 2 monitors
```

12. *在所有 dcn2 节点上，删除上一步中输出的 /etc/ceph/dcn2.conf 中的 v1 和 v2 监控条目，并从 'mon 初始成员' 中删除节点名称：*

之前

mon host = [v2:172.23.3.150:3300,v1:172.23.3.150:6789],* [v2:172.23.3.153:3300,v1:172.23.3.153:6789]*,[v2:172.23.3.124:3300,v1:172.23.3.124:6789] + mon initial members = dcn2-computehci2-0,*dcn2-computehci2-1*,dcn2-computehci2-2

After

mon host = [v2:172.23.3.150:3300,v1:172.23.3.150:6789], [v2:172.23.3.124:3300,v1:172.23.3.124:6789] + mon initial members = dcn2-computehci2-0,dcn2-computehci2-2

13. *停止并禁用 **mgr** 服务：*

```
[root@dcn2-computehci2-1 ~]# systemctl --type=service | grep ceph
  ceph-crash@dcn2-computehci2-1.service    loaded active     running       Ceph crash dump
collector
  ceph-mgr@dcn2-computehci2-1.service      loaded active     running       Ceph Manager
[root@dcn2-computehci2-1 ~]# systemctl stop ceph-mgr@dcn2-computehci2-1
[root@dcn2-computehci2-1 ~]# systemctl --type=service | grep ceph
  ceph-crash@dcn2-computehci2-1.service  loaded active running Ceph crash dump
collector
[root@dcn2-computehci2-1 ~]# systemctl disable ceph-mgr@dcn2-computehci2-1
Removed /etc/systemd/system/multi-user.target.wants/ceph-mgr@dcn2-computehci2-
1.service.
```

14. *验证节点的 **mgr** 服务是否已从集群中移除。*

```
[root@dcn2-computehci2-0 ~]# podman exec ceph-mon-dcn2-computehci2-0 ceph -s -c
/etc/ceph/dcn2.conf
cluster:
    id:     b9b53581-d590-41ac-8463-2f50aa985001
    health: HEALTH_WARN
            3 pools have too many placement groups
            mons are allowing insecure global_id reclaim

  services:
    mon: 2 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0 (age 2h)
    mgr: dcn2-computehci2-2(active, since 20h), standbys: dcn2-computehci2-0 (1)
    osd: 15 osds: 15 up (since 3h), 15 in (since 3h)

  data:
    pools:   3 pools, 384 pgs
```

```
    objects: 32 objects, 88 MiB
    usage:   16 GiB used, 734 GiB / 750 GiB avail
    pgs:     384 active+clean
```

1 当 **mgr** 服务被成功移除时，将不再列出 **mgr** 服务的节点。

8.3. 删除镜像服务(GLANCE)服务

从服务中删除镜像时，从节点中删除镜像服务。

流程

- 要禁用镜像服务，请在您要删除的节点上使用 **systemctl** 来禁用它们：

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_glance_api.service
[root@dcn2-computehci2-1 ~]# systemctl stop  tripleo_glance_api_tls_proxy.service

[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_glance_api.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api.service.
[root@dcn2-computehci2-1 ~]# systemctl disable  tripleo_glance_api_tls_proxy.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_glance_api_tls_proxy.service.
```

8.4. 删除 BLOCK STORAGE (CINDER)服务

从服务中删除时，您必须从 DistributedComputeHCI 节点中删除 **cinder-volume** 和 **etcd** 服务。

流程

1. 在您要删除的节点上识别并禁用 **cinder-volume** 服务：

```
(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
| cinder-volume | dcn2-computehci2-1@tripleo_ceph | az-dcn2    | enabled | up    | 2022-03-
23T17:41:43.000000 |
(central) [stack@site-undercloud-0 ~]$ openstack volume service set --disable dcn2-
computehci2-1@tripleo_ceph cinder-volume
```

2. 登录到堆栈中的不同 DistributedComputeHCI 节点：

```
$ ssh heat-admin@dcn2-computehci2-0
```

3. 删除与您要删除的节点关联的 **cinder-volume** 服务：

```
[root@dcn2-computehci2-0 ~]# podman exec -it cinder_volume cinder-manage service
remove cinder-volume dcn2-computehci2-1@tripleo_ceph
Service cinder-volume on host dcn2-computehci2-1@tripleo_ceph removed.
```

4. 在您要删除的节点上停止并禁用 **tripleo_cinder_volume** 服务：

```
[root@dcn2-computehci2-1 ~]# systemctl stop tripleo_cinder_volume.service
[root@dcn2-computehci2-1 ~]# systemctl disable tripleo_cinder_volume.service
Removed /etc/systemd/system/multi-user.target.wants/tripleo_cinder_volume.service
```

8.5. 删除 DISTRIBUTEDCOMPUTEHCI 节点

要保留环境，**openstack overcloud node delete** 命令必须包含所有相关模板和环境文件：

流程

1. 删除 DistributedComputeHCI 节点

```
$ openstack overcloud node delete /
--stack <stack-name> /
<node_UUID>
```

2. 如果要重复使用该节点，请使用 ironic 清理磁盘。如果节点将托管 Ceph OSD，则需要此项：

```
openstack baremetal node manage $UUID
openstack baremetal node clean $UUID --clean-steps '[{"interface":"deploy", "step":
"erase_devices_metadata"}]'
openstack baremetal provide $UUID
```

8.6. 替换删除的 DISTRIBUTEDCOMPUTEHCI 节点

8.6.1. 替换删除的 DistributedComputeHCI 节点

要在 DCN 部署中添加新的 HCI 节点，您必须使用额外的节点重新部署边缘堆栈，执行该堆栈的 **ceph** 导出，然后对中央位置执行堆栈更新。中央位置的堆栈更新会添加特定于边缘站点的配置。

先决条件

您要替换节点的堆栈的 nodes_data.yaml 文件中的节点数是正确的。

流程

1. 在您的部署脚本调用的一个模板中将 **EtcdIntialClusterState** 参数设置为 **existing**：

```
parameter_defaults:
  EtcdInitialClusterState: existing
```

2. 使用特定于堆栈的部署脚本重新部署：

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy_dcn2.sh
…
Overcloud Deployed without error
```

3. 从堆栈导出 Red Hat Ceph Storage 数据：

```
(undercloud) [stack@site-undercloud-0 ~]$ sudo -E openstack overcloud export ceph --stack
dcn1,dcn2 --config-download-dir /var/lib/mistral --output-file
~/central/dcn2_scale_up_ceph_external.yaml
```

4. 将 dcn_ceph_external.yaml 替换为部署脚本中新生成的 dcn2_scale_up_ceph_external.yaml。

5. 在中央执行堆栈更新：

```
(undercloud) [stack@site-undercloud-0 ~]$ ./overcloud_deploy.sh
...
Overcloud Deployed without error
```

8.7. 验证替换的 DISTRIBUTEDCOMPUTEHCI 节点的功能

1. 确保 **status** 字段的值 **已启用**，并且 **State** 字段的值为 **up** ：

```
(central) [stack@site-undercloud-0 ~]$ openstack compute service list -c Binary -c Host -c
Zone -c Status -c State
+----------------+-----------------------------------------+------------+---------+-------+
| Binary         | Host                                    | Zone       | Status  | State |
+----------------+-----------------------------------------+------------+---------+-------+
...
| nova-compute   | dcn1-compute1-0.redhat.local            | az-dcn1    | enabled | up    |
| nova-compute   | dcn1-compute1-1.redhat.local            | az-dcn1    | enabled | up    |
| nova-compute   | dcn2-computehciscaleout2-0.redhat.local | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computehci2-0.redhat.local         | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computescaleout2-0.redhat.local    | az-dcn2    | enabled | up    |
| nova-compute   | dcn2-computehci2-2.redhat.local         | az-dcn2    | enabled | up    |
...
```

2. 确保所有网络代理都处于 **up** 状态：

```
(central) [stack@site-undercloud-0 ~]$ openstack network agent list -c "Agent Type" -c Host -
c Alive -c State
+--------------------+-----------------------------------------+-------+-------+
| Agent Type         | Host                                    | Alive | State |
+--------------------+-----------------------------------------+-------+-------+
| DHCP agent         | dcn3-compute3-1.redhat.local            | :-)   | UP    |
| Open vSwitch agent | central-computehci0-1.redhat.local      | :-)   | UP    |
| DHCP agent         | dcn3-compute3-0.redhat.local            | :-)   | UP    |
| DHCP agent         | central-controller0-2.redhat.local      | :-)   | UP    |
| Open vSwitch agent | dcn3-compute3-1.redhat.local            | :-)   | UP    |
| Open vSwitch agent | dcn1-compute1-1.redhat.local            | :-)   | UP    |
| Open vSwitch agent | central-computehci0-0.redhat.local      | :-)   | UP    |
| DHCP agent         | central-controller0-1.redhat.local      | :-)   | UP    |
| L3 agent           | central-controller0-2.redhat.local      | :-)   | UP    |
| Metadata agent     | central-controller0-1.redhat.local      | :-)   | UP    |
| Open vSwitch agent | dcn2-computescaleout2-0.redhat.local    | :-)   | UP    |
| Open vSwitch agent | dcn2-computehci2-5.redhat.local         | :-)   | UP    |
| Open vSwitch agent | central-computehci0-2.redhat.local      | :-)   | UP    |
| DHCP agent         | central-controller0-0.redhat.local      | :-)   | UP    |
| Open vSwitch agent | central-controller0-1.redhat.local      | :-)   | UP    |
| Open vSwitch agent | dcn2-computehci2-0.redhat.local         | :-)   | UP    |
| Open vSwitch agent | dcn1-compute1-0.redhat.local            | :-)   | UP    |
...
```

3. 验证 Ceph 集群的状态：

   a. 使用 SSH 连接到新的 DistributedComputeHCI 节点，并检查 Ceph 集群的状态：

```
[root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 \
ceph -s -c /etc/ceph/dcn2.conf
```

b. *验证新节点的 ceph mon 和 ceph mgr 服务都存在：*

```
services:
    mon: 3 daemons, quorum dcn2-computehci2-2,dcn2-computehci2-0,dcn2-
computehci2-5 (age 3d)
    mgr: dcn2-computehci2-2(active, since 3d), standbys: dcn2-computehci2-0, dcn2-
computehci2-5
    osd: 20 osds: 20 up (since 3d), 20 in (since 3d)
```

c. *使用 'ceph osd tree' 验证 ceph osds 的状态。确保我们的新节点的所有 osds 都位于 STATUS up 中：*

```
[root@dcn2-computehci2-5 ~]# podman exec -it ceph-mon-dcn2-computehci2-5 ceph
osd tree -c /etc/ceph/dcn2.conf
ID CLASS WEIGHT  TYPE NAME                     STATUS REWEIGHT PRI-AFF
-1       0.97595 root default
-5       0.24399     host dcn2-computehci2-0
 0   hdd 0.04880         osd.0                 up  1.00000 1.00000
 4   hdd 0.04880         osd.4                 up  1.00000 1.00000
 8   hdd 0.04880         osd.8                 up  1.00000 1.00000
13   hdd 0.04880         osd.13                up  1.00000 1.00000
17   hdd 0.04880         osd.17                up  1.00000 1.00000
-9       0.24399     host dcn2-computehci2-2
 3   hdd 0.04880         osd.3                 up  1.00000 1.00000
 5   hdd 0.04880         osd.5                 up  1.00000 1.00000
10   hdd 0.04880         osd.10                up  1.00000 1.00000
14   hdd 0.04880         osd.14                up  1.00000 1.00000
19   hdd 0.04880         osd.19                up  1.00000 1.00000
-3       0.24399     host dcn2-computehci2-5
 1   hdd 0.04880         osd.1                 up  1.00000 1.00000
 7   hdd 0.04880         osd.7                 up  1.00000 1.00000
11   hdd 0.04880         osd.11                up  1.00000 1.00000
15   hdd 0.04880         osd.15                up  1.00000 1.00000
18   hdd 0.04880         osd.18                up  1.00000 1.00000
-7       0.24399     host dcn2-computehciscaleout2-0
 2   hdd 0.04880         osd.2                 up  1.00000 1.00000
 6   hdd 0.04880         osd.6                 up  1.00000 1.00000
 9   hdd 0.04880         osd.9                 up  1.00000 1.00000
12   hdd 0.04880         osd.12                up  1.00000 1.00000
16   hdd 0.04880         osd.16                up  1.00000 1.00000
```

4. *验证新 DistributedComputeHCI 节点的* ***cinder-volume*** *服务是否处于 Status 'enabled' 和 in State 'up' 中：*

```
(central) [stack@site-undercloud-0 ~]$ openstack volume service list --service cinder-volume
-c Binary -c Host -c Zone -c Status -c State
+--------------+---------------------------------+------------+---------+-------+
| Binary       | Host                            | Zone       | Status  | State |
+--------------+---------------------------------+------------+---------+-------+
| cinder-volume | hostgroup@tripleo_ceph          | az-central | enabled | up    |
| cinder-volume | dcn1-compute1-1@tripleo_ceph    | az-dcn1    | enabled | up    |
| cinder-volume | dcn1-compute1-0@tripleo_ceph    | az-dcn1    | enabled | up    |
| cinder-volume | dcn2-computehci2-0@tripleo_ceph | az-dcn2    | enabled | up    |
```

```
| cinder-volume | dcn2-computehci2-2@tripleo_ceph | az-dcn2    | enabled | up    |
| cinder-volume | dcn2-computehci2-5@tripleo_ceph | az-dcn2    | enabled | up    |
+---------------+---------------------------------+------------+---------+-------+
```

注意

*如果 **cinder-volume** 服务的 State 为 **down**，则该服务尚未在节点上启动。*

5. *使用 ssh 连接到新的 DistributedComputeHCI 节点，并使用 'systemctl' 检查 Glance 服务的状态：*

```
[root@dcn2-computehci2-5 ~]# systemctl --type service | grep glance
  tripleo_glance_api.service                   loaded active     running       glance_api container
  tripleo_glance_api_healthcheck.service       loaded activating start   start glance_api
healthcheck
  tripleo_glance_api_tls_proxy.service         loaded active     running
glance_api_tls_proxy container
```

8.8. 对 DISTRIBUTEDCOMPUTEHCI 状态进行故障排除

*如果替换节点部署在 EtcdInitialClusterState 参数值的情况下，则替换节点的 **cinder-volume** 服务会在运行 **openstack volume service list** 时显示 **down**。*

流程

1. ***登录到替换节点并检查 etcd 服务的日志。检查日志是否在 /var/log/containers/stdouts/etcd.log 日志文件中显示 etcd 服务是否报告集群 ID 不匹配：***

```
2022-04-06T18:00:11.834104130+00:00 stderr F 2022-04-06 18:00:11.834045 E | rafthttp:
request cluster ID mismatch (got 654f4cf0e2cfb9fd want 918b459b36fe2c0c)
```

2. ***将 EtcdInitialClusterState 参数设置为部署模板中 现有 值，并重新运行部署脚本。***

3. ***使用 SSH 连接到替换节点，并以 root 用户身份运行以下命令：***

```
[root@dcn2-computehci2-4 ~]# systemctl stop tripleo_etcd
[root@dcn2-computehci2-4 ~]# rm -rf /var/lib/etcd/*
[root@dcn2-computehci2-4 ~]# systemctl start tripleo_etcd
```

4. ***重新检查 /var/log/containers/stdouts/etcd.log 日志文件，以验证节点是否成功加入集群：***

```
2022-04-06T18:24:22.130059875+00:00 stderr F 2022-04-06 18:24:22.129395 I |
etcdserver/membership: added member 96f61470cd1839e5 [https://dcn2-computehci2-
4.internalapi.redhat.local:2380] to cluster 654f4cf0e2cfb9fd
```

5. *检查 **cinder-volume** 服务的状态，确定在运行 **openstack volume service list** 时它在替换节点上为 **up**。*

第 9 章 使用密钥管理器进行部署

如果您在 Red Hat OpenStack Platform 16.1.2 发布前部署了边缘站点，则需要重新生成 roles.yaml 才能实现此功能：要实现这个功能：要实现该功能，请重新生成用于 DCN 站点部署的 roles.yaml 文件。

```
$ openstack overcloud roles generate DistributedComputeHCI DistributedComputeHCIScaleOut -o
~/dcn0/roles_data.yaml
```

9.1. 使用密钥管理器部署边缘站点

如果要在边缘站点包含对密钥管理器(barbican)服务的访问权限，您必须在中央位置配置 barbican。有关安装和配置 barbican 的详情，请参考 部署 Barbican。

- 您可以通过包括 /usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml 来配置从 DCN 站点对 barbican 的访问。

```
openstack overcloud deploy \
  --stack dcn0 \
  --templates /usr/share/openstack-tripleo-heat-templates/ \
  -r ~/dcn0/roles_data.yaml \
  ....
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/barbican-edge.yaml
```

第 10 章 将 GLANCE 镜像预缓存到 NOVA

当您将 OpenStack Compute 配置为使用本地临时存储时，glance 镜像会被缓存来加快实例部署。如果实例所需的镜像尚未缓存，则在创建实例时将其下载到 Compute 节点的本地磁盘。

下载 glance 镜像的过程需要变量的时间，具体取决于镜像大小和网络特征，如带宽和延迟。

如果您尝试启动实例，且镜像在本地的 Ceph 集群上不可用，则启动实例会失败并显示以下信息：

> Build of instance 3c04e982-c1d1-4364-b6bd-f876e399325b aborted: Image 20c5ff9d-5f54-4b74-830f-88e78b9999ed is unacceptable: No image locations are accessible

您可以在 Compute 服务日志中看到以下内容：

> 'Image %s is not on my ceph and [workarounds]/ never_download_image_if_on_rbd=True; refusing to fetch and upload.',

由于 nova.conf 配置文件中的一个参数（名为 never_download_image_if_on_rbd）对于 DCN 部署被默认设为 true，则实例会启动失败。您可以使用 heat 参数 NovaDisableImageDownloadToRbd 控制这个值，您可以在 dcn-hci.yaml 文件中找到。

如果在部署 overcloud 前将 NovaDisableImageDownloadToRbd 的值设置为 false，则会出现以下情况：

- 如果镜像在本地不可用，计算服务(nova)将自动流传输在 中央位置 可用的镜像。
- 您无法使用 glance 镜像中的 COW 副本。
- 计算(nova)存储可能会包含同一镜像的多个副本，具体取决于使用它的实例数量。
- 您可以饱和 WAN 链接到 中央位置 和 nova 存储池。

红帽建议将此值设置为 true，并确保在启动实例前在本地提供所需的镜像。有关使镜像提供给边缘的更多信息，请参阅 第 A.1.3 节 “将镜像复制到新站点”。

对于本地的镜像，您可以使用 tripleo_nova_image_cache.yml ansible playbook 来预缓存常用的镜像或镜像，从而加快虚拟机创建速度。

10.1. 运行 TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK

先决条件

- 在 shell 环境中对正确的 API 进行身份验证凭据。

在每个步骤中提供的命令前，您必须确保提供正确的身份验证文件。

流程

1. 为堆栈创建 ansible 清单文件。您可以在以逗号分隔的列表中指定多个堆栈，以在多个站点缓存镜像：

```
$ source stackrc

$ tripleo-ansible-inventory --plan central,dcn0,dcn1 \
--static-yaml-inventory inventory.yaml
```

2. 创建您要预缓存的镜像 ID 列表：

   a. 检索可用镜像的完整列表：

```
$ source centralrc

$ openstack image list
+--------------------------------------+---------+--------+
| ID                                   | Name    | Status |
+--------------------------------------+---------+--------+
| 07bc2424-753b-4f65-9da5-5a99d8383fe6 | image_0 | active |
| d5187afa-c821-4f22-aa4b-4e76382bef86 | image_1 | active |
+--------------------------------------+---------+--------+
```

   b. 创建名为 nova_cache_args.yml 的 ansible playbook 参数文件，并添加您要预缓存的镜像 ID：

```
---
tripleo_nova_image_cache_images:
  - id: 07bc2424-753b-4f65-9da5-5a99d8383fe6
  - id: d5187afa-c821-4f22-aa4b-4e76382bef86
```

3. 运行 **tripleo_nova_image_cache.yml ansible playbook:**

```
$ source centralrc

$ ansible-playbook -i inventory.yaml \
--extra-vars "@nova_cache_args.yml" \
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

10.2. 性能考虑

您可以指定您要与 **ansible fork** 参数同时下载的镜像数量，该参数默认为 **5**。您可以通过增加 **fork** 参数的值来缩短发布此镜像的时间，但您必须以增加网络和 **glance-api** 负载来平衡这个值。

使用 **--forks** 参数来调整并发性，如下所示：

```
ansible-playbook -i inventory.yaml \
--forks 10 \
--extra-vars "@nova_cache_args.yml" \
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

10.3. 优化镜像分发到 DCN 站点

您可以使用代理进行 **glance** 镜像分发来减少 **WAN** 流量。配置代理时：

- **Glance** 镜像下载到充当代理的单个 **Compute** 节点上。
- 代理将 **glance** 镜像重新分发到清单中的其他 **Compute** 节点。

您可以将以下参数放在 **nova_cache_args.yml ansible** 参数文件中，以配置代理节点。

将 **tripleo_nova_image_cache_use_proxy** 参数设置为 **true** 以启用镜像缓存代理。

镜像代理使用安全复制 scp 将镜像分发到清单中的其他节点。SCP 与具有高延迟的网络（如 DCN 站点间的 WAN）相比效率较低。红帽建议您将 playbook 目标限制为单个 DCN 位置，该位置与单个堆栈相关联。

使用 tripleo_nova_image_cache_proxy_hostname 参数来选择镜像缓存代理。默认代理是 ansible 清单文件中的第一个计算节点。使用 tripleo_nova_image_cache_plan 参数将 playbook 清单限制为单个站点：

```
tripleo_nova_image_cache_use_proxy: true
tripleo_nova_image_cache_proxy_hostname: dcn0-novacompute-1
tripleo_nova_image_cache_plan: dcn0
```

10.4. 配置 NOVA-CACHE 清理

当满足以下任一条件时，后台进程会定期运行来从 nova 缓存中删除镜像：

- 该镜像不供实例使用。
- 镜像的年龄大于 nova 参数 remove_unused_original_minimum_age_seconds 的值。

remove_unused_original_minimum_age_seconds 参数的默认值为 86400。该值以秒为单位表示，等于 24 小时。您可以在初始部署期间使用 NovaImageCachTTL tripleo-heat-templates 参数或云堆栈更新期间控制这个值：

```
parameter_defaults:
  NovaImageCacheTTL: 604800 # Default to 7 days for all compute roles
  Compute2Parameters:
    NovaImageCacheTTL: 1209600 # Override to 14 days for the Compute2 compute role
```

当您指示 playbook 预缓存 Compute 节点上已存在的镜像时，ansible 不会报告更改，但镜像的年龄重置为 0。运行 ansible 频率比 NovaImageCacheTTL 参数的值更频繁，以维护镜像的缓存。

第 11 章 TLS-E 用于 DCN

您可以在为分布式计算节点基础架构设计的云中启用 TLS （传输层安全）。您可以选择启用 TLS 以进行公共访问，或使用 TLS-e 在每个网络上启用 TLS，这允许在所有内部和外部数据流上加密。

您无法在边缘堆栈上启用公共访问，因为边缘站点没有公共端点。有关 TLS 进行公共访问的更多信息，请参阅在 Overcloud 公共端点上启用 SSL/TLS。

11.1. 使用 TLS-E 部署分布式计算节点架构

先决条件

当您使用 Red Hat Identity Manager (IdM)在 Red Hat OpenStack Platform (RHOSP)分布式计算节点架构中配置 TLS-e 时，根据为 Red Hat Identity Manager 部署的 Red Hat Enterprise Linux 版本执行以下操作。

Red Hat Enterprise Linux 8.4

1. 在 Red Hat Identity Management 节点上，允许信任子网在 ipa-ext.conf 文件中进行 ACL：

```
acl "trusted_network" {
  localnets;
  localhost;
  192.168.24.0/24;
  192.168.25.0/24;
};
```

1. 在 /etc/named/ipa-options-ext.conf 文件中，允许递归和查询缓存：

```
allow-recursion { trusted_network; };
allow-query-cache { trusted_network; };
```

2. 重启 'named-pkcs11 服务：

```
systemctl restart named-pkcs11
```

Red Hat Enterprise Linux 8.2

如果您在 Red Hat Enterprise Linux (RHEL) 8.2 上有 Red Hat Identity Manager (IdM) 8.2，您必须升级 Red Hat Enterprise Linux，然后按照 RHEL 8.4 的说明进行操作

Red Hat Enterprise Linux 7.x

如果您在 Red Hat Enterprise Linux (RHEL) 7.x 上有 Red Hat Identity Manager (IdM)，您必须手动为您的域名添加访问控制指令(ACI)。例如，如果域名是 redhat.local，请在 Red Hat Identity Manager 上运行以下命令来配置 ACI：

```
ADMIN_PASSWORD=redhat_01
DOMAIN_LEVEL_1=local
DOMAIN_LEVEL_2=redhat

cat << EOF | ldapmodify -x -D "cn=Directory Manager" -w ${ADMIN_PASSWORD}
dn: cn=dns,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}
changetype: modify
add: aci
aci: (targetattr = "aaaarecord || arecord || cnamerecord || idnsname || objectclass || ptrrecord")
(targetfilter = "(&(objectclass=idnsrecord)(|(aaaarecord=)(arecord=)(cnamerecord=)(ptrrecord=)
(idnsZoneActive=TRUE)))")(version 3.0; acl "Allow hosts to read DNS A/AAA/CNAME/PTR records";
allow (read,search,compare) userdn =
"ldap:///fqdn=*,cn=computers,cn=accounts,dc=${DOMAIN_LEVEL_2},dc=${DOMAIN_LEVEL_1}";)
EOF
```

流程

对于分布式计算节点(DCN)架构，需要使用基于 ansible 的 tripleo-ipa 方法实施 TLS-e，而不是之前的 novajoin 方法。有关使用 tripleo-ipa 部署 TLS-e 的更多信息，请参阅使用 Ansible 实施 TLS-e。

要使用 tripleo-ipa 用于 DCN 架构部署 TLS-e，您还需要完成以下步骤：

1. 如果要在边缘部署存储，请在修改后的用于边缘堆栈的 tripleo heat 模板中包括以下参数：

```
TEMPLATES=/usr/share/openstack-tripleo-heat-templates

resource_registry:
  OS::TripleO::Services::IpaClient:
    ${TEMPLATES}/deployment/ipa/ipaservices-baremetal-ansible.yaml

parameter_defaults:
  EnableEtcdInternalTLS: true
```

由于中央位置和边缘位置之间的设计差异，请不要在边缘堆栈中包含以下文件：

tls-everywhere-endpoints-dns.yaml

此文件在边缘站点中被忽略，它集的端点会被从中央堆栈导出的端点覆盖。

haproxy-public-tls-certmonger.yaml

此文件会导致部署失败，因为边缘没有公共端点。

第 12 章 创建用于外部访问的 CEPH 密钥

对 Ceph 存储的外部访问可从任何非本地站点访问 Ceph。中心位置上的 Ceph 存储是边缘(DCN)站点的外部存储，就像边缘的 Ceph 存储是中央位置的外部。

当您使用 Ceph 存储部署中央或 DCN 站点时，您可以选择将默认 openstack 密钥环用于本地和远程访问。例如，您可以创建单独的密钥供非本地站点访问。

如果您决定使用额外的 Ceph 密钥来访问外部站点，每个密钥必须具有相同的名称。密钥名称在以下示例中是外部的。

如果您使用单独的密钥进行非本地站点访问，则获得额外的安全优势，以便在不中断本地访问的情况下对外部事件进行撤销并重新发布外部密钥。但是，对外部访问使用单独的密钥将导致丢失访问某些功能，如跨可用区备份和离线卷迁移。您必须根据所需的功能集对安全公开的要求进行平衡。

默认情况下，中央和所有 DCN 站点的密钥都会被共享。

12.1. 创建用于外部访问的 CEPH 密钥

完成以下步骤，为非本地访问创建 外部 密钥。

Process

1. 创建用于外部访问的 Ceph 密钥。这个密钥是敏感的。您可以使用以下方法生成密钥：

```
python3 -c 'import os,struct,time,base64; key = os.urandom(16) ; \
header = struct.pack("<hiih", 1, int(time.time()), 0, len(key)) ; \
print(base64.b64encode(header + key).decode())'
```

2. 在您要部署的堆栈的目录中，使用密钥的输出结果创建一个 ceph_keys.yaml 环境文件，其内容如下：

```
parameter_defaults:
  CephExtraKeys:
    - name: "client.external"
      caps:
        mgr: "allow *"
        mon: "profile rbd"
```

```
    osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
  key: "AQD29WteAAAAABAAphgOjFD7nyjdYe8Lz0mQ5Q=="
  mode: "0600"
```

3. 在站点部署中包含 ceph_keys.yaml 环境文件。例如，要使用 ceph_keys.yaml 环境文件部署中央站点，请运行以下命令：

```
overcloud deploy \
    --stack central \
    --templates /usr/share/openstack-tripleo-heat-templates/ \
    ....
    -e ~/central/ceph_keys.yaml
```

12.2. 使用外部 CEPH 密钥

您只能使用已部署的密钥。有关使用 外部 密钥部署站点的详情，请参考 第 12.1 节 “创建用于外部访问的 Ceph 密钥”。这应该为中央和边缘站点完成。

- 当您部署使用中心提供的一个外部密钥的边缘站点时，请完成以下操作：

1. 为边缘站点创建 dcn_ceph_external.yaml 环境文件。您必须包含 cephx-key-client-name 选项，才能指定要包含的部署密钥。

```
sudo -E openstack overcloud export ceph \
--stack central \
--config-download-dir /var/lib/mistral \
--cephx-key-client-name external \
--output-file ~/dcn-common/dcn_ceph_external.yaml
```

2. 包含 dcn_ceph_external.yaml 文件，以便边缘站点可以访问中央站点的 Ceph 集群。包含 ceph_keys.yaml 文件，以在边缘站点为 Ceph 集群部署外部密钥。

- 当您在部署边缘站点后更新中央位置时，请确保中央位置使用 dcn 外部 密钥：

1. 确保 CephClientUserName 参数与所导出的密钥匹配。如果您使用这些示例所示的名称 external，请创建 glance_update.yaml 类似于以下内容：

```
parameter_defaults:
  GlanceEnabledImportMethods: web-download,copy-image
  GlanceBackend: rbd
  GlanceStoreDescription: 'central rbd glance store'
```

```
  CephClusterName: central
  GlanceBackendID: central
  GlanceMultistoreConfig:
  dcn0:
    GlanceBackend: rbd
   GlanceStoreDescription: 'dcn0 rbd glance store'
   CephClientUserName: 'external'
   CephClusterName: dcn0
   GlanceBackendID: dcn0
  dcn1:
   GlanceBackend: rbd
   GlanceStoreDescription: 'dcn1 rbd glance store'
   CephClientUserName: 'external'
   CephClusterName: dcn1
   GlanceBackendID: dcn1
```

2. **使用 openstack overcloud export ceph 命令，使其包含从中央位置访问 DCN 边缘访问 的外部 密钥。要做到这一点，您必须为 --stack 参数提供一个以逗号分隔的堆栈列表，并包含 cephx-key-client-name 选项：**

```
sudo -E openstack overcloud export ceph \
--stack dcn0,dcn1,dcn2 \
--config-download-dir /var/lib/mistral \
--cephx-key-client-name external \
--output-file ~/central/dcn_ceph_external.yaml
```

3. **使用原始模板重新部署中央站点，并包含新创建的 dcn_ceph_external.yaml 和 glance_update.yaml 文件。**

```
openstack overcloud deploy \
      --stack central \
      --templates /usr/share/openstack-tripleo-heat-templates/ \
      -r ~/central/central_roles.yaml \
   ...
      -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
      -e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
      -e ~/central/central-images-env.yaml \
      -e ~/central/role-counts.yaml \
      -e ~/central/site-name.yaml
      -e ~/central/ceph.yaml \
      -e ~/central/ceph_keys.yaml \
      -e ~/central/glance.yaml \
      -e ~/central/dcn_ceph_external.yaml
```

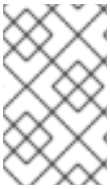
附录 A. 部署迁移选项

本节包含与 DCN 存储相关的主题，以及迁移或更改架构。

A.1. 验证边缘存储

通过测试 glance 多存储和实例创建，确保中央和边缘站点的部署正在工作。

您可以将镜像导入到本地文件系统上可用的 glance 中，或者在 web 服务器上可用。



注意

始终将镜像副本存储在中央站点，即使没有在中央位置使用该镜像的实例。

先决条件

1. 使用 glance stores-info 命令检查镜像服务可用的存储。在以下示例中，提供了三个存储：central、dcn1 和 dcn2。它们分别与 glance 分别存储在中央位置和边缘站点：

```
$ glance stores-info
+----------+----------------------------------------------------------------------------------+
| Property | Value                                                                            |
+----------+----------------------------------------------------------------------------------+
| stores   | [{"default": "true", "id": "central", "description": "central rbd glance         |
|          | store"}, {"id": "dcn0", "description": "dcn0 rbd glance store"},                  |
|          | {"id": "dcn1", "description": "dcn1 rbd glance store"}]                          |
+----------+----------------------------------------------------------------------------------+
```

A.1.1. 从本地文件导入

您必须首先将镜像上传到中央位置的存储中，然后将镜像复制到远程站点。

1. 确保您的镜像文件采用 RAW 格式。如果镜像不采用 raw 格式，则必须在将镜像导入到镜像服务前转换镜像：

```
file cirros-0.5.1-x86_64-disk.img
cirros-0.5.1-x86_64-disk.img: QEMU QCOW2 Image (v3), 117440512 bytes
```

```
qemu-img convert -f qcow2 -O raw cirros-0.5.1-x86_64-disk.img cirros-0.5.1-x86_64-
disk.raw
```

Import the image into the default back end at the central site:

```
glance image-create \
--disk-format raw --container-format bare \
--name cirros --file cirros-0.5.1-x86_64-disk.raw \
--store central
```

A.1.2. 从 Web 服务器导入镜像

如果镜像托管在 web 服务器上，您可以使用 GlanceImageImportPlugins 参数将镜像上传到多个存储。

此流程假设在 glance 中启用了默认镜像转换插件。此功能自动将 QCOW2 文件格式转换为 RAW 镜像，这是 Ceph RBD 的最佳选择。您可以通过运行 glance image-show ID | grep disk_format 来确认 glance 镜像采用 RAW 格式。

流程

1. 使用 glance 命令的 image-create-via-import 参数从 Web 服务器导入镜像。使用 --stores 参数。

```
# glance image-create-via-import \
--disk-format qcow2 \
--container-format bare \
--name cirros \
--uri http://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img \
--import-method web-download \
--stores central,dcn1
```

在本例中，qcow2 cirros 镜像从官方 Cirros 站点下载，由 glance 转换为 RAW，并导入到由 --stores 参数指定的中央站点和边缘站点 1。

或者，您可以将 --stores 替换为 --all-stores True，以将镜像上传到所有存储。

A.1.3. 将镜像复制到新站点

您可以将现有镜像从中央位置复制到边缘站点，这可让您在新创建的位置访问之前创建的镜像。

1. 将 glance 镜像的 UUID 用于复制操作：

```
ID=$(openstack image show cirros -c id -f value)

glance image-import $ID --stores dcn0,dcn1 --import-method copy-image
```

注意

在本例中，**--stores** 选项指定 **cirros** 镜像将从中央站点复制到边缘站点 **dcn1** 和 **dcn2**。或者，您可以使用 **--all-stores True** 选项，该选项将镜像上传到当前没有镜像的所有存储中。

2. 确认镜像的副本位于每个存储中。请注意，**store** 键（属性映射中的最后一个项）被设置为 **central,dcn0,dcn1** ：

```
 $ openstack image show $ID | grep properties
 | properties       | direct_url=rbd://d25504ce-459f-432d-b6fa-
79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap, locations=[{u'url:
u'rbd://d25504ce-459f-432d-b6fa-79854d786f2b/images/8083c7e7-32d8-4f7a-b1da-
0ed7884f1076/snap', u'metadata': {u'store': u'central'}}, {u'url': u'rbd://0c10d6b5-a455-4c4d-
bd53-8f2b9357c3c7/images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata':
{u'store': u'dcn0'}}, {u'url': u'rbd://8649d6c3-dcb3-4aae-8c19-8c2fe5a853ac/images/8083c7e7-
32d8-4f7a-b1da-0ed7884f1076/snap', u'metadata': {u'store': u'dcn1'}}],
os_glance_failed_import=', os_glance_importing_to_stores=', os_hash_algo='sha512,
os_hash_value=b795f047a1b10ba0b7c95b43b2a481a59289dc4cf2e49845e60b194a911819d
3ada03767bbba4143b44c93fd7f66c96c5a621e28dff51d1196dae64974ce240e,
os_hidden=False, stores=central,dcn0,dcn1 |
```

注意

即使没有在该站点上使用虚拟机，也始终将镜像副本存储在中央站点。

A.1.4. 确认边缘站点中的实例可以使用基于镜像的卷引导

您可以使用边缘站点中的镜像来创建持久的根卷。

流程

1. 识别要作为卷创建的镜像 **ID**，并将该 **ID** 传递给 **openstack volume create** 命令：

```
IMG_ID=$(openstack image show cirros -c id -f value)
openstack volume create --size 8 --availability-zone dcn0 pet-volume-dcn0 --image $IMG_ID
```

2. 识别新创建的卷的卷 ID，并将其传递给 **openstack server create** 命令：

```
VOL_ID=$(openstack volume show -f value -c id pet-volume-dcn0)
openstack server create --flavor tiny --key-name dcn0-key --network dcn0-network --security-group basic --availability-zone dcn0 --volume $VOL_ID pet-server-dcn0
```

3. 您可以通过在 **dcn0** 边缘站点的 **ceph-mon** 容器中运行 **rbd** 命令来验证卷是否基于镜像，以列出卷池。

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                                        SIZE  PARENT                                                FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2      excl
$
```

4. 确认您可以创建实例的根卷的 **cinder** 快照。确保服务器停止以静止数据以创建干净的快照。使用 **--force** 选项，因为实例关闭时卷状态会一直 使用。

```
openstack server stop pet-server-dcn0
openstack volume snapshot create pet-volume-dcn0-snap --volume $VOL_ID --force
openstack server start pet-server-dcn0
```

5. 列出 **dcn0 Ceph** 集群上 **volumes** 池的内容，以显示新创建的快照。

```
$ sudo podman exec ceph-mon-$HOSTNAME rbd --cluster dcn0 -p volumes ls -l
NAME                                                                SIZE  PARENT
FMT PROT LOCK
volume-28c6fc32-047b-4306-ad2d-de2be02716b7                                  8 GiB
images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2      excl
volume-28c6fc32-047b-4306-ad2d-de2be02716b7@snapshot-a1ca8602-6819-45b4-a228-b4cd3e5adf60 8 GiB images/8083c7e7-32d8-4f7a-b1da-0ed7884f1076@snap   2 yes
```

A.1.5. 确认可以在站点之间创建和复制镜像快照

1. 验证您可以在 **dcn0** 网站创建新镜像。确保服务器停止以静止数据来创建干净的快照：

```
NOVA_ID=$(openstack server show pet-server-dcn0 -f value -c id)
openstack server stop $NOVA_ID
openstack server image create --name cirros-snapshot $NOVA_ID
openstack server start $NOVA_ID
```

2. 将镜像从 dcn0 边缘站点复制到 hub 位置，这是 glance 的默认后端：

```
IMAGE_ID=$(openstack image show cirros-snapshot -f value -c id)
glance image-import $IMAGE_ID --stores central --import-method copy-image
```

有关 glance 多存储操作的更多信息，请参阅 具有多个存储的镜像服务。

A.2. 迁移到 SPINE 和 LEAF 部署

可将预先存在的网络配置的现有云迁移到具有 spine leaf 架构的一个。因此，需要以下条件：

- 所有裸机端口都必须将其 physical-network 属性值设置为 ctlplane。
- 参数 enable_routed_networks 被添加，在 undercloud.conf 中设置为 true，随后重新运行 undercloud 安装命令，openstack undercloud install。

重新部署 undercloud 后，overcloud 被视为一个 spine leaf0，它带有一个 leaf leaf0。您可以按照以下步骤为部署添加额外的置备。

1. 将所需的子网添加到 undercloud.conf，如 在 undercloud 中配置路由的 spine-leaf 所示。
2. 重新运行 undercloud 安装命令 openstack undercloud install。
3. 分别将所需的额外网络和角色添加到 overcloud 模板 network_data.yaml 和 roles_data.yaml 中。

注意

如果您在网络配置文件中使用 {{network.name}}InterfaceRoutes 参数，则需要确保 NetworkDeploymentActions 参数包含值 UPDATE。

```
NetworkDeploymentActions: ['CREATE','UPDATE'])
```

4. 最后，重新运行 overcloud 安装脚本，其中包含用于您的云部署的所有相关 heat 模板。

A.3. 迁移到多堆栈部署

您可以通过将现有部署视为中央站点并添加额外的边缘站点，从单一堆栈部署迁移到多堆栈部署。

从本发行版本中从单堆栈迁移到多堆栈的功能 是技术预览，因此不受红帽完全支持。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。

您无法分割现有的堆栈。如果需要，您可以缩减现有堆栈以移除计算节点。然后可将这些计算节点添加到边缘站点。

注意

如果删除了所有计算节点，此操作会创建工作负载中断。

A.4. 在边缘站点间备份和恢复

您可以在边缘站点和可用区中的分布式计算节点(DCN)架构中备份和恢复块存储服务(cinder)卷。cinder-backup 服务在中央可用区(AZ)中运行，备份存储在中央 AZ 中。块存储服务不会在 DCN 站点存储备份。

先决条件

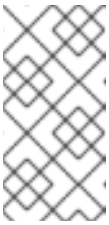
- 中央站点使用位于 /usr/share/openstack-tripleo-heat-templates/environments 中的 cinder-backup.yaml 环境文件进行部署。如需更多信息，请参阅块存储备份服务部署。
- Block Storage 服务(cinder) CLI 可用。
- 所有站点都必须使用通用的 openstack cephx 客户端名称。有关更多信息，请参阅为外部访问创建 Ceph 密钥。

流程

1. 在第一个 DCN 站点中创建卷备份：

```
$ cinder --os-volume-api-version 3.51 backup-create --name <volume_backup> --availability-zone <az_central> <edge_volume>
```

- 将 <volume_backup > 替换为卷备份的名称。
- 将 <az_central > 替换为托管 cinder-backup 服务的中央可用区的名称。
- 将 < edge_volume > 替换为您要备份的卷的名称。



注意

如果遇到 Ceph 密钥环的问题，您可能需要重启 cinder-backup 容器，以便主机中的密钥环成功复制到容器。

2. 将备份恢复到第二个 DCN 站点中的新卷：

```
$ cinder --os-volume-api-version 3.51 create --availability-zone <az_2> --name <new_volume> --backup-id <volume_backup> <volume_size>
```

- 将 &lt ;az_2 > 替换为您要恢复备份的可用区的名称。
- 将 <new_volume > 替换为新卷的名称。
- 将 &lt ;volume_backup > 替换为您在上一步中创建的卷备份的名称。
- 将 &lt ;volume_size > 替换为等于或大于原始卷大小的值。

A.5. 删除 DCN 网站

要删除边缘站点，请从 Red Hat OpenStack 部署中删除堆栈、存储和相关服务。在以下示例中，dcn2 正在停用并从 DCN （分布式计算节点）部署中删除。调整命令以适合您的环境。

先决条件

- *您有一个站点已停用，它是完全部署的 Red Hat OpenStack Platform 集群的一部分*
- *必须删除边缘站点上活跃的所有实例。*
- *必须删除要删除的边缘站点上激活的所有卷。*

流程

1. *如果此站点置备了 Red Hat Ceph Storage，您必须首先删除与您要删除的边缘位置关联的 glance 存储、Ceph 配置和相关权限。如果您还没有在边缘部署存储，您可以继续步骤 2：*

   a. *登录到 Red Hat Openstack Platform director，并提供中央站点的 RC 凭证文件：*

   ```
   source /home/stack/centralrc
   ```

   b. *删除与堆栈关联的所有镜像：*

   ```
   glance stores-delete --store dcn2 <IMAGE>
   ```

   将 <_IMAGE_> 替换为您要删除的镜像的名称。

   c. *从 dcn_ceph_external.yaml 文件中删除与堆栈关联的小节。在本例中，删除 dcn2 配置。*

   ```
   ...
     - cluster: "dcn1"
       fsid: "e70c3c80-4eca-4f1a-8cbb-3e9753b401c9"
       external_cluster_mon_ips: "172.23.2.58, 172.23.2.153, 172.23.2.32"
       keys:
         - name: "client.external"
           caps:
             mgr: "allow *"
             mon: "profile rbd"
             osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
           key: "AQD4Ae9gAAAAABAAeeG5N0E3Jka7bXRDhB8CtQ=="
   ```

```
        mode: "0600"
    dashboard_enabled: false
    ceph_conf_overrides:
      client:
        keyring: /etc/ceph/dcn1.client.external.keyring
-   - cluster: "dcn2"
-     fsid: "c699978c-876d-4c65-984f-2179d22244ea"
-     external_cluster_mon_ips: "172.23.3.239, 172.23.3.99, 172.23.3.67"
-     keys:
-      - name: "client.external"
-        caps:
-          mgr: "allow *"
-          mon: "profile rbd"
-          osd: "profile rbd pool=vms, profile rbd pool=volumes, profile rbd pool=images"
-        key: "AQA1Du9gAAAAABAAMOBAczLIw1SzBR7vxwYqJg=="
-        mode: "0600"
-     dashboard_enabled: false
-     ceph_conf_overrides:
-       client:
-         keyring: /etc/ceph/dcn2.client.external.keyring
```

d. ***从 glance_update.yaml 文件中删除与堆栈关联的小节。在本例中，删除 dcn2 配置。***

```
parameter_defaults:
  GlanceMultistoreConfig:
    dcn0:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn0 rbd glance store'
      CephClientUserName: 'openstack'
      CephClusterName: dcn0
    dcn1:
      GlanceBackend: rbd
      GlanceStoreDescription: 'dcn1 rbd glance store'
      CephClientUserName: 'openstack'
      CephClusterName: dcn1
-   dcn2:
-     GlanceBackend: rbd
-     GlanceStoreDescription: 'dcn2 rbd glance store'
-     CephClientUserName: 'openstack'
-     CephClusterName: dcn2
```

e. ***删除与您要删除的站点相关的 /etc/ceph 中的文件。在每个 Controller 节点上执行此步骤。***

```
source /home/stack/stackrc
for i in $(openstack server list | awk '/controller/ {print $8}' | cut -d= -f2); do
    ssh heat-admin@${i} sudo -E \
    rm -f /etc/ceph/dcn2.conf /etc/ceph/dcn2.client.openstack.keyring
done
```

f. 为堆栈禁用 cinder-volume 服务：

i. 识别您要删除的堆栈的 cinder-volume 服务：

```
source /home/stack/centralrc
VOLUME_HOST=openstack volume service list --service cinder-volume
```

ii. 禁用 cinder-volume 服务

```
openstack volume service set --disable ${VOLUME_HOST} cinder-volume
```

iii. 从中央位置其中一个 Controller 节点上的 cinder_api 容器中删除 cinder-volume 服务：

```
ssh heat-admin@CONTROLLER_IP sudo podman exec cinder_api cinder-manage
service remove cinder-volume ${VOLUME_HOST}
```

2. 如果使用 OVS，请清理您要删除的堆栈的 neutron 代理：

a. 识别网络代理的 UUID：

```
source /home/stack/centralrc
openstack network agent list
```

b. 使用上一步中的 UUID 来识别要删除的网络代理。为每个 UUID 运行以下命令：

```
openstack network agent delete $UUID
```

注意

您无法删除 OVN 代理。

3. 删除与堆栈关联的计算服务：

a. 查找与 dcn2 关联的计算服务的 ID。您可以将唯一主机名用于此步骤。如果主机名包含

stacknames，您可以使用 grep 来只列出您要删除的计算服务：

```
source /home/stack/centralrc
openstack compute service list | grep dcn2
```

b. **使用第一个字段中显示的 ID 删除计算服务。为每个 ID 运行以下命令：**

```
openstack compute service delete <ID>
```

4. **删除与站点关联的主机聚合：**

```
source /home/stack/centralrc
openstack aggregate delete dcn2
```

5. **从 /home/stack/central/network/network_data.yaml 文件中删除您要删除的堆栈的子网：**

```
 - name: Tenant
   name_lower: tenant
   vip: false  # Tenant network does not use VIPs
   vlan: 1189
   ip_subnet: '172.19.1.0/24'
   allocation_pools: [{'start': '172.19.1.4', 'end': '172.19.1.250'}]
   gateway_ip: '172.19.1.254'
   subnets:
     tenant_leaf1:
       vlan: 1179
       ip_subnet: '172.19.2.0/24'
       allocation_pools: [{'start': '172.19.2.4', 'end': '172.19.2.250'}]
       gateway_ip: '172.19.2.254'
-    tenant_leaf2:
-      vlan: 1169
-      ip_subnet: '172.19.3.0/24'
-      allocation_pools: [{'start': '172.19.3.4', 'end': '172.19.3.250'}]
-      gateway_ip: '172.19.3.254
```

6. **删除堆栈：**

```
source /home/stack/stackrc
openstack overcloud delete -y dcn2
```

7. **可选：从停用的站点中删除裸机节点：**

a.

列出节点，并确定与已删除的 stack dcn2 关联的节点

```
source /home/stack/stackrc
openstack baremetal node list
```

b.

使用上一步中的节点 UUID 从 ironic 中删除每个节点：

```
openstack baremetal node delete $UUID
```