Red Hat OpenStack Platform 17.1

部署分布式 Compute 节点(DCN)架构

Red Hat OpenStack Platform 的边缘和存储配置

Last Updated: 2024-06-06

Red Hat OpenStack Platform 17.1 部署分布式 Compute 节点(DCN)架构

Red Hat OpenStack Platform 的边缘和存储配置

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)，以实现边缘站点操作功能，并实现 heat 堆栈分离。每个站点可以拥有自己的 Ceph 存储后端用于镜像服务(glance)多存储。

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

分布式计算节点(DCN)架构适用于边缘用例，允许在共享通用中央化 control plane 时远程部署远程计算和存储节点。DCN 架构允许您根据操作需求进行战略性定位，以实现更高的性能。

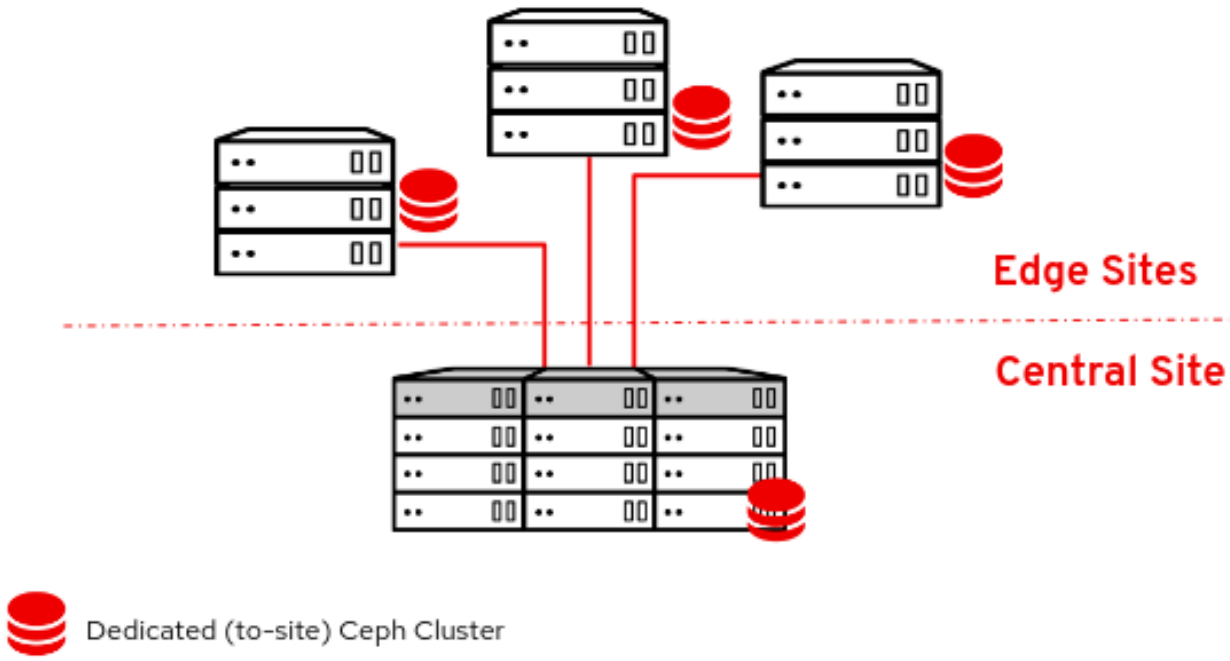
中央位置可由任何角色组成，但至少需要三个控制器。Compute 节点可以存在于边缘，也可以位于中央位置。

DCN 架构是一个 hub 和 spoke 路由网络部署。DCN 与通过 Red Hat OpenStack Platform director 进行路由置备和 control plane 联网的 spine 和 leaf 部署类似。

- hub 是具有核心路由器和数据中心网关(DC-GW)的中央站点。
- 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。
- 未安装 Pacemaker。
- Block Storage 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 所需的软件和最低版本：

平台	版本	选填
Red Hat Enterprise Linux	9.2	否
Red Hat OpenStack Platform	17.1	否

平台	版本	选填
Red Hat Ceph Storage	5	是

1.2. MULTISTACK 设计

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

除非部署是从 Red Hat OpenStack Platform 13 升级，否则不支持将 DCN 架构作为单一堆栈管理。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

您可以在没有存储的情况下部署每个边缘站点，也可以使用 Ceph 在超融合节点上部署。您部署的存储专用于您部署它的站点。

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，可以使用其他工具，以便您可以在计算服务(nova)缓存中缓存和存储镜像。在 nova 中缓存 glance 镜像，通过避免跨 WAN 链接下载镜像的过程，为实例提供更快的引导时间。更多信息请参阅 第 11 章 *将缓存 glance 镜像预缓存到 nova*。

1.4. DCN 边缘

使用分布式 Compute 节点架构，您可以在中央站点上部署控制节点，并使用这些控制器来管理地理分散的边缘站点。当您部署边缘站点时，您仅部署计算节点，这使得边缘站点架构与 Red Hat OpenStack Platform 的传统部署不同。当您在边缘站点启动实例时，所需的镜像会自动复制到本地镜像服务(glance)存储中。您可以使用 glance multistore 将镜像从中央镜像存储复制到边缘站点，以便在实例启动期间节省时间。如需更多信息，请参阅 具有多个存储的镜像服务。

在边缘站点：

- control plane 服务在中央位置远程运行。
- Pacemaker 不在 DCN 站点中运行。
- Block Storage 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

规划 DCN 架构时，请检查您所需的技术是否可用和支持。

2.1. DCN 构架上存储的注意事项

DCN 架构目前不支持以下功能：

- 在边缘站点之间复制卷快照。您可以通过从卷创建镜像并使用 glance 复制镜像来解决此问题。复制镜像后，您可以从中创建一个卷。
- 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 镜像功能。
- 实例迁移、实时或冷站点，可以是边缘站点，或者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，请参阅可在站点之间创建并复制镜像快照。

另外，您必须考虑以下内容：

- 您必须将镜像上传到中央位置，然后才能将它们复制到边缘站点；在中央位置，每个镜像的副本都必须存在于镜像服务(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。
- 对于每个站点，分配一个唯一的可用区，并为 NovaComputeAvailabilityZone 和 CinderStorageAvailabilityZone 参数使用相同的值。
- 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：

- DPDK 节点上的 DHCP
- TC Flower Hardware Offload 的 conntrack

TC Flower Hardware Offload 的 conntrack 在 DCN 上作为技术预览提供，因此红帽不完全支持使用这些解决方案。此功能应该只与 DCN 一起使用，不应在生产环境中部署。有关技术预览功能的更多信息，请参阅覆盖范围详情。

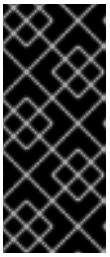
完全支持以下 ML2/OVS 技术：

- DPDK 节点上没有 DHCP 的 OVS-DPDK
- SR-IOV
- TC 流硬件卸载，没有 conntrack
- 带有边缘网络节点的 Neutron 可用区(AZ)，每个站点有一个 AZ

- 路由提供商网络

以下 ML2/OVN 网络技术被完全支持：

- DPDK 节点上没有 DHCP 的 OVS-DPDK
- SR-IOV （没有 DHCP）
- TC 流硬件卸载，没有 conntrack
- 路由提供商网络
- 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，*请参阅配置 Red Hat OpenStack Platform 网络 中的 使用 ML2/OVN 配置网络服务 可用域*。

另外，您必须考虑以下内容：

- 网络延迟：平衡往返时间(RTT)中测量的延迟，预期的并发 API 操作数量可以保持可接受的性能。最大 TCP/IP 吞吐量与 RTT 相反。您可以通过调优内核 TCP 参数来缓解具有高带宽的高延迟连接的问题。如果跨站点通信超过 100 ms，请联系红帽支持。

- Network drop outs：如果边缘站点临时丢失与中央站点的连接，则在中断期间，无法在受影响的边缘站点执行 OpenStack control plane API 或 CLI 操作。例如，边缘站点的 Compute 节点无法创建实例的快照，发出身份验证令牌或删除镜像。在此中断期间，一般的 OpenStack control plane API 和 CLI 操作仍可以正常工作，并可以继续为具有工作连接的其它边缘站点提供服务。镜像类型 ：在使用 Ceph 存储部署 DCN 架构时，您必须使用原始镜像。

- 镜像大小：

  - Overcloud 节点镜像 - overcloud 节点镜像从中央 undercloud 节点下载。这些镜像可能是在置备过程中在所有必要网络上传输的大型文件，从中央站点传输到边缘站点。

  - 实例镜像：如果边缘没有块存储，则镜像服务镜像在第一次使用时会遍历 WAN。镜像在本地复制或缓存到目标边缘节点，供以后使用。glance 镜像的大小限制没有大小限制。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，则镜像会在 WAN 异步复制，以便在边缘进行更快的引导时间。

- 提供商网络：这是 DCN 部署的推荐网络方法。如果您在远程站点中使用提供商网络，则必须考虑网络服务(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，如果您在边缘站点 A 中只使用提供商网络，您必须确保不要尝试附加到边缘站点 B 中的提供商网络。这是因为当将其绑定到 Compute 节点时，在提供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 网络中有一个限制：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 节点列表并相互自动形成隧道；隧道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 节点都会形成一个隧道，与其他 Compute 节点和 Controller 节点组成，无论它们是本地或远程的。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 节点都有相同的网桥映射，并且所有 VLAN 都在每个站点都可用。

- 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。
- 如果没有预置备边缘服务器，您必须配置 DHCP 转发以便在路由片段上内省和置备。
- 路由必须在云上配置，或者在将每个边缘站点连接到 hub 的联网基础架构中进行配置。您应该实施一种网络设计，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。

第 3 章 在 UNDERCLOUD 中配置路由的 SPINE-LEAF

本节介绍了如何配置 undercloud 以适应带有可组合网络的路由的 spine-leaf 的用例。

3.1. 配置 SPINE LEAF PROVISIONING 网络

要为您的 spine leaf infrastructure 配置置备网络，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 的外部面向外部的 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 中：

```
undercloud_admin_host = 192.168.10.2
```

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

```
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 物理 L2 段本地关联的子网：

```
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：

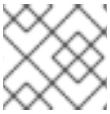
```
[stack@director ~]$ openstack undercloud install
```

此配置会在 provisioning 网络或 control plane 上创建三个子网。overcloud 使用每个网络来调配各个子叶中的系统。

为确保 DHCP 请求正确中继到 undercloud，您可能需要配置 DHCP 转发。

3.2. 配置 DHCP 转发

您可以在连接到您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：

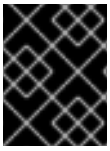
- 内省 DHCP 服务器。
- 置备 DHCP 服务器。

您必须配置 DHCP 转发，以将 DHCP 请求转发到 undercloud 上的两个 DHCP 服务器。

您可以将 UDP 广播用于支持它的设备，将 DHCP 请求中继到连接 undercloud 置备网络的 L2 网络段。或者，您可以使用 UDP 单播，它将 DHCP 请求中继到特定的 IP 地址。

注意

在特定设备类型上配置 DHCP 转发超出了本文档的范围。作为参考，本文档使用 ISC DHCP 软件中的实现提供 DHCP 转发配置示例。如需更多信息，请参阅 man page dhcrelay (8)。



重要

对于某些转发，需要 DHCP 选项 79，特别是为 DHCPv6 地址提供服务的转发，在原始 MAC 地址上不会传递的中继。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 服务器或服务器的 L2 网络段。网络段中的所有设备都会接收广播流量。在使用 UDP 广播时，undercloud 上的两个 DHCP 服务器都会接收转发的 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

指定连接到转发 DHCP 请求的 L2 网络段的接口。

IP 网络地址

指定转发 DHCP 请求的 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 **undercloud.conf** 文件中的 **inspection_interface** 定义的接口。如果您还没有设置此参数，undercloud 的默认接口是 **br-ctlplane**。

注意

通常使用 **br-ctlplane** 接口进行内省。您在 **undercloud.conf** 文件中作为 **local_ip** 定义的 IP 地址位于 **br-ctlplane** 接口上。

分配给 Neutron DHCP 命名空间的 IP 地址是您在 **undercloud.conf** 文件中为 **local_subnet** 配置的 IP 范围中可用的第一个地址。IP 范围中的第一个地址是您在配置中定义为 **dhcp_start** 的第一个地址。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 范围内的第一个地址。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：

- 用于转发传入的 DHCP 请求的接口： **eth1**、**eth2** 和 **eth3**。
- 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 ：**192.168.10.1**。
- 用于调配的 DHCP 服务器正在侦听 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 配置执行以下任务：

- 配置用于 provisioning 网络的 VLAN。
- 添加 leaf 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. 为叶节点设计角色

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 leaf。完成以下步骤以创建并分配每个类别到角色。

流程

1. Source **stackrc** 文件：

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

2. 检索节点列表来识别它们的 UUID：

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

3. 为您要为角色指定的每个裸机节点分配带有标识其叶网络和角色的自定义资源类。

```
openstack baremetal node set \
 --resource-class baremetal.<ROLE> <node>
```

   - 将 <ROLE> 替换为标识角色的名称。
   - 将 <node> 替换为裸机节点的 ID。
     例如，输入以下命令将 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
```

4. 将每个角色添加到 **overcloud-baremetal-deploy.yaml** 中（如果尚未定义）。
5. 定义您要分配给角色节点的资源类：

```
- name: <role>
  count: 1
  defaults:
    resource_class: baremetal.<ROLE>
```

- 将 <role> 替换为角色的名称。
- 将 <ROLE> 替换为标识角色的名称。

6. 在 baremetal-deploy.yaml 文件中，定义您要分配给角色节点的资源类。指定您要部署的角色、配置集、数量和关联的网络：

```
- name: <role>
  count: 1
  hostname_format: <role>-%index%
  ansible_playbooks:
    - playbook: bm-deploy-playbook.yaml
  defaults:
    resource_class: baremetal.<ROLE>
    profile: control
    networks:
      - network: external
        subnet: external_subnet
      - network: internal_api
        subnet: internal_api_subnet01
      - network: storage
        subnet: storage_subnet01
      - network: storage_mgmt
        subnet: storage_mgmt_subnet01
      - network: tenant
        subnet: tenant_subnet01
    network_config:
      template: templates/multiple_nics/multiple_nics_dvr.j2
      default_route_network:
        - external
```

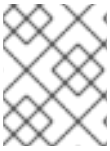
- 将 <role> 替换为角色的名称。
- 将 <ROLE> 替换为标识角色的名称。

注意

您必须在 **/home/** stack/<stack> 中为每个要部署的堆栈创建一个 **baremetal-deploy.yaml** 环境文件。

3.4. 将裸机节点端口映射到 CONTROL PLANE 网络片段

要在 L3 路由网络上启用部署，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** 的子网的物理网络名称始终命名为 **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 **注册或可 管理状态**。如果裸机节点不在这些状态之一，则在 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. 检查哪些裸机端口与哪个裸机节点关联：

```
$ 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 为 **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 之前内省节点。包含 **--all-manageable** 和 **--provide** 选项，以设置可用于部署的节点：

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

3.5. 将新的 LEAF 添加到 SPINE-LEAF PROVISIONING 网络

在增加可包括添加新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 上调配叶时，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 **stackrc** 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，并为您要添加的 leaf 添加一个新子网。
   子网代表路由 spine 和 leaf 中的 L2 片段：

   示例

   在本例中，新子网(**leaf3**)已为新的叶(**leaf3**)添加：

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

   b. 为添加的子网创建一个部分。

   示例

   在本例中，为新子网 (**leaf3**) 增加了 **[leaf3]** 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

[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

- 向 spine-leaf 部署中添加一个新的 leaf

第 4 章 为 DCN 部署准备 OVERCLOUD 模板

4.1. 使用单独的 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：

- 已安装的 Red Hat OpenStack Platform director 17.1 实例。
- 对于 Ceph Storage 用户：访问 Red Hat Ceph Storage 5。
- 对于中央位置：三个能够充当中央 Controller 节点的节点。所有三个 Controller 节点必须位于同一 heat 堆栈中。您无法将 Controller 节点或任何 control plane 服务拆分到单独的 heat 堆栈中。
- 如果您计划在边缘部署 Ceph 存储，Ceph 存储在中央位置是必需的。
- 对于每个额外的 DCN 网站：三个 HCI 计算节点。
- 所有节点必须预置备，或者从中央部署网络进行 PXE 引导。您可以使用 DHCP 转发为 DCN 启用此连接。
- 所有节点都已被 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，您的 overcloud 将相同的主机名用于不同堆栈中的分布式计算节点。确保分布式计算节点使用 %stackname% 前缀来区分来自不同边缘站点的节点。例如，如果您部署了两个名为 **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 命令。
- 提供 **centralrc** 身份验证文件，以在边缘站点以及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. 单独的 HEAT 堆栈部署的限制

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，在分布式计算节点(DCN)部署中是必需的。
- Ironic DHCP Relay - 本指南不包括如何配置带有 DHCP 中继的 Ironic。

4.3. 设计您的独立 HEAT 堆栈部署

要在单独的 heat 堆栈内对部署进行分段，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。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一个单独的可用区(AZ)。

4.4. 管理独立的 HEAT 堆栈

本指南中的流程演示了如何为三个 heat 堆栈组织环境文件： **中央**、**dcn0** 和 **dcn1**。红帽建议将每个 heat 堆栈的模板存储在单独的目录中，以保持每个部署的信息隔离。

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
```

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

touch 用于提供文件组织的示例。每个文件必须包含成功部署的适当内容。

4.5. 检索容器镜像

使用以下流程及其示例文件内容，检索使用独立 heat 堆栈部署所需的容器镜像。您必须使用边缘站点的环境文件运行 **openstack container image prepare** 命令来确保包含用于可选或特定于边缘服务的容器镜像。

如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform 指南中的准备容器镜像*。https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/17.1/html/installing_and_managing_red_hat_openstack_platform_with_dir for-director-installation#proc_preparing-container-images_preparing-for-director-installation

流程

1. 将 Registry Service Account 凭证添加到 **containers.yaml** 中。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ceph_namespace: registry.redhat.io/rhceph
      ceph_image: rhceph-6-rhel9
```

```
    ceph_tag: latest
    name_prefix: openstack-
    namespace: registry.redhat.io/rhosp17-rhel9
    tag: latest
ContainerImageRegistryCredentials:
  # https://access.redhat.com/RegistryAuthentication
  registry.redhat.io:
    registry-service-account-username: registry-service-account-password
```

2. 以 **images-env.yaml** 的形式生成环境文件：

```
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 部署过程的一部分。

4.6. 为边缘创建快速数据路径角色

要在边缘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，您可以包含新创建的角色，以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 **roles** 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 文件中包括带有 **+** 的参数，但不存在于 DistributedComputeDpdk.yaml 文件中。在新的 **DistributedComputeDpdk.yaml** 文件中包括这些参数。

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
```

您可以使用这个相同的方法为 SR-IOV 创建快速数据路径角色，或者边缘的 SR-IOV 和 DPDK 的组合来满足您的要求。

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 6.2 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 7.4 节 “使用超融合存储部署边缘站点”

第 5 章 安装中央位置

当您使用分布式计算节点(DCN)架构部署 Red Hat OpenStack 平台时，您必须提前决定您的存储策略。如果您在没有 Red Hat Ceph Storage 的情况下部署 Red Hat OpenStack Platform，则无法使用 Red Hat Ceph 存储部署任何边缘站点。此外，您无法在稍后重新部署后将 Red Hat Ceph Storage 添加到中央位置的选项。

当您为分布式计算节点(DCN)架构部署中心位置时，您可以部署集群：

- 带有或没有 Compute 节点
- 没有 Red Hat Ceph Storage

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)作为镜像服务(glance)的后端，您可以在边缘站点部署分布式计算节点集群，而无需在边缘站点进行块存储。因为每个架构的不同角色和网络配置集不同，在没有块存储的情况下部署的站点无法更新，以具有块存储。

重要：以下流程使用 lvm 作为 Cinder 的后端，它不支持用于生产环境。您必须部署一个经过认证的块存储解决方案，作为 Cinder 的后端。

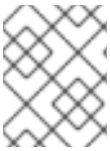
以类似于典型的 overcloud 部署的方式部署中央控制器集群。此集群不需要任何 Compute 节点，因此您可以将计算数设置为 **0** 以覆盖默认的 **1**。中央控制器具有特定存储和 Oslo 配置要求。使用以下步骤解决这些问题。

先决条件

- 您必须创建特定于环境的 **network_data.yaml** 和 **vip_data.yaml** 文件。您可以在 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 中找到示例文件。
- 您必须创建一个特定于环境的 **overcloud-baremetal-deploy.yaml** 文件。有关更多信息，请参阅为 overcloud 置备裸机节点。

流程

以下流程概述了中央位置初始部署的步骤。



注意

以下步骤详细说明了与没有 glance 多存储示例 DCN 部署关联的部署命令和环境文件。这些步骤不包括与不相关的，但必要的配置方面，如网络。

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

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

3. 生成环境文件：

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

4. 在主目录中，为您计划部署的每个堆栈创建目录。将中央位置的 **network_data.yaml**、**vip_data.yaml** 和 **overcloud-baremetal-deploy.yaml** 模板移到 **/home/stack/central/**。

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

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. 调配 overcloud 的网络。此命令取 overcloud 网络的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 为 overcloud 调配虚拟 IP。此命令取虚拟 IP 的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. 置备裸机实例。该命令为裸机节点使用一个定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. 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
  GlanceBackend: swift
```

- **ControllerCount: 3** 指定将部署三个节点。它们将 swift 用于 glance，lvm 用于 cinder，并且托管边缘计算节点的 control-plane 服务。
- **ComputeCount: 0** 是一个可选参数，用于阻止 Compute 节点使用中央 Controller 节点进行部署。
- **GlanceBackend: swift** 使用 Object Storage (swift)作为 Image Service (glance)后端。生成的配置与分布式计算节点(DCN)交互，其方式如下：

- 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

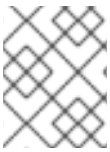
中央 Controller 节点必须能够连接到分布式计算节点(DCN)站点。中央 Controller 节点可以使用路由的第 3 层连接。

9. 在 **site-name.yaml** 环境文件中配置站点的命名约定。Nova 可用区，Cinder 存储可用域必须匹配：

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

10. 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：

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-n /home/stack/central/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml
```



注意

您必须在 **openstack overcloud deploy** 命令中包含用于配置网络的 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 配置叶型网络。

5.2. 使用存储部署中央站点

要将带有多个存储和 Ceph 存储的镜像服务部署为后端，请完成以下步骤：

先决条件

- 您必须创建特定于环境的 **network_data.yaml** 和 **vip_data.yaml** 文件。您可以在 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 中找到示例文件。
- 您必须创建一个特定于环境的 **overcloud-baremetal-deploy.yaml** 文件。有关更多信息，请参阅为 overcloud 置备裸机节点。
- 在中央位置和每个可用区中有 Ceph 集群硬件，或者在每个地理位置需要存储服务。
- 您有三个镜像服务(glance)服务器的硬件，位于中央位置和每个可用区，或者在每个需要存储服务的地理位置。在边缘位置，镜像服务被部署到 DistributedComputeHCI 节点。

流程

部署 Red Hat OpenStack Platform 中央位置，以便镜像服务(glance)可与多个存储一起使用。

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

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

3. 生成环境文件 /home/stack/central/central-images-env.yaml

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

4. 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 /home/stack/central/central_roles.yaml
```

5. 在主目录中，为您计划部署的每个堆栈创建目录。将中央位置的 **network_data.yaml**、**vip_data.yaml** 和 **overcloud-baremetal-deploy.yaml** 模板移到 **/home/stack/central/**。

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

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

6. 调配 overcloud 的网络。此命令取 overcloud 网络的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. 为 overcloud 调配虚拟 IP。此命令取虚拟 IP 的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

8. 置备裸机实例。该命令为裸机节点使用一个定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud node provision \
--stack central \
--network-config \
```

```
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

9. 如果要使用超融合存储部署中央位置，则必须使用以下参数创建 **initial-ceph.conf** 配置文件：如需更多信息，请参阅为 HCI 配置 Red Hat Ceph Storage 集群 ：

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

10. 使用 **deployed_metal.yaml** 文件作为 **openstack overcloud ceph deploy** 命令的输入。**openstack overcloud ceph deploy 命令输出** 描述部署的 Ceph 集群的 yaml 文件：

```
openstack overcloud ceph deploy \
--stack central \
/home/stack/central/deployed_metal.yaml \
--config /home/stack/central/initial-ceph.conf \ 1
--output /home/stack/central/deployed_ceph.yaml \
--container-image-prepare /home/stack/containers.yaml \
--network-data /home/stack/network-data.yaml \
--cluster central \
--roles-data /home/stack/central/central_roles.yaml
```

1 仅在部署超融合基础架构时包括 initial-ceph.com。

11. 在继续操作前，验证功能 Ceph 部署。使用 **ssh** 连接到运行 **ceph-mon** 服务的服务器。在 HCI 部署中，这是控制器节点。运行以下命令：

```
cephadm shell --config /etc/ceph/central.conf \
--keyring /etc/ceph/central.client.admin.keyring
```

注意

您必须使用 **--config** 和 **--keyring** 参数。

12. 在 **site-name.yaml** 环境文件中配置站点的命名约定。Nova 可用区和 Cinder 存储可用域必须匹配：

```
parameter_defaults:
  NovaComputeAvailabilityZone: central
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: central
  NovaCrossAZAttach: false
  CinderStorageAvailabilityZone: central
  GlanceBackendID: central
```

13. 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
```

14. 为中央位置部署堆栈：

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r /home/stack/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e ~/central/glance.yaml
```

15. 为中央位置部署 overcloud 后，导出边缘站点的额外堆栈部署的输入数据并将其放置在 **/home/stack/overcloud-deploy** 目录中。确保存在 **central-export.yaml** 文件：

```
stat /home/stack/overcloud-deploy/central/central-export.yaml
```

16. 导出 Ceph 特定数据：

```
openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

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，并集成预先部署的 Red Hat Ceph Storage 解决方案。当您在没有 director 的情况下部署 Red Hat Ceph Storage 时，director 没有环境中 Red Hat Ceph 存储的信息。您无法运行 **openstack overcloud export ceph 命令**，并且必须手动创建 **central_ceph_external.yaml**。

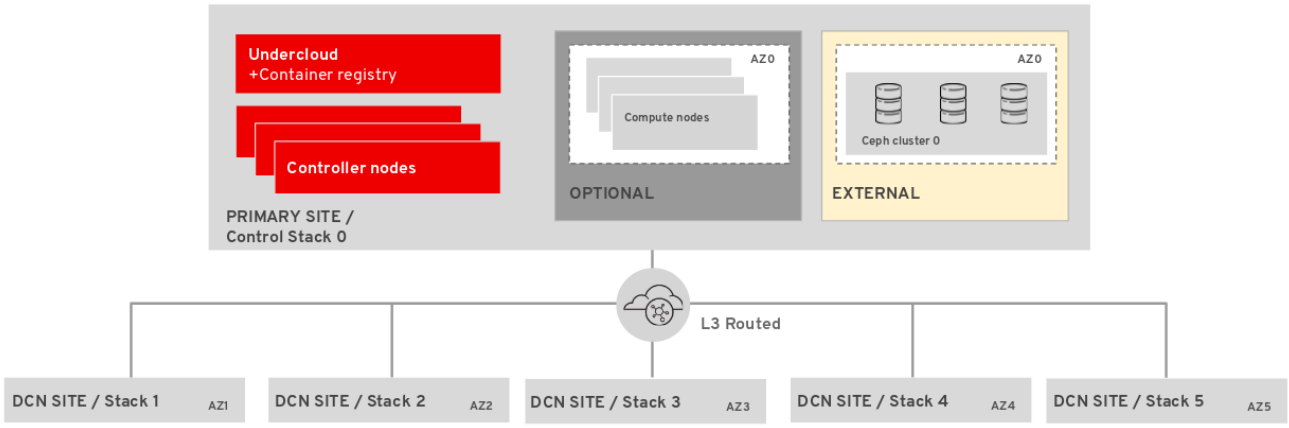
先决条件

- 您必须创建特定于环境的 **network_data.yaml** 和 **vip_data.yaml** 文件。您可以在 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 中找到示例文件。
- 您必须创建一个特定于环境的 **overcloud-baremetal-deploy.yaml** 文件。有关更多信息，请参阅为 overcloud 置备裸机节点。
- Ceph 集群的硬件位于中央位置和每个可用区，或位于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**。

- 一个堆栈位于边缘站点，名为 **dcn0**。
- 类似于 **dcn0** 的其他堆栈，如 **dcn1**、 **dcn2** 等。



流程

您可以安装中央位置，使其与 现有 Red Hat Ceph Storage 集群集成 。将 Red Hat Ceph Storage 与 DCN 部署的中央站点集成没有特殊要求，但您必须在部署 overcloud 前完成特定于 DCN 的步骤：

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

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

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。使用它来分隔为各自站点设计的模板。将中央位置的 **network_data.yaml**、**vip_data.yaml** 和 **overcloud-baremetal-deploy.yaml** 模板移到 **/home/stack/central/**。

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

mv network_data.yaml /home/stack/central
mv vip_data.yaml /home/stack/central
mv overcloud-baremetal-deploy.yaml /home/stack/central
```

5. 调配 overcloud 的网络。此命令取 overcloud 网络的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 为 overcloud 调配虚拟 IP。此命令取虚拟 IP 的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud network vip provision \
--stack central \
--output /home/stack/central/overcloud-vip-deployed.yaml \
/home/stack/central/vip_data.yaml
```

7. 置备裸机实例。该命令为裸机节点使用一个定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
openstack overcloud node provision \
--stack central \
--network-config \
-o /home/stack/central/deployed_metal.yaml \
/home/stack/central/overcloud-baremetal-deploy.yaml
```

8. 在 **site-name.yaml** 环境文件中配置站点的命名约定。Compute (nova)可用区和 Block Storage (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
```

9. 使用类似如下的内容配置 **external-ceph.yaml** 模板：

```
parameter_defaults:
    CinderEnableIscsiBackend: false
    CinderEnableRbdBackend: true
    CinderEnableNfsBackend: false
    NovaEnableRbdBackend: true
    GlanceBackend: rbd
    GlanceBackendID: central
    GlanceEnabledImportMethods: web-download,copy-image
    GlanceStoreDescription: 'central rbd glance store'
    CinderRbdPoolName: "openstack-cinder"
    NovaRbdPoolName: "openstack-nova"
    GlanceRbdPoolName: "openstack-images"
    CinderBackupRbdPoolName: "automation-backups"
    GnocchiRbdPoolName: "automation-metrics"
    CephClusterFSID: 38dd387e-837a-437c-891c-7fc69e17a3c
    CephClusterName: central
    CephExternalMonHost: 10.9.0.1,10.9.0.2,10.9.0.3
    CephClientKey: "AQAKtECeLemfiBBdQp7cjNYQRGW9y8GnhhFZg=="
    CephClientUserName: "openstack
```

10. 部署中央位置：

```
openstack overcloud deploy \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
```

```
-n /home/stack/central/network-data.yaml \
...
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/external-ceph.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/external-ceph.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/central_roles.yaml
```

11. 为中央位置部署 overcloud 后，导出边缘站点的额外堆栈部署的输入数据并将其放置在 **/home/stack/overcloud-deploy** 目录中。确保存在此 control-plane-export.yaml 文件：

```
stat ~/overcloud-deploy/control-plane/control-plane-export.yaml
```

12. 创建名为 **central_ceph_external.yaml** 的环境文件，其中包含有关 Red Hat Ceph Storage 部署的详细信息。此文件可以传递给边缘站点的额外堆栈部署。

```
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]** 部分中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。

其他资源

- 验证外部 Ceph Storage 集群集成

第 6 章 在没有存储的情况下部署边缘

如果您使用对象存储服务(swift)作为中央位置的镜像服务(glance)的后端，您可以在边缘站点部署分布式计算节点(DCN)集群，而无需块存储。如果您在没有块存储的情况下部署站点，则无法稍后更新它使其具有块存储。

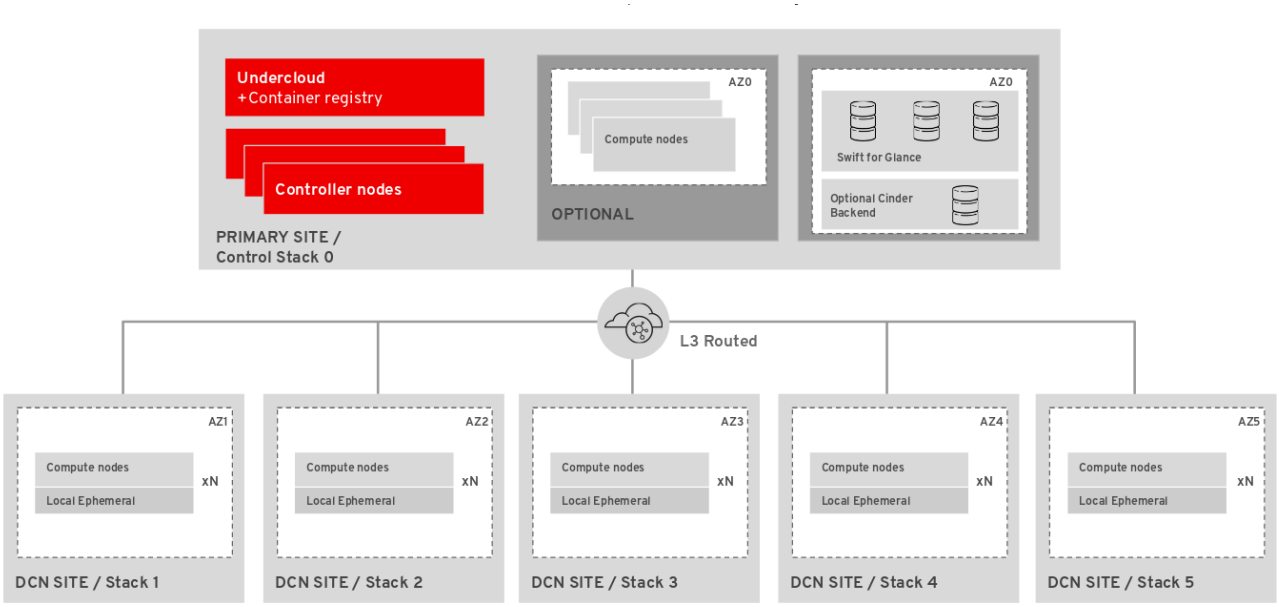
在在没有存储的情况下部署边缘站点时使用 **compute** 角色。

重要

以下流程使用 lvm 作为块存储服务(cinder)的后端，它不支持用于生产环境。您必须部署一个经过认证的块存储解决方案，作为块存储服务的后端。

6.1. 没有存储的 DCN 边缘站点的构架

要部署此架构，请使用 **Compute** 角色。



在边缘没有块存储

- control plane 上的 Object Storage (swift)服务用作镜像(glance)服务后端。
- 多后端镜像服务不可用。
  - 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 11 章 *将缓存 glance 镜像预缓存到 nova*。
- 实例本地存储在 Compute 节点上。
- Block Storage (cinder)等卷服务在边缘站点上不可用。

重要

如果您没有使用 Red Hat Ceph 存储部署中央位置，则不会再选择使用存储部署边缘站点。

有关在边缘部署没有块存储的更多信息，请参阅 第 6.2 节 “在没有存储的情况下部署边缘节点”。

6.2. 在没有存储的情况下部署边缘节点

在边缘站点部署 Compute 节点时，您可以使用中央位置作为 control plane。您可以将新的 DCN 堆栈添加到部署中，并重复使用中央位置的配置文件来创建新的环境文件。

先决条件

- 您必须创建特定于环境的 **network_data.yaml** 文件。您可以在 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 中找到示例文件。
- 您必须创建一个特定于环境的 **overcloud-baremetal-deploy.yaml** 文件。有关更多信息，请参阅为 overcloud 置备裸机节点。
- 您必须将镜像上传到中央位置，然后才能将它们复制到边缘站点；在中央位置，每个镜像的副本都必须存在于镜像服务(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. 以 stack 用户身份登录 undercloud。
2. 查找 stackrc 文件：

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

3. 生成环境文件 ~/dcn0/dcn0-images-env.yaml[d]：

```
sudo[e] openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file ~/dcn0/dcn0-images-env.yaml
```

4. 为边缘位置生成角色文件。使用适合您的环境的角色，为边缘位置生成角色：

```
(undercloud)$ openstack overcloud roles \
 generate Compute \
 -o /home/stack/dcn0/dcn0_roles.yaml
```

5. 如果使用 ML2/OVS 进行网络覆盖，您必须编辑 Compute 角色包括 **NeutronDhcpAgent** 和 **NeutronMetadataAgent** 服务：
   a. 为 Compute 角色创建一个角色文件：

```
openstack overcloud roles \
generate Compute \
-o /home/stack/dcn0/dcn0_roles.yaml
```

   b. 编辑 /home/stack/dcn0/dcn0_roles.yaml 文件，使其包含 **NeutronDhcpAgent** 和 **NeutronMetadataAgent** 服务：

```
...
    - 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，请参阅准备路由供应商网络。

6. 调配 overcloud 的网络。此命令取 overcloud 网络的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/dcn0/overcloud-networks-deployed.yaml \
/home/stack/dcn0/network_data.yaml
```

重要

如果您的 **network_data.yaml** 模板包含您置备网络时不包含的额外网络，则必须在中央位置重新运行网络置备命令：

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

7. 置备裸机实例。该命令为裸机节点使用一个定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud node provision \
--stack dcn0 \
--network-config \
-o /home/stack/dcn0/deployed_metal.yaml \
~/overcloud-baremetal-deploy.yaml
```

8. 在 site-name.yaml 环境文件中配置站点的命名约定。

```
parameter_defaults:
  NovaComputeAvailabilityZone: dcn0
  ControllerExtraConfig:
    nova::availability_zone::default_schedule_zone: dcn0
  NovaCrossAZAttach: false
```

9. 为 dcn0 边缘站点部署堆栈：

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r /home/stack/dcn0/dcn0_roles.yaml \
```

```
-n /home/stack/network_data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml
```

6.3. 在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 存储的边缘站点使用时效率低下。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

不公告 RAW 镜像的计算节点无法托管从 RAW 镜像创建的实例。这会影响 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. 在其中一个自定义环境文件中包含 **NovaImageTypeExcludeList** 参数：

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

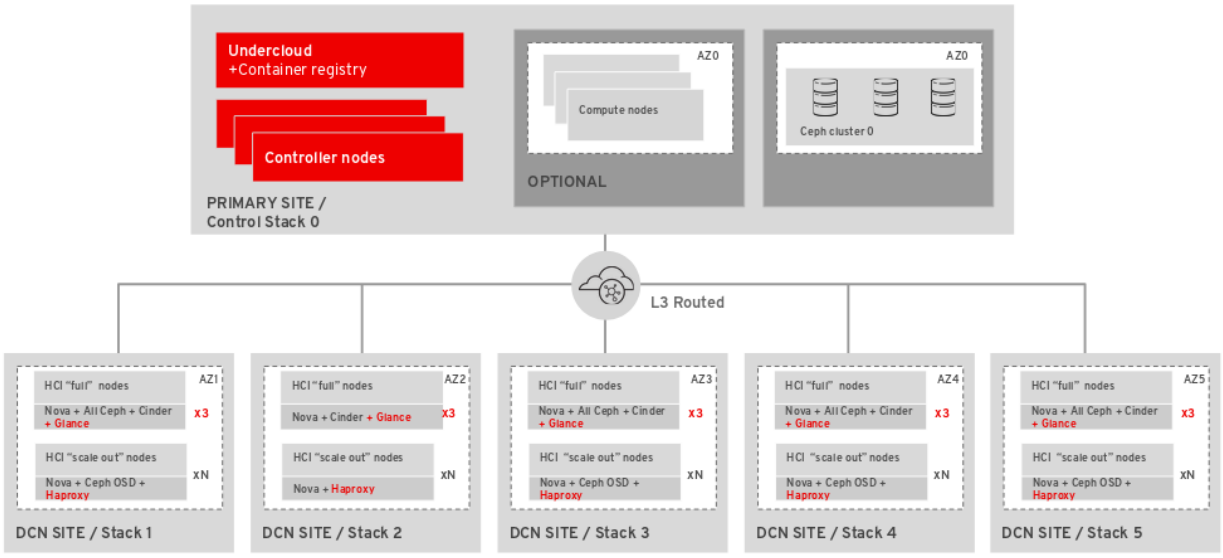
4. 在 overcloud 部署命令中包含 **NovaImageTypeExcludeList** 参数的环境文件，以及与部署相关的任何其他环境文件：

```
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，使其包含边缘的分布式镜像管理和持久性存储，以及使用 Red Hat OpenStack Platform 和 Ceph Storage 的好处。



7.1. 使用存储的边缘部署的角色

以下角色可用于使用存储的边缘部署。根据您选择的配置，为您的环境选择适当的角色。

7.1.1. 没有超融合节点的存储

当您使用存储部署边缘时，且没有部署超融合节点，请使用以下四个角色之一。

DistributedCompute

DistributedCompute 角色用于存储部署中的前三个计算节点。**DistributedCompute** 角色包含 **GlanceApiEdge** 服务，可确保镜像服务在本地边缘站点而不是中央 hub 位置使用。对于任何其他节点，请使用 **DistributedComputeScaleOut** 角色。

DistributedComputeScaleOut

DistributedComputeScaleOut 角色包括 **HAproxyEdge** 服务，它允许 DistributedComputeScaleOut 角色中创建的实例将镜像服务的请求代理到在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** 角色部署所需的最小主机数量。

CephAll

CephAll 角色包括 Ceph OSD、Ceph mon 和 Ceph Mgr 服务。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 角色包括 Ceph OSD 服务。如果三个 CephAll 节点没有提供足够的存储容量，则根据需要添加任意数量的 CephStorage 节点。

7.1.2. 使用超融合节点存储

当您使用存储部署边缘时，并且计划具有组合计算和存储的超融合节点，请使用以下两个角色之一：

DistributedComputeHCI

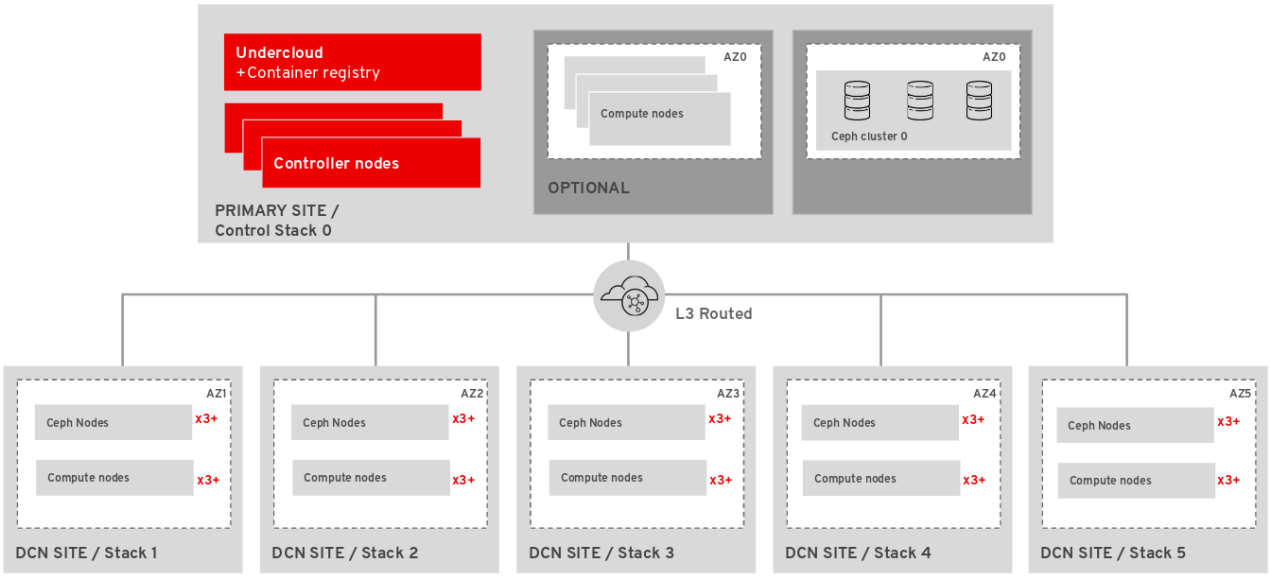
DistributedComputeHCI 角色通过包含 Ceph 管理和 OSD 服务在边缘启用超融合部署。在使用 DistributedComputeHCI 角色时，您必须使用三个节点。

DistributedComputeHCIScaleOut

DistributedComputeHCIScaleOut 角色包括 **Ceph OSD** 服务，允许在添加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融合部署。在使用 **DistributedComputeHCI** 角色时，您必须使用三个节点。

7.2. 带有存储的 DCN 边缘站点的构架

要使用存储部署 DCN，还必须在中央位置部署 Red Hat Ceph Storage。您必须使用 **dcn-storage.yaml** 和 **cephadm.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**, **DistributedComputeScaleOut**, **CephAll**, 和 **CephStorage** 角色。

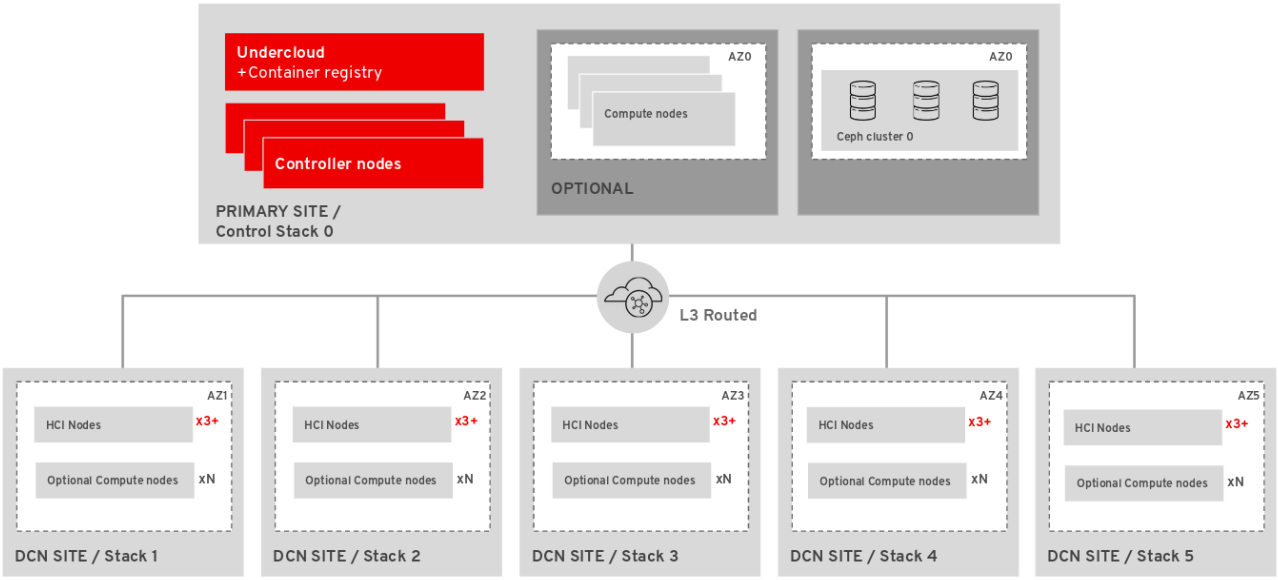


使用边缘的块存储

- Red Hat Ceph 块设备(RBD)用作镜像(glance)服务后端。
- 多后端镜像服务(glance)可用，以便在中央和 DCN 站点之间复制镜像。
- Block Storage (cinder)服务在所有站点上可用，并使用 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 在专用存储节点上单独运行。
- Nova 临时存储由 Ceph (RBD)支持。
  如需更多信息，请参阅 第 5.2 节 “使用存储部署中央站点”。

7.3. 带有超融合存储的 DCN 边缘站点的构架

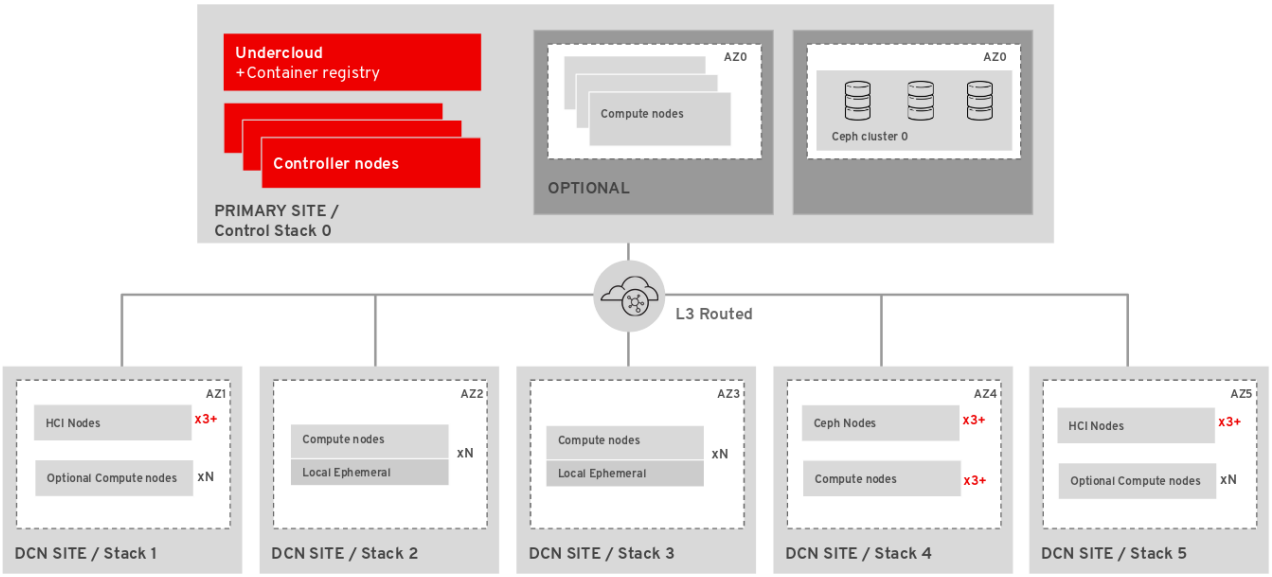
要部署此配置，还必须在中央位置部署 Red Hat Ceph Storage。您需要配置 **dcn-storage.yaml** 和 **cephadm.yaml** 环境文件。使用 **DistributedComputeHCI**，和 **DistributedComputeHCIScaleOut** 角色。您还可以使用 **DistributedComputeScaleOut** 角色添加不参与提供 Red Hat Ceph Storage 服务的 Compute 节点。



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。
- 多后端镜像服务(glance)可用，以便在中央和 DCN 站点之间复制镜像。
- Block Storage (cinder)服务在所有站点上可用，并使用 Red Hat Ceph Block Devices (RBD)驱动程序访问。
- 块存储服务和 Red Hat Ceph Storage 在 Compute 节点上运行。
  更多信息请参阅 第 7.4 节 “使用超融合存储部署边缘站点”。

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，每个站点都有唯一的配置。您必须使用 Red Hat Ceph 存储部署中央位置，以使用存储部署任何边缘站点。



7.4. 使用超融合存储部署边缘站点

部署中央站点后，构建边缘站点，并确保每个边缘位置都连接到其自身存储后端，以及中央位置上的存储后端。此配置中应包含 spine 和 leaf network 配置，增加了 ceph 需要的 storage 和 storage_mgmt 网

络。如需更多信息，请参阅 Spine Leaf Networking。您必须在每个边缘站点的中央位置和存储网络之间具有连接性，以便在站点之间移动镜像服务(glance)镜像。

确保中央位置可以与每个边缘站点的 mons 和 OSD 通信。但是，您应该在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。

先决条件

- 您必须创建特定于环境的 **network_data.yaml** 文件。您可以在 **/usr/share/openstack-tripleo-heat-templates/network-data-samples** 中找到示例文件。
- 您必须创建一个特定于环境的 **overcloud-baremetal-deploy.yaml** 文件。有关更多信息，请参阅为 overcloud 置备裸机节点。
- 您有三个镜像服务(glance)服务器的硬件，位于中央位置和每个可用区，或者在每个需要存储服务的地理位置。在边缘位置，镜像服务被部署到 DistributedComputeHCI 节点。

流程

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

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

3. 生成环境文件 ~/dcn0/dcn0-images-env.yaml ：

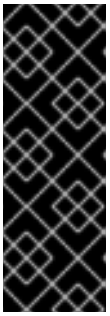
```
sudo openstack tripleo container image prepare \
-e containers.yaml \
--output-env-file /home/stack/dcn0/dcn0-images-env.yaml
```

4. 为 dcn0 边缘位置生成适当的角色：

```
openstack overcloud roles generate DistributedComputeHCI
DistributedComputeHCIScaleOut \
-o ~/dcn0/dcn0_roles.yaml
```

5. 调配 overcloud 的网络。此命令取 overcloud 网络的定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/dcn0/overcloud-networks-deployed.yaml \
/home/stack/network_data.yaml
```



重要

如果您的 **network_data.yaml** 模板包含您置备网络时不包含的额外网络，则必须在中央位置重新运行网络置备命令：

```
(undercloud)$ openstack overcloud network provision \
--output /home/stack/central/overcloud-networks-deployed.yaml \
/home/stack/central/network_data.yaml
```

6. 置备裸机实例。该命令为裸机节点使用一个定义文件作为输入。您必须使用命令中的输出文件来部署 overcloud：

```
(undercloud)$ openstack overcloud node provision \
--stack dcn0 \
--network-config \
-o /home/stack/dcn0/deployed_metal.yaml \
/home/stack/overcloud-baremetal-deploy.yaml
```

7. 如果要使用超融合存储部署边缘站点，您必须创建一个带有以下参数的 **initial-ceph.conf** 配置文件。如需更多信息，请参阅为 HCI 配置 Red Hat Ceph Storage 集群 ：

```
[osd]
osd_memory_target_autotune = true
osd_numa_auto_affinity = true
[mgr]
mgr/cephadm/autotune_memory_target_ratio = 0.2
```

8. 使用 **deployed_metal.yaml** 文件作为 **openstack overcloud ceph deploy** 命令的输入。**openstack overcloud ceph deploy 命令输出** 描述部署的 Ceph 集群的 yaml 文件：

```
openstack overcloud ceph deploy \
/home/stack/dcn0/deployed_metal.yaml \
--stack dcn0 \
--config ~/dcn0/initial-ceph.conf \ 1
--output ~/dcn0/deployed_ceph.yaml \
--container-image-prepare ~/containers.yaml \
--network-data ~/network-data.yaml \
--cluster dcn0 \
--roles-data dcn_roles.yaml
```

1 仅在部署超融合基础架构时包括 initial-ceph.conf。

9. 在 site-name.yaml 环境文件中配置站点的命名约定。Nova 可用区和 Cinder 存储可用域必须匹配：

```
parameter_defaults:
    NovaComputeAvailabilityZone: dcn0
    ControllerExtraConfig:
        nova::availability_zone::default_schedule_zone: dcn0
    NovaCrossAZAttach: false
    CinderStorageAvailabilityZone: dcn0
    CinderVolumeCluster: dcn0
    GlanceBackendID: dcn0
```

10. 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: 'dcn0 rbd glance store'
    GlanceBackendID: dcn0
    GlanceMultistoreConfig:
      central:
```

```
    GlanceBackend: rbd
    GlanceStoreDescription: 'central rbd glance store'
    CephClusterName: central
```

11. 为 dcn0 位置部署堆栈：[d]

```
openstack overcloud deploy \
--deployed-server \
--stack dcn0 \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/dcn0/dcn0_roles.yaml \
-n ~/dcn0/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm-rbd-only.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/overcloud-deploy/central/central-export.yaml \
-e /home/stack/dcn0/deployed_ceph.yaml \
-e /home/stack/dcn-common/central_ceph_external.yaml \
-e /home/stack/dcn0/overcloud-vip-deployed.yaml \
-e /home/stack/dcn0/deployed_metal.yaml \
-e /home/stack/dcn0/overcloud-networks-deployed.yaml \
-e ~/control-plane/glance.yaml
```

7.5. 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，该集群位于 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 **backups** 和 **metrics** 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 Ceph Placement Groups (PG) per Pool Calculator 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，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，将节点配置为连接到非受管 Ceph 集群。使用可识别的命名约定，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. 为 CephClusterFSID 和 CephClientKey 参数使用之前保存的值。

   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. 您必须在边缘站点内为 **CephClusterName** 参数选择唯一值。重复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 在中央位置部署了 Red Hat Ceph 存储，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，因此信息特定于前面步骤中部署的 Ceph 集群：

```
sudo -E openstack overcloud export ceph \
--stack central \
--output-file /home/stack/dcn-common/central_ceph_external.yaml
```

如果中央位置在外部部署了 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，其中包含用于中央位置的托管 Red Hat Ceph Storage 集群的每个站点类似的详细信息。**openstack overcloud export ceph** 命令不适用于带有非受管 Red Hat Ceph Storage 集群的站点。更新中央位置时，此文件将允许在边缘站点中的存储集群作为次要位置进行中央位置

```
parameter_defaults:
  CephExternalMultiConfig:
cluster: dcn1
…
cluster: dcn2
…
```

7. 在部署 overcloud 时，使用 external-ceph.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/external-ceph.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 /home/stack/dnc1/ceph-external-dcn1.yaml  \
  ....
  -e /home/stack/overcloud-deploy/central/central-export.yaml \
  -e /home/stack/dcn-common/central_ceph_external.yaml \
  -e /home/stack/dcn1/dcn_ceph_keys.yaml \
  -e /home/stack/dcn1/role-counts.yaml \
  -e /home/stack/dcn1/ceph.yaml \
  -e /home/stack/dcn1/site-name.yaml \
  -e /home/stack/dcn1/tuning.yaml \
  -e /home/stack/dcn1/glance.yaml
```

8. 在部署所有边缘位置后重新部署中央位置。

7.6. 更新中央位置

使用示例流程配置和部署所有边缘站点后，在中央位置更新配置，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 Image 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，则该镜像副本中断，您必须重启它。如需更多信息，请参阅 中断镜像服务进程后清除重新发送数据。

流程

1. 创建一个类似如下的 **~/central/glance_update.yaml** 文件：本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 central 站点的 glance 服务配置为边缘站点的 Ceph 集群的客户端 **dcn0** 和 **dcn1**。

```
openstack overcloud export ceph \
--stack dcn0,dcn1 \
--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。

注意

如果在创建中央堆栈时最初不提供其叶网络，请在 **overcloud 部署** 命令中包含 **deployed_metal.yaml**。

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
```

```
-r ~/central/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml
```

4. 在位于中央位置的控制器上，重新启动 **cinder-volume** 服务。如果您使用 **cinder-backup** 服务部署了中央位置，则也重启 **cinder-backup** 服务：

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

7.6.1. 在中断的镜像服务进程后清除数据

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 节点上的数据。将用于暂存存储的 **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，则必须从 staging 中删除数据并重启复制导入。
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。从导入作业中清除处于导入状态的镜像中断：

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

7.7. 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 Dashboard 添加到 overcloud 部署中。部署中央位置之前，应完成这些步骤。

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

- 您必须部署自己的解决方案以进行负载平衡，以创建高可用性虚拟 IP。边缘站点没有部署 haproxy 或 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。控制面板安装在每个具有不同 IP 地址的 DistributedComputeHCI 节点上，而无需负载平衡解决方案。

您可以创建额外网络来托管可以公开 Ceph 仪表板的虚拟 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。创建的网络的名称为 **StorageDashboard**。

流程

1. 以 **stack** 身份登录 Red Hat OpenStack Platform Director。
2. 生成 **DistributedComputeHCIDashboard** 角色以及适合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/cephadm/cephadm-rbd-only.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/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。

您可以通过类似步骤确认仪表板在边缘位置运行，但存在例外情况，因为边缘位置没有负载均衡器。

1. 检索特定于所选堆栈的仪表板管理员登录凭证：

```
grep grafana_admin /home/stack/config-download/<stack>/cephadm/cephadm-extra-vars-heat.yml
```

2. 在特定于所选堆栈的清单中，**/home/stack/config-download/<stack>/cephadm/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. 如果可以访问这些 IP 地址，您可以检查 Ceph 控制面板是否处于活动状态。这些 IP 地址位于存储网络中，且没有路由。如果这些 IP 地址不可用，您必须为从清单获取的三个 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 在边缘负载均衡网络流量

您可以使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)在边缘站点中创建负载均衡器来提高流量吞吐量并缩短延迟。

本节中包含的主题有：

- 为负载均衡服务可用区创建网络资源
- 创建负载均衡服务可用区
- 在可用区中创建负载均衡器

8.1. 为负载均衡服务可用区创建网络资源

在创建 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)可用区(AZ)之前，您必须是一个 RHOSP 管理员并运行 Ansible playbook **octavia-dcn-deployment.yaml**。

通过运行 **octavia-dcn-deployment.yaml**，您可以创建网络、子网和路由器等网络资源，它们是负载均衡服务 AZ。您可以为 playbook 提供配置输入文件 **octavia-dcn-parameters.yaml**，在其中指定了 AZ 名称和每个 AZ 使用的管理网络。

运行 playbook 并创建了必要的网络资源后，您必须创建实际的 RHOSP 负载均衡服务 AZ，然后才能在 AZs 中创建适合其分布式计算节点(DCN)区域的负载均衡器。

此流程演示了为 3 负载均衡服务 AZ 创建所需的网络资源，名为 **az-central**、**az-dcn1** 和 **az-dcn2**。这些负载平衡服务 AZ 名称与计算服务 AZ 的名称匹配，也是此部署中使用的 3 DCN 的名称。

先决条件

- 对于您要创建的每个负载均衡服务 AZ，您必须有一个 Compute 服务(nova) AZ。
- 对于您要创建的每个负载均衡服务 AZ，还必须有一个网络服务(neutron) AZ。这些网络服务 AZ 必须与计算服务 AZ 的名称匹配。
- 您的负载均衡服务供应商驱动程序必须是 amphora。OVN 供应商驱动程序不支持 AZ。
- 您必须是一个具有 **admin** 角色的 RHOSP 用户。

流程

1. 提供您的凭据文件。

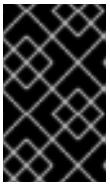
   示例

   ```
   $ source ~/centralrc
   ```

2. 创建一个文件 **octavia-dcn-parameters.yaml**，并使用以下语法，添加负载平衡服务 AZ 及其管理网络，您希望 Ansible playbook 创建所需的网络资源。
   octavia_controller_AZ_name 的值是所有 Load-balancer 服务运行的 AZ 的名称：

   ```
   octavia_controller_availability_zone: <octavia_controller_AZ_name>
   octavia_availability_zones:
     <octavia_controller_AZ_name>: # no cidr needed, it uses the already existing subnet
     <octavia_AZ_n>:
   ```

```
    lb_mgmt_subnet_cidr: <CIDR_address_n>
  <octavia_AZ_n2>:
    lb_mgmt_subnet_cidr: <CIDR_address_n2>
```



重要

您指定的负载均衡服务 AZ 的名称必须与预先存在的 Compute 服务 AZ 的名称匹配。您可以通过运行 **openstack availability zone list --compute** 来获取计算服务 AZ 的名称。

Ansible playbook 为每个 AZ 创建一个网络、子网和路由器，并使用您在 **octavia-dcn-parameters.yaml** 中指定的 AZ 名称： **lb-mgmt-<AZ_name>-net**、**lb-mgmt-<AZ_name>-**subnet、**lb-mgmt-<AZ_name>-subnet** 和 **lb-mgmt-<AZ_name>-router**。例外是 **octavia_controller_AZ_name** 的网络资源：该 playbook 使用现有的负载平衡管理网络和子网、**lb-mgmt-net** 和 **lb-mgmt-subnet**，并分别创建一个关联的路由器，其名称为 **lb-mgmt-router**。

在本例中，指定了 3 AZ： **az-central**、**az-dcn1** 和 **az-dcn2**。**az-central** AZ 使用现有的负载平衡管理网络 **lb-mgmt-net**。其他两个 AZ 分别使用 **172.47.0.0/16** 和 **172.48.0.0/16** ：

示例

```
octavia_controller_availability_zone: az-central
octavia_availability_zones:
  az-central: # no cidr needed; it uses the existing subnet
  az-dcn1:
    lb_mgmt_subnet_cidr: 172.47.0.0/16
  az-dcn2:
    lb_mgmt_subnet_cidr: 172.48.0.0/16
```

3. 运行 Ansible playbook **octavia-dcn-deployment.yaml**，并包含您在 **octavia-dcn-parameters.yaml** 中创建的 AZ 定义：

示例

```
$ ansible-playbook -i overcloud-deploy/central/config-download/\
central/tripleo-ansible-inventory.yaml \
/usr/share/ansible/tripleo-playbooks/octavia-dcn-deployment.yaml \
-e @octavia-dcn-parameters.yaml -e stack=central -v
```

验证

1. 确认存在所需的 **lb-mgmtmgmt** 子网。

```
$ openstack subnet list -c Name -c Subnet
```

输出示例

```
+------------------------+-----------------+
| Name                   | Subnet          |
+------------------------+-----------------+
| lb-mgmt-az-dcn2-subnet | 172.48.0.0/16   |
| segment5               | 10.0.20.0/24    |
```

```
| segment3                | 10.101.30.0/24   |
| segment2                | 10.101.20.0/24   |
| lb-mgmt-az-dcn1-subnet  | 172.47.0.0/16    |
| heat_tempestconf_subnet | 192.168.199.0/24 |
| segment4                | 10.0.10.0/24     |
| lb-mgmt-subnet          | 172.24.0.0/16    |
| segment1                | 10.101.10.0/24   |
| lb-mgmt-backbone-subnet | 172.49.0.0/16    |
| segment6                | 10.0.30.0/24     |
+-------------------------+------------------+
```

2. 确认存在所需的虚拟路由器。

```
$ openstack router list -c Name -c Status
```

输出示例

```
+-----------------------+--------+
| Name                  | Status |
+-----------------------+--------+
| lb-mgmt-az-dcn2-router | ACTIVE |
| lb-mgmt-az-dcn1-router | ACTIVE |
| lb-mgmt-router         | ACTIVE |
+-----------------------+--------+
```

后续步骤

- 为负载均衡服务创建可用区

8.2. 为负载均衡服务创建可用区

使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，RHOSP 管理员可以创建可用区(AZ)，使项目用户在分布式计算节点(DCN)环境中创建负载均衡器以增加流量吞吐量并降低延迟。

创建负载平衡服务 AZ 需要两个步骤：RHOSP 管理员必须首先创建一个 AZ 配置集，然后使用配置集创建用户可见的负载平衡服务 AZ。

AZ 配置集必须具有以下内容：

- Compute 服务(nova) AZ 的名称。
- 要使用的管理网络。
  每个 AZ 都有多个管理网络，一个唯一的网络。中央 AZ 使用现有的负载平衡管理网络 **lb-mgmt-net**，另外 AZ 使用其相应的网络，**lb-mgmt-<AZ_name>-net**，例如 **lb-mgmt-az-dcn1-net**、**lb-mgmt-az-dcn2-net** 等等。

先决条件

- 您必须有一个 DCN 环境，它已通过运行 **octavia-dcn-deployment.yaml** Ansible playbook 创建所需的网络资源。
  如需更多信息，请参阅为负载均衡服务可用区创建网络资源。
- 您的负载均衡服务供应商驱动程序必须是 amphora。OVN 供应商驱动程序不支持 AZ。

- 您必须是一个具有 **admin** 角色的 RHOSP 用户。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/centralrc
   ```

2. 收集要用于命名负载平衡服务 AZ 的计算服务 AZ 的名称。

   重要

   您创建的负载均衡服务 AZ 的名称必须与计算服务 AZ 的名称匹配。

   ```
   $ openstack availability zone list --compute
   ```

   输出示例

   ```
   +------------+-------------+
   | Zone Name  | Zone Status |
   +------------+-------------+
   | az-central | available   |
   | az-dcn1    | available   |
   | az-dcn2    | available   |
   | internal   | available   |
   +------------+-------------+
   ```

3. 收集用于创建负载平衡服务 AZ 的管理网络的 ID：

   ```
   $ openstack network list -c Name -c ID
   ```

   输出示例

   ```
   +--------------------------------------+--------------------------+
   | ID                                   | Name                     |
   +--------------------------------------+--------------------------+
   | 10458d6b-e7c9-436f-92d9-711677c9d9fd | lb-mgmt-az-dcn2-net      |
   | 662a94f5-51eb-4a4c-86c4-52dcbf471ef9 | lb-mgmt-net              |
   | 6b97ef58-2a25-4ea5-931f-b7c07cd09474 | lb-mgmt-backbone-net     |
   | 99f4215b-fad8-432d-8444-1f894154dc30 | heat_tempestconf_network |
   | a2884aaf-846c-4936-9982-3083f6a71d9b | lb-mgmt-az-dcn1-net      |
   | d7f7de6c-0e84-49e2-9042-697fa85d2532 | public                   |
   | e887a9f9-15f7-4854-a797-033cedbfe5f3 | public2                  |
   +--------------------------------------+--------------------------+
   ```

4. 创建 AZ 配置集。重复此步骤，为您要创建的每个负载均衡服务 AZ 创建 AZ 配置集：

   ```
   $ openstack loadbalancer availabilityzoneprofile create \
   --name <AZ_profile_name> --provider amphora --availability-zone-data '{"compute_zone": "
   <compute_AZ_name>","management_network": "<lb_mgmt_AZ_net_UUID>"}'
   ```

示例 - 为 az-central创建配置集

在本例中，创建了一个 AZ 配置集(**az_profile_central**)，它使用管理网络(**lb-mgmt-net**)在 Compute AZ (**az-central**)中运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az_profile_central --provider amphora --availability-zone-data \
'{"compute_zone": "az-central","management_network": \
"662a94f5-51eb-4a4c-86c4-52dcbf471ef9"}'
```

5. 重复步骤 4，为您要创建的每个负载均衡服务 AZ 创建 AZ 配置集。

示例 - 为 az-dcn1创建配置文件

在本例中，会创建一个 AZ 配置集(**az-profile-dcn1**)，它使用管理网络(**lb-mgmt-az-dcn1-net**)在计算 AZ (**az-dcn1**)上运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn1 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn1","management-network": \
"a2884aaf-846c-4936-9982-3083f6a71d9b"}'
```

示例 - 为 az-dcn2创建配置文件

在本例中，会创建一个 AZ 配置集(**az-profile-dcn2**)，它使用管理网络(**lb-mgmt-az-dcn2-net**)在计算 AZ (**az-dcn2**)中运行的 Compute 节点上：

```
$ openstack loadbalancer availabilityzoneprofile create \
--name az-profile-dcn2 --provider amphora --availability-zone-data \
'{"compute-zone": "az-dcn2","management-network": \
"10458d6b-e7c9-436f-92d9-711677c9d9fd"}'
```

6. 使用 AZ 配置集，创建一个负载均衡服务 AZ。为每个 AZ 使用适当的配置集，为任何其他 AZ 重复此步骤。

示例 - 创建 AZ: az-central

在本例中，使用 AZ 配置集(**az-profile-central**)创建负载平衡服务 AZ (**az-central**)：

```
$ openstack loadbalancer availabilityzone create --name az-central \
--availabilityzoneprofile az-profile-central \
--description "AZ for Headquarters" --enable
```

示例 - 创建 AZ: az-dcn1

在本例中，使用 AZ 配置集(az-profile-**az-dcn1**)创建负载平衡服务 AZ (**az-dcn1**)：

```
$ openstack loadbalancer availabilityzone create --name az-dcn1 \
--availabilityzoneprofile az-profile-az-dcn1 \
--description "AZ for South Region" --enable
```

示例 - 创建 AZ: az-dcn2

在本例中，负载均衡服务 AZ (**az-dcn2**)使用 AZ 配置集(**az-profile-az-dcn2**)创建：

```
$ openstack loadbalancer availabilityzone create --name az-dcn2 \
--availabilityzoneprofile az-profile-az-dcn2 \
--description "AZ for North Region" --enable
```

验证

- 确认 AZ (**az-central**)已创建。对任何其他 AZ 重复此步骤，为每个 AZ 使用适当的名称。

示例 - 验证 az-central

```
$ openstack loadbalancer availabilityzone show az-central
```

输出示例

```
+-----------------------------+--------------------------------------+
| Field                       | Value                                |
+-----------------------------+--------------------------------------+
| name                        | az-central                           |
| availability_zone_profile_id | 5ed25d22-52a5-48ad-85ec-255910791623 |
| enabled                     | True                                 |
| description                 | AZ for Headquarters                  |
+-----------------------------+--------------------------------------+
```

示例 - 验证 az-dcn1

```
$ openstack loadbalancer availabilityzone show az-dcn1
```

输出示例

```
+-----------------------------+--------------------------------------+
| Field                       | Value                                |
+-----------------------------+--------------------------------------+
| name                        | az-dcn1                              |
| availability_zone_profile_id | e0995a82-8e67-4cea-b32c-256cd61f9cf3 |
| enabled                     | True                                 |
| description                 | AZ for South Region                  |
+-----------------------------+--------------------------------------+
```

示例 - 验证 az-dcn2

```
$ openstack loadbalancer availabilityzone show az-dcn2
```

输出示例

```
+-----------------------------+--------------------------------------+
| Field                       | Value                                |
+-----------------------------+--------------------------------------+
| name                        | az-dcn2                              |
| availability_zone_profile_id | 306a4725-7dac-4046-8f16-f2e668ee5a8d |
| enabled                     | True                                 |
| description                 | AZ for North Region                  |
+-----------------------------+--------------------------------------+
```

-

后续步骤

- 在可用区中创建负载均衡器

其他资源

- *命令行界面参考中的*LoadBalancer availabilityzoneprofile create
- *命令行界面参考中的*LoadBalancer availabilityzone create

8.3. 在可用区中创建负载均衡器

使用 Red Hat OpenStack Platform (RHOSP)负载均衡服务(octavia)，您可以在分布式计算节点(DCN)环境中在可用区(AZ)中创建负载均衡器，以增加流量吞吐量并降低延迟。

先决条件

- 您必须具有 RHOSP 管理员提供的负载均衡服务 AZ。
- 与负载均衡器关联的虚拟 IP (VIP)网络必须在负载均衡器所属的 AZ 中提供。

流程

1. 提供您的凭据文件。

   示例

   ```
   $ source ~/centralrc
   ```

2. 要为 DCN 环境创建负载均衡器，请使用 **loadbalancer create** 命令和 **--availability-zone** 选项并指定适当的 AZ。

   示例

   例如，要在可用区(**az-central**)上在公共子网(**public_subnet**)上创建非终止的 HTTPS 负载均衡器(**lb1**)，您需要输入以下命令：

   ```
   $ openstack loadbalancer create --name lb1 --vip-subnet-id \
   public_subnet --availability-zone az-central
   ```

3. 通过添加监听程序、池、运行状况监视器和负载均衡器成员继续创建负载均衡器。如需更多信息，请参阅将 *负载均衡配置为服务* 指南。

验证

- 确认负载平衡器(lb1)是可用性区域的成员(**az-central**)。

  示例

  ```
  $ openstack loadbalancer show lb1
  ```

  输出示例

```
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| admin_state_up      | True                                 |
| availability_zone   | az-central                           |
| created_at          | 2023-07-12T16:35:05                  |
| description         |                                      |
| flavor_id           | None                                 |
| id                  | 85c7e567-a0a7-4fcb-af89-a0bbc9abe3aa |
| listeners           |                                      |
| name                | lb1                                  |
| operating_status    | ONLINE                               |
| pools               |                                      |
| project_id          | d303d3bda9b34d73926dc46f4d0cb4bc     |
| provider            | amphora                              |
| provisioning_status | ACTIVE                               |
| updated_at          | 2023-07-12T16:36:45                  |
| vip_address         | 10.101.10.229                        |
| vip_network_id      | d7f7de6c-0e84-49e2-9042-697fa85d2532 |
| vip_port_id         | 7f916764-d171-4317-9c86-a1750a54b16e |
| vip_qos_policy_id   | None                                 |
| vip_subnet_id       | a421cbcf-c5db-4323-b7ab-1df20ee6acab |
| tags                |                                      |
+---------------------+--------------------------------------+
```

其他资源

- 为负载均衡服务创建可用区
- *命令行界面参考*中的 LoadBalancer
- 将 *负载均衡配置为服务* 指南

第 9 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，请从您要替换的节点中删除服务，缩减节点数量，然后按照扩展这些节点的步骤进行备份。

9.1. 删除 RED HAT CEPH STORAGE 服务

从集群中删除 HCI （超融合）节点前，您必须删除 Red Hat Ceph Storage 服务。要删除 Red Hat Ceph 服务，您必须从您要删除的节点上的集群服务禁用和移除 **ceph-osd** 服务，然后停止并禁用 **mon**、**mgr** 和 **osd** 服务。

流程

1. 在 undercloud 上，使用 SSH 连接到您要删除的 DistributedComputeHCI 节点：

```
$ ssh tripleo-admin@<dcn-computehci-node>
```

2. 启动 cephadm shell。对于正在移除主机的站点，使用配置文件和密钥环文件：

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring
```

3. 记录与您要删除的 DistributedComputeHCI 节点关联的 OSD （对象存储设备），以便在后续步骤中使用：

```
[ceph: root@dcn2-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
…
```

4. 使用 SSH 连接到同一集群中的另一节点，并从集群中删除该监控器：

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring

[ceph: root@dcn-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
```

5. 使用 SSH 再次登录到您要从集群中删除的节点。

6. 停止并禁用 **mgr** 服务：

```
[tripleo-admin@dcn2-computehci2-1 ~]$ sudo 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

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl stop ceph-mgr@dcn2-computehci2-
1

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl --type=service | grep ceph
ceph-crash@dcn2-computehci2-1.service  loaded active running Ceph crash dump collector

[tripleo-admin@dcn2-computehci2-1 ~]$ sudo systemctl disable ceph-mgr@dcn2-
computehci2-1
Removed /etc/systemd/system/multi-user.target.wants/ceph-mgr@dcn2-computehci2-
1.service.
```

7. 启动 cephadm shell：

```
$ sudo cephadm shell --config /etc/ceph/dcn2.conf \
--keyring /etc/ceph/dcn2.client.admin.keyring
```

8. 验证节点的 **mgr** 服务是否已从集群中移除：

```
[ceph: root@dcn2-computehci2-1 ~]# ceph -s

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 服务成功删除时，从中删除 mgr 服务的节点不再被列出。

9. 导出 Red Hat Ceph Storage 规格：

```
[ceph: root@dcn2-computehci2-1 ~]# ceph orch ls --export > spec.yml
```

10. 编辑 **spec.yaml** 文件中的规格：

- 从 spec.yml 中删除主机 <dcn-computehci-node> 的所有实例
- 从以下内容中删除 <dcn-computehci-node> 条目的所有实例：
  - service_type: osd
  - service_type: mon

- service_type: host

11. 重新应用 Red Hat Ceph Storage 规格：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch apply -i spec.yml
```

12. 移除使用 **ceph osd tree** 来标识的 OSD：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm --zap 1 7 11 15 18
Scheduled OSD(s) for removal
```

13. 验证正在移除的 OSD 的状态。在以下命令返回没有输出前不要继续：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch osd rm status
OSD_ID  HOST                STATE     PG_COUNT  REPLACE  FORCE
DRAIN_STARTED_AT
1       dcn2-computehci2-1      draining  27       False    False  2021-04-23 21:35:51.215361
7       dcn2-computehci2-1      draining  8        False    False  2021-04-23 21:35:49.111500
11      dcn2-computehci2-1      draining  14       False    False  2021-04-23 21:35:50.243762
```

14. 验证您要删除的主机上没有保留守护进程：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch ps dcn2-computehci2-1
```

如果守护进程仍然存在，您可以使用以下命令删除它们：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host drain dcn2-computehci2-1
```

15. 从 Red Hat Ceph Storage 集群中删除 <dcn-computehci-node> 主机：

```
[ceph: root@dcn2-computehci2-1 /]# ceph orch host rm dcn2-computehci2-1
Removed host 'dcn2-computehci2-1'
```

9.2. 删除镜像服务(GLANCE)服务

从服务中删除镜像时，从节点移除镜像服务。

流程

- 要禁用镜像服务，请在您要删除的节点上使用 **systemctl** 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.
```

9.3. 删除 BLOCK STORAGE (CINDER)服务

从服务中删除 DistributedComputeHCI 节点时，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 tripleo-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
```

9.4. 删除 DISTRIBUTEDCOMPUTEHCI 节点

将 **provisioned** 参数设置为 **false**，并将节点从堆栈中删除。禁用 **nova-compute** 服务并删除相关的网络代理。

流程

1. 复制 **baremetal-deployment.yaml** 文件：

```
cp /home/stack/dcn2/overcloud-baremetal-deploy.yaml \
/home/stack/dcn2/baremetal-deployment-scaledown.yaml
```

2. 编辑 **baremetal-deployement-scaledown.yaml** 文件。识别您要删除的主机，并将 **provisioned** 参数设置为 **false** ：

```
instances:
...
  - hostname: dcn2-computehci2-1
    provisioned: false
```

3. 从堆栈中删除节点：

```
openstack overcloud node delete --stack dcn2 --baremetal-deployment
/home/stack/dcn2/baremetal_deployment_scaledown.yaml
```

4. 可选：如果要重复使用节点，请使用 ironic 清理磁盘。如果节点托管 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
```

5. 重新部署中央站点。包括用于初始配置的所有模板：

```
openstack overcloud deploy \
--deployed-server \
--stack central \
--templates /usr/share/openstack-tripleo-heat-templates/ \
-r ~/control-plane/central_roles.yaml \
-n ~/network-data.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/dcn-storage.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/cephadm/cephadm.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/nova-az-config.yaml \
-e /home/stack/central/overcloud-networks-deployed.yaml \
-e /home/stack/central/overcloud-vip-deployed.yaml \
-e /home/stack/central/deployed_metal.yaml \
-e /home/stack/central/deployed_ceph.yaml \
-e /home/stack/central/dcn_ceph.yaml \
-e /home/stack/central/glance_update.yaml
```

9.5. 替换已删除的 DISTRIBUTEDCOMPUTEHCI 节点

9.5.1. 替换已删除的 DistributedComputeHCI 节点

要将新的 HCI 节点添加到 DCN 部署中，您必须使用额外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添加了特定于边缘站点的配置。

先决条件

节点数在您要替换节点或添加新节点的堆栈的 nodes_data.yaml 文件中是正确的。

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，用于中央位置。

5. 在中心执行堆栈更新：

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

9.6. 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. 验证 ceph osds with 'ceph osd tree' 的状态。确保新节点的所有 osds 都为 STATUS up：

```
[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** 服务在 Status 'enabled' 和 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**，则该服务尚未在该节点上启动。

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
```

9.7. DISTRIBUTEDCOMPUTEHCI 状态故障排除

如果替换节点没有将 EtcdInitialClusterState 参数值设置为 **现有**，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。检查日志显示 **etcd** 服务是否在 **/var/log/containers/stdouts/etcd.log** 日志文件中报告集群 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 **部署模板中的现有** 值，然后重新运行部署脚本。

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 10 章 使用密钥管理器部署

如果您已在 Red Hat OpenStack Platform 16.1.2 发布之前部署了边缘站点，您需要重新生成 roles.yaml 来实现此功能：要实施该功能，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
```

10.1. 使用密钥管理器部署边缘站点

如果要在边缘站点中包含对 Key Manager (barbican)服务的访问，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。

- 您可以通过包含 **/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 11 章 将缓存 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，将缓存 glance 镜像来快速部署实例。如果实例所需的镜像尚未缓存，则在创建实例时，它将下载到 Compute 节点的本地磁盘。

下载 glance 镜像的过程需要一些时间，具体取决于镜像大小和网络特征，如带宽和延迟。

如果您试图启动实例，且镜像在本地的 Ceph 集群上不可用，则启动实例将失败，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** 来控制这个值，您可以在 **dcn-storage.yaml** 文件中找到。

如果在部署 overcloud 之前将 **NovaDisableImageDownloadToRbd** 的值设置为 **false**，则会出现以下内容：

- 如果计算服务(nova)在本地不可用，则会自动流传输 **中央位置** 可用的镜像。
- 您不会在 glance 镜像中使用 COW 副本。
- Compute (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。
- 您可以同时向 **中央位置** 和 nova 存储池饱和。

红帽建议将这个值设置为 true，并确保在启动实例前在本地提供所需的镜像。有关将镜像可用于边缘的更多信息，请参阅 第 A.1.3 节 “将镜像复制到新站点”。

对于本地镜像，您可以使用 **tripleo_nova_image_cache.yml** ansible playbook 来预缓存通常用于部署的镜像或镜像，从而加快创建虚拟机。

11.1. 运行 TRIPLEO_NOVA_IMAGE_CACHE.YML ANSIBLE PLAYBOOK

先决条件

- shell 环境中正确的 API 的身份验证凭据。

在每个步骤中提供的命令之前，您必须确保提供正确的身份验证文件。

流程

1. 为您的 overcloud 堆栈创建一个 ansible 清单目录：

```
$ mkdir inventories

$ find ~/overcloud-deploy/*/config-download \
```

```
-name tripleo-ansible-inventory.yaml |\
while read f; do cp $f inventories/$(basename $(dirname $f)).yaml; done
```

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 inventories \
--extra-vars "@nova_cache_args.yml" \
/usr/share/ansible/tripleo-playbooks/tripleo_nova_image_cache.yml
```

11.2. 性能考虑

您可以指定您要与 ansible **forks** 参数同时下载的镜像数量，其默认值为 **5**。您可以通过增加 **forks** 参数的值来缩短分发此镜像的时间，但您必须随着网络和 glance-api 负载的增加而平衡。

使用 **--forks** 参数调整并发，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
```

11.3. 优化到 DCN 站点的镜像分发

您可以通过将代理用于 glance 镜像分发来降低 WAN 流量。配置代理时：

- Glance 镜像下载到充当代理的单个 Compute 节点。
- 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 比具有高延迟的网络效率低下，如 DCN 站点之间的 WAN。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
```

11.4. 配置 NOVA-CACHE 清理

当出现以下任一条件时，后台进程定期从 nova 缓存中删除镜像：

- 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 参数来控制这个值，或者在云的堆栈更新期间使用：

```
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，但镜像的年龄将重置为 0。运行 ansible play 比 **NovaImageCacheTTL** 参数的值更频繁，以维护镜像的缓存。

第 12 章 用于 DCN 的 TLS-E

您可以在专为分布式计算节点基础架构设计的云上启用 TLS （传输层安全）。您可以选择只启用 TLS 进行公共访问，或使用 TLS-e 在每个网络上启用 TLS，允许对所有内部和外部数据流进行加密。

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。有关用于公共访问的 TLS 的更多信息，请参阅在 Overcloud 公共端点上启用 SSL/TLS。

12.1. 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上配置 TLS-e 时，请基于为 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，允许可信子网到 **ipa-ext.conf** 文件中的 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)，您必须升级 Red Hat Enterprise Linux，然后按照 RHEL 8.4 的指示进行操作。

Red Hat Enterprise Linux 7.x

如果您在 Red Hat Enterprise Linux (RHEL) 7.x 上有 Red Hat Identity Manager (IdM)，则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
```

由于中央位置和边缘位置设计的区别，请不要在边缘堆栈中包含以下文件：

tls-everywhere-endpoints-dns.yaml

在边缘站点中会忽略此文件，它设置的端点会被从中央堆栈导出的端点覆盖。

haproxy-public-tls-certmonger.yaml

此文件导致部署失败，因为边缘没有公共端点。

附加信息：

- 使用 Ansible 实施 TLS-e

第 13 章 创建用于外部访问的 CEPH 密钥

警告

这个功能的内容在此发行版本中作为 *文档预览* 提供，因此红帽不会完全验证。仅用于测试，不要在生产环境中使用。

对 Ceph 存储的外部访问是从不是本地的站点访问 Ceph。Ceph 存储在边缘(DCN)站点外部，就像位于边缘的 Ceph 存储对中央位置外部使用。

当您使用 Ceph 存储部署 central 或 DCN 站点时，您可以选择将默认的 **openstack** keyring 用于本地和外部访问。Altenatively，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。在下面的示例中，密钥名称为 **external**。

如果您使用单独的密钥供非本地站点访问，您可以在不中断本地访问的情况下撤销和重新发布外部密钥以响应安全事件。但是，将单独的密钥用于外部访问将导致无法访问某些功能，如跨可用性区域备份和离线卷迁移。您必须平衡安全性与所需功能集的需求。

默认情况下，中央和所有 DCN 站点的密钥将被共享。

13.1. 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，使用上一命令的输出为密钥创建一个 **ceph_keys.yaml** 环境文件，其中包含以下内容：

```
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. 在站点的部署中包含 **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
```

13.2. 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 13.1 节 “创建用于外部访问的 Ceph 密钥”。这应该为中央站点和边缘站点完成。

- 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 \
     --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 **外部** 的名称，如这些示例所示，请创建 **glance_update.yaml** 类似如下：

     ```
     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，从中央位置包含 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 \
--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/cephadm/cephadm.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
```

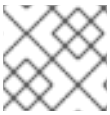
附录 A. 部署迁移选项

本节包括 DCN 存储的相关主题验证，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，确保中央站点和边缘站点的部署正常工作。

您可以将镜像导入到 Glance 中，在本地文件系统上可用或 web 服务器上可用。



注意

始终在中央站点中保存镜像副本，即使中央位置上没有使用该镜像的实例。

先决条件

1. 使用 **glance store-info 命令检查通过镜像服务提供的存储**。在以下示例中，提供三个存储：central、dcn1 和 dcn2。它们分别对应于中央位置和边缘站点的 glance 存储：

```
$ 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。如果镜像不是原始格式，则必须在将其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。

此流程假设默认镜像转换插件在 glance 中已启用。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 中。

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，它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，**storage** 键（即属性映射中的最后一个项目）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

始终将镜像副本存储在中央站点中，即使该站点上没有虚拟机使用它。

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根卷。

流程

1. 识别要作为卷创建的镜像的 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 命令来列出卷池，来验证卷是否基于镜像。

```
$ 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. 确认您可以为实例的 root 卷创建 cinder 快照。确保服务器已停止静止数据，以创建干净的快照。使用 *--force* 选项，因为当实例关闭时，卷状态会一直 处于**使用状**态。

```
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. 确认可以在站点间创建并复制镜像快照

1. 验证您可以在 dcn0 站点创建一个新镜像。确保服务器停止静默数据以创建干净的快照：

```
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** 边缘站点复制回 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

可以使用预先存在的网络配置将现有云迁移到具有 spine leaf 架构的现有云。为此，需要满足以下条件：

- 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 spine leaf，只有一个叶叶 **0**。您可以按照以下步骤在部署中添加额外的置备。

1. 将所需的子网添加到 undercloud.conf，如在 undercloud 中配置路由 spine-leaf 所示。
2. 重新运行 undercloud 安装命令 **openstack undercloud install**。
3. 将所需的额外网络和角色分别添加到 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，重新运行包含云部署的所有相关 heat 模板的 overcloud 安装脚本。

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。

您无法分割现有的堆栈。如果需要，您可以缩减现有堆栈以删除计算节点。然后，这些计算节点可以添加到边缘站点。

注意

如果删除了所有计算节点，则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)架构间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。块存储服务不在 DCN 站点存储备份。

先决条件

- 部署可选的块存储备份服务。如需更多信息，请参阅 *备份块存储卷*中的块存储备份服务部署。
- 块存储(cinder) REST API 微版本 3.51 或更高版本。
- 所有站点都必须使用通用的 **openstack** cephx 客户端名称。如需更多信息，请参阅 *部署分布式 Compute 节点(DCN)架构* 中的 为外部访问创建 Ceph 密钥。

流程

1. 在第一个 DCN 站点中创建卷的备份：

```
$ 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 Ceph 密钥环出现问题，您可能需要重启 **cinder-backup** 容器，以便从主机中的密钥环复制到容器。

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 < **volume_size** > 替换为等于或大于原始卷大小的 GB 值。

A.5. OVERCLOUD 采用和准备 DCN 环境

您必须执行以下任务才能采用 overcloud：

- 每个站点都单独升级，一个站点从中央位置开始。
- 将网络和主机置备配置导出到 overcloud，用于中央位置堆栈。'suggestion:-0+0
- 定义新容器和其他兼容性配置。

采用后，您必须运行执行以下任务的升级准备脚本：

- 将 overcloud 计划更新至 OpenStack Platform 17.1
- 为升级准备节点

有关此升级过程的持续时间和影响的详情，请参阅升级 持续时间和影响。

先决条件

- 所有节点都处于 **ACTIVE** 状态：

```
$ openstack baremetal node list
```

如果有任何节点处于 **MAINTENANCE** 状态，请将它们设置为 **ACTIVE** ：

```
$ openstack baremetal node maintenance unset <node_uuid>
```

- 将 **<node_uuid** > 替换为节点的 UUID。

流程

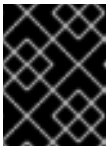
1. 以 **stack** 用户身份登录 undercloud 主机。
2. 查找 **stackrc** undercloud 凭证文件：

```
$ source ~/stackrc
```

3. 验证 undercloud 升级过程中导出的以下文件是否包含 overcloud 升级的预期配置。您可以在 **~/overcloud-deploy** 目录中找到以下文件：
   - **tripleo-<stack>-passwords.yaml**
   - **tripleo-<stack>-network-data.yaml**
   - **tripleo-<stack>-virtual-ips.yaml**
   - **tripleo-<stack>-baremetal-deployment.yaml**

注意

如果在 undercloud 升级后没有生成这些文件，请联系红帽支持。



重要

如果您有多单元环境，请参阅 Overcloud 采用多单元环境，例如将文件复制到每个单元堆栈的示例。

4. 在主堆栈上，将 password **.yaml** 文件复制到 **~/overcloud-deploy/$(<stack>)** 目录。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-passwords.yaml ~/overcloud-
deploy/<stack>/<stack>-passwords.yaml
```

- 将 **<stack>** 替换为您的堆栈的名称。

5. 如果您要在中央位置执行准备和采用，请将 **network-data.yaml** 文件复制到 stack 用户的主目录并部署网络。这只适用于中央位置：

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-network-data.yaml ~/
$ mkdir /home/stack/overcloud_adopt
$ openstack overcloud network provision --debug \
--output /home/stack/overcloud_adopt/generated-networks-deployed.yaml tripleo-central-network-data.yaml
```

有关更多信息，请参阅*使用 director 安装和管理 Red Hat OpenStack Platform* 中的置备和部署 overcloud。

6. 如果您要在中央位置执行准备和采用，请将 **virtual-ips.yaml** 文件复制到 stack 用户的主目录，并置备网络 VIP。这只适用于中央位置：

```
$ cp /home/stack/overcloud-deploy/central/tripleo-central-virtual-ips.yaml ~/
$ openstack overcloud network vip provision --debug \
--stack <stack> --output \
/home/stack/overcloud_adopt/generated-vip-deployed.yaml tripleo-central-virtual-ips.yaml
```

7. 在主堆栈上，将 **baremetal-deployment.yaml** 文件复制到 stack 用户的主目录，并置备 overcloud 节点。在环境中的每个堆栈中重复此步骤：

```
$ cp ~/overcloud-deploy/<stack>/tripleo-<stack>-baremetal-deployment.yaml ~/
$ openstack overcloud node provision --debug --stack <stack> \
--output /home/stack/overcloud_adopt/baremetal-central-deployment.yaml \
tripleo-<stack>-baremetal-deployment.yaml
```

注意

这是 overcloud 采用的最后一步。如果您的 overcloud 采用用 10 分钟的时间完成，请联系红帽支持。

8. 完成以下步骤以准备容器：

   a. 备份用于 undercloud 升级的 **containers-prepare-parameter.yaml** 文件：

```
$ cp containers-prepare-parameter.yaml \
containers-prepare-parameter.yaml.orig
```

   b. 在运行脚本以更新 **containers-prepare-parameter.yaml** 文件前，定义以下环境变量：

   - **NAMESPACE**: UBI9 镜像的命名空间。例如，**NAMESPACE="'namespace":"example.redhat.com:5002",'**
   - **EL8_NAMESPACE** ：UBI8 镜像的命名空间。
   - **NEUTRON_DRIVER** ：要使用的驱动程序，并确定要使用的 OpenStack Networking (neutron)容器。设置为用于部署原始堆栈的容器类型。例如，设置为 **NEUTRON_DRIVER="'neutron_driver":"ovn",** ' 以使用基于 OVN 的容器。
   - **EL8_TAGS** ：UBI8 镜像的标签，如 **EL8_TAGS="'tag":"17.1",'**。
     - 将 **"17.**1" 替换为您在内容视图中使用的标签。

- **EL9_TAGS** ：UBI9 镜像的标签，如 **EL9_TAGS='"tag":"17.1",'**。
  - 将 **"17.**1" 替换为您在内容视图中使用的标签。
    如需有关 **tag** 参数的更多信息，请参阅自定义 *Red Hat OpenStack Platform 部署中的容器镜像准备参数*。
- **CONTROL_PLANE_ROLES** ：使用 **--role** 选项的 control plane 角色列表，如 **--role ControllerOpenstack, --role Database, --role Messaging, --role Networker, --role CephStorage**。要查看环境中的 control plane 角色列表，请运行以下命令：

```
$ export STACK=<stack> \
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -vi compute
```

  - 将 **<stack>** 替换为您的堆栈的名称。
- **COMPUTE_ROLES** ：使用 **--role** 选项的 Compute 角色列表，如 **--Compute-1**。要查看环境中的 Compute 角色列表，请运行以下命令：

```
$ sudo awk '/tripleo_role_name/ {print "--role " $2}' \
/var/lib/mistral/${STACK}/tripleo-ansible-inventory.yaml \
| grep -i compute
```

- **CEPH_OVERRIDE** ：如果您部署 Red Hat Ceph Storage，请指定 Red Hat Ceph Storage 5 容器镜像。例如：
  CEPH_OVERRIDE='"ceph_image":"rhceph-5-rhel8","ceph_tag":"<latest>",'
  - 将 **<latest** > 替换为最新的 **ceph_tag** 版本，如 **5-499**。
    以下是 **containers-prepare-parameter.yaml** 文件配置示例：

```
NAMESPACE='"namespace":"registry.redhat.io/rhosp-rhel9",'
EL8_NAMESPACE='"namespace":"registry.redhat.io/rhosp-rhel8",'
NEUTRON_DRIVER='"neutron_driver":"ovn",'
EL8_TAGS='"tag":"17.1",'
EL9_TAGS='"tag":"17.1",'
CONTROL_PLANE_ROLES="--role Controller"
COMPUTE_ROLES="--role Compute"
CEPH_TAGS='"ceph_tag":"5",'
```

c. 运行以下脚本以更新 **containers-prepare-parameter.yaml** 文件：

警告

如果您部署了 Red Hat Ceph Storage，请确保在执行以下命令前将 **CEPH_OVERRIDE** 环境变量设置为正确的值。如果不这样做，会导致升级 Red Hat Ceph Storage 时出现问题。

```
$ python3 /usr/share/openstack-tripleo-heat-templates/tools/multi-rhel-container-image-
```

```
prepare.py \
    ${COMPUTE_ROLES} \
    ${CONTROL_PLANE_ROLES} \
    --enable-multi-rhel \
    --excludes collectd \
    --excludes nova-libvirt \
    --minor-override "
{${EL8_TAGS}${EL8_NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}\"no_t
ag\":\"not_used\"}" \
    --major-override "
{${EL9_TAGS}${NAMESPACE}${CEPH_OVERRIDE}${NEUTRON_DRIVER}\"no_tag\":\
"not_used\"}" \
    --output-env-file \
   /home/stack/containers-prepare-parameter.yaml
```

multi-rhel-container-image-prepare.py 脚本支持以下参数：

--output-env-file

编写包含默认 **ContainerImagePrepare** 值的环境文件。

--local-push-destination

触发上传到本地 registry 的上传。

--enable-registry-login

启用允许系统在拉取容器前尝试登录到远程 registry 的标记。当没有使用 **--local-push-destination** 且目标系统有到远程 registry 的网络连接时，请使用此标志。不要将这个标志用于可能没有到远程 registry 的网络连接的 overcloud。

--enable-multi-rhel

启用多rhel。

--excludes

列出要排除的服务。

--major-override

列出主发行版本的覆盖参数。

--minor-override

列出次发行版本的覆盖参数。

--role

角色列表。

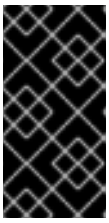
--role-file

role_data.yaml 文件。

d. 如果您部署了 Red Hat Ceph Storage，请打开 **containers-prepare-parameter.yaml** 文件，以确认指定了 Red Hat Ceph Storage 5 容器镜像，且没有引用 Red Hat Ceph Storage 6 容器镜像。

9. 如果您有 director 部署的 Red Hat Ceph Storage 部署，请创建一个名为 **ceph_params.yaml** 的文件，并包含以下内容：

```
parameter_defaults:
  CephSpecFqdn: true
  CephConfigPath: "/etc/ceph"
  CephAnsibleRepo: "rhceph-5-tools-for-rhel-8-x86_64-rpms"
  DeployedCeph: true
```



重要

在 RHOSP 升级完成后不要删除 **ceph_params.yaml** 文件。此文件必须存在于 director 部署的 Red Hat Ceph Storage 环境中。此外，每次运行 **openstack overcloud deploy** 时，都必须包含 **ceph_params.yaml** 文件，例如 **-e ceph_params.yaml**。

注意

如果您的 Red Hat Ceph Storage 部署包含短名称，您必须将 **CephSpecFqdn** 参数设置为 **false**。如果设置为 **true**，则清单会生成一个短名称和域名，从而导致 Red Hat Ceph Storage 升级失败。

10. 在 templates 目录中创建一个名为 **upgrade-environment.yaml** 的环境文件，并包含以下内容：

```
parameter_defaults:
  ExtraConfig:
    nova::workarounds::disable_compute_service_check_for_ffu: true
  DnsServers: ["<dns_servers>"]
  DockerInsecureRegistryAddress: <undercloud_FQDN>
  UpgradeInitCommand: |
    sudo subscription-manager repos --disable=*
      if $( grep -q  9.2  /etc/os-release )
      then
        sudo subscription-manager repos --enable=rhel-9-for-x86_64-baseos-eus-rpms --enable=rhel-9-for-x86_64-appstream-eus-rpms --enable=rhel-9-for-x86_64-highavailability-eus-rpms --enable=openstack-17.1-for-rhel-9-x86_64-rpms --enable=fast-datapath-for-rhel-9-x86_64-rpms
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-tools-for-rhel-9-x86_64-rpms
        sudo subscription-manager release --set=9.2
      else
        sudo subscription-manager repos --enable=rhel-8-for-x86_64-baseos-tus-rpms --enable=rhel-8-for-x86_64-appstream-tus-rpms --enable=rhel-8-for-x86_64-highavailability-tus-rpms --enable=openstack-17.1-for-rhel-8-x86_64-rpms --enable=fast-datapath-for-rhel-8-x86_64-rpms
        sudo podman ps | grep -q ceph && subscription-manager repos --enable=rhceph-5-tools-for-rhel-8-x86_64-rpms
        sudo subscription-manager release --set=8.4
      fi

      if $(sudo podman ps | grep -q ceph )
      then
        sudo dnf -y install cephadm
      fi
```

- 将 **<dns_servers>** 替换为 DNS 服务器 IP 地址的逗号分隔列表，例如 **["10.0.0.36", "10.0.0.37"]**。

- 将 **<undercloud_FQDN** > 替换为 undercloud 主机的完全限定域名(FQDN)，例如 **"undercloud-0.ctlplane.redhat.local:8787"**。
  有关您可以在环境文件中配置的升级参数的更多信息，请参阅升级 参数。

11. 如果要在边缘位置执行准备和采用，请将 **AuthCloudName** 参数设置为中央位置的名称：

```
parameter_defaults:
  AuthCloudName: central
```

12. 如果部署了多个镜像服务(glance)存储，请将 copy-image 的镜像服务 API 策略设置为允许所有规则：

```
parameter_defaults:
    GlanceApiPolicies: {glance-copy_image: {key 'copy-image', value: ""}}
```

13. 在 undercloud 上，在 templates 目录中创建一个名为 **overcloud_upgrade_prepare.sh** 的文件。

- 您必须为环境中的每个堆栈创建此文件。此文件包含 overcloud 部署文件的原始内容，以及与您的环境相关的环境文件。
- 如果您要为 DCN 边缘位置创建 **overcloud_upgrade_prepare.sh**，您必须包含以下模板：
  - 包含导出的中央站点参数的环境模板。您可以在 **/home/stack/overcloud-deploy/centra/central-export.yaml** 中找到此文件。
  - **generated-networks-deployed.yaml**，生成的文件来自在中央位置运行 **openstack overcloud network provision** 命令。
  - **generated-vip-deployed.yaml**，从在中央位置运行 **openstack overcloud network vip provision** 命令的结果文件。+ 例如：

```
#!/bin/bash
openstack overcloud upgrade prepare --yes \
  --timeout 460 \
  --templates /usr/share/openstack-tripleo-heat-templates \
  --ntp-server 192.168.24.1 \
  --stack <stack> \
  -r /home/stack/roles_data.yaml \
  -e /home/stack/templates/internal.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  -e /home/stack/templates/network/network-environment.yaml \
  -e /home/stack/templates/inject-trust-anchor.yaml \
  -e /home/stack/templates/hostnames.yml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
  -e /home/stack/templates/nodes_data.yaml \
  -e /home/stack/templates/debug.yaml \
  -e /home/stack/templates/firstboot.yaml \
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/overcloud-params.yaml \
  -e /home/stack/overcloud-deploy/<stack>/overcloud-network-environment.yaml \
  -e /home/stack/overcloud-adopt/<stack>-passwords.yaml \
  -e /home/stack/overcloud_adopt/<stack>-baremetal-deployment.yaml \
  -e /home/stack/overcloud_adopt/generated-networks-deployed.yaml \
  -e /home/stack/overcloud_adopt/generated-vip-deployed.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/nova-hw-machine-type-upgrade.yaml \
  -e /home/stack/skip_rhel_release.yaml \
  -e ~/containers-prepare-parameter.yaml
```

注意

如果您有多单元环境，请参阅 Overcloud 采用多单元环境，例如，为每个单元堆栈创建 **overcloud_upgrade_prepare.sh** 文件。

a. 在原始 **network-environment.yaml** 文件中 (**/home/stack/templates/network/network-environment.yaml**)，删除所有指向 **OS::TripleO::*::Net::SoftwareConfig** 的 resource_registry 资源。

b. 在 **overcloud_upgrade_prepare.sh** 文件中，包含与您的环境相关的以下选项：

- 环境文件 (**upgrades-environment.yaml**) 带有特定于升级的参数 (**-e**)。
- 包含新容器镜像位置(**-e**)的环境文件(**containers-prepare-parameter.yaml**)。在大多数情况下，这是 undercloud 使用的环境文件。
- 环境文件(**skip_rhel_release.yaml**)，带有发行版本参数(-e)。
- 与您的部署相关的任何自定义配置文件(**-e**)。
- 如果适用，使用 **--roles-file** 的自定义角色(**roles_data**)文件。
- 对于 Ceph 部署，环境文件(**ceph_params.yaml**)以及 Ceph 参数(-e)。
- 在 overcloud 采用过程中生成的文件(network**-deployed.yaml**,**vip-deployed.yaml**,**baremetal-deployment.yaml**) (-e)。
- 如果适用，带有您的 IPA 服务(-e)的环境文件(**ipa-environment.yaml**)。
- 如果您使用可组合网络，则使用 **--network-file** 的(**network_data**)文件。

注意

不要在 overcloud 部署文件或 **overcloud_upgrade_prepare.sh** 文件中包括 **network-isolation.yaml** 文件。网络隔离在 **network_data.yaml** 文件中定义。

- 如果使用自定义堆栈名称，请使用 **--stack** 选项传递名称。

注意

您必须在模板中包含 **nova-hw-machine-type-upgrade.yaml** 文件，直到所有 RHEL 8 Compute 节点都升级到环境中的 RHEL 9。如果排除了此文件，则 **/var/log/containers/nova** 目录中的 **nova_compute.log** 中会出现一个错误。将所有 RHEL 8 Compute 节点升级到 RHEL 9 后，您可以从配置中删除此文件并更新堆栈。

a. 在 director 部署的 Red Hat Ceph Storage 用例中，如果您在您要升级的部署中通过 NFS 启用共享文件系统服务(manila)，您必须在 **overcloud_upgrade_prepare.sh** 脚本文件末尾指定额外的环境文件。您必须在脚本末尾添加环境文件，因为它会覆盖之前在脚本中指定的另一个环境文件：

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-cephfsganesha-config.yaml
```

b. 在外部 Red Hat Ceph Storage 用例中，如果您在您要升级的部署中通过 NFS 启用共享文件系统服务(manila)，您必须检查 **overcloud_upgrade_prepare.sh** 脚本中的关联环境文件是否指向 tripleo-based **ceph-nfs** 角色。如果存在，请删除以下环境文件：

```
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/manila-cephfsganesha-config.yaml
```

并添加以下环境文件：

```
-e /usr/share/openstack-tripleo-heat-templates/environments/manila-cephfsganesha-config.yaml
```

14. 为环境中的每个堆栈运行升级准备脚本：

```
$ source stackrc
$ chmod 755 /home/stack/overcloud_upgrade_prepare.sh
$ sh /home/stack/overcloud_upgrade_prepare.sh
```

注意

如果您有多单元环境，您必须为每个为每个单元堆栈创建的 **overcloud_upgrade_prepare.sh** 文件运行脚本。例如，请参阅 Overcloud 对多单元环境采用。

15. 等待升级准备完成。

16. 下载容器镜像：

```
$ openstack overcloud external-upgrade run --stack <stack> --tags container_image_prepare
```