Red Hat OpenStack Platform 17.1

配置 spine-leaf 网络

使用 Red Hat OpenStack Platform director 配置路由的 spine-leaf 网络

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 配置 spine-leaf 网络

使用 Red Hat OpenStack Platform director 配置路由的 spine-leaf 网络

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

本指南提供了有关如何在 overcloud 上配置路由的 spine-leaf 网络的基本场景。这包括配置 undercloud，编写主配置文件，并为节点创建角色。

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 SPINE-LEAF 网络简介

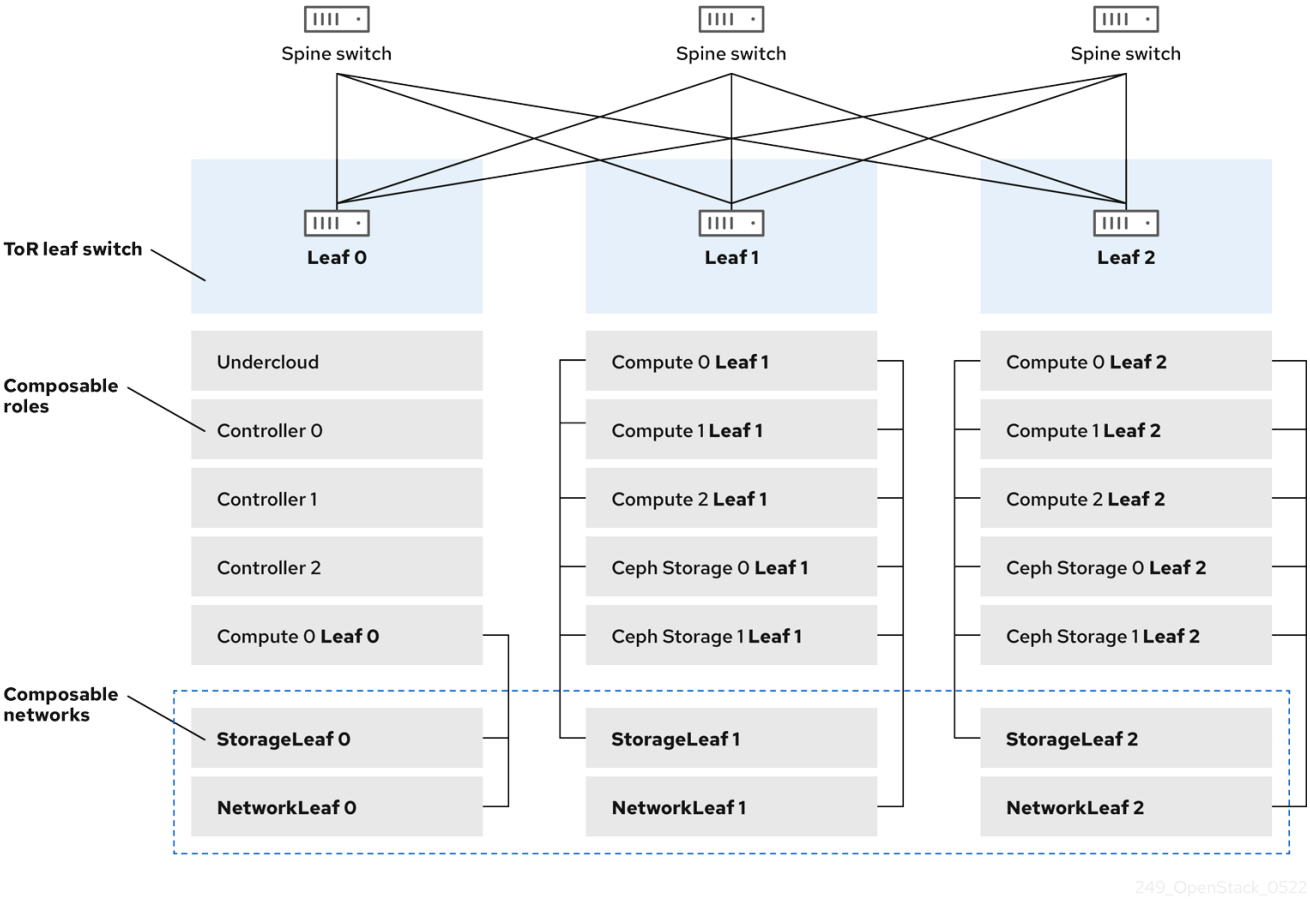
以下章节提供有关为您的 Red Hat OpenStack Platform 环境构建 spine-leaf 网络拓扑的信息。这包括完整的端到端场景和示例文件，以帮助在您自己的环境中复制更广泛的网络拓扑。

1.1. SPINE-LEAF 网络

Red Hat OpenStack Platform 有一个可组合的网络架构，可用于将网络适应路由的 spine-leaf 数据中心拓扑。在路由的 spine-leaf 的实际应用程序中，leaf 以可组合的计算或存储角色表示，通常位于数据中心机架中，如图 1.1 所示，"Routed spine-leaf example" 所示。*Leaf 0* 机架具有 undercloud 节点、Controller 节点和 Compute 节点。可组合网络呈现给节点，它们已分配给可组合角色。下图显示了以下配置：

- **StorageLeaf** 网络会看到 Ceph 存储和 Compute 节点。
- **NetworkLeaf** 代表您可能要编写的任何网络的示例。

图 1.1. 路由 spine-leaf 示例



1.2. SPINE-LEAF 网络拓扑

spine-leaf 场景利用 OpenStack Networking (neutron)功能在单个网络片段中定义多个子网。每个网络都使用基础网络，它充当 Leaf 0。director 创建 Leaf 1 和 Leaf 2 子网，作为主网络的片段。

这个场景使用以下网络：

表 1.1. leaf 0 Networks （基础网络）

Network	附加的角色	子网
Provisioning / Ctlplane / Leaf0	Controller, ComputeLeaf0, CephStorageLeaf0	192.168.10.0/24
存储	Controller, ComputeLeaf0, CephStorageLeaf0	172.16.0.0/24
StorageMgmt	Controller, CephStorageLeaf0	172.17.0.0/24
InternalApi	Controller, ComputeLeaf0	172.18.0.0/24
租户 [1]	Controller, ComputeLeaf0	172.19.0.0/24
外部	Controller	10.1.1.0/24

[1] 租户网络也称为项目网络。

表 1.2. leaf 1 Networks

Network	附加的角色	子网
Provisioning / Ctlplane / Leaf1	ComputeLeaf1, CephStorageLeaf1	192.168.11.0/24
StorageLeaf1	ComputeLeaf1, CephStorageLeaf1	172.16.1.0/24
StorageMgmtLeaf1	CephStorageLeaf1	172.17.1.0/24
InternalApiLeaf1	ComputeLeaf1	172.18.1.0/24
TenantLeaf1 [1]	ComputeLeaf1	172.19.1.0/24

[1] 租户网络也称为项目网络。

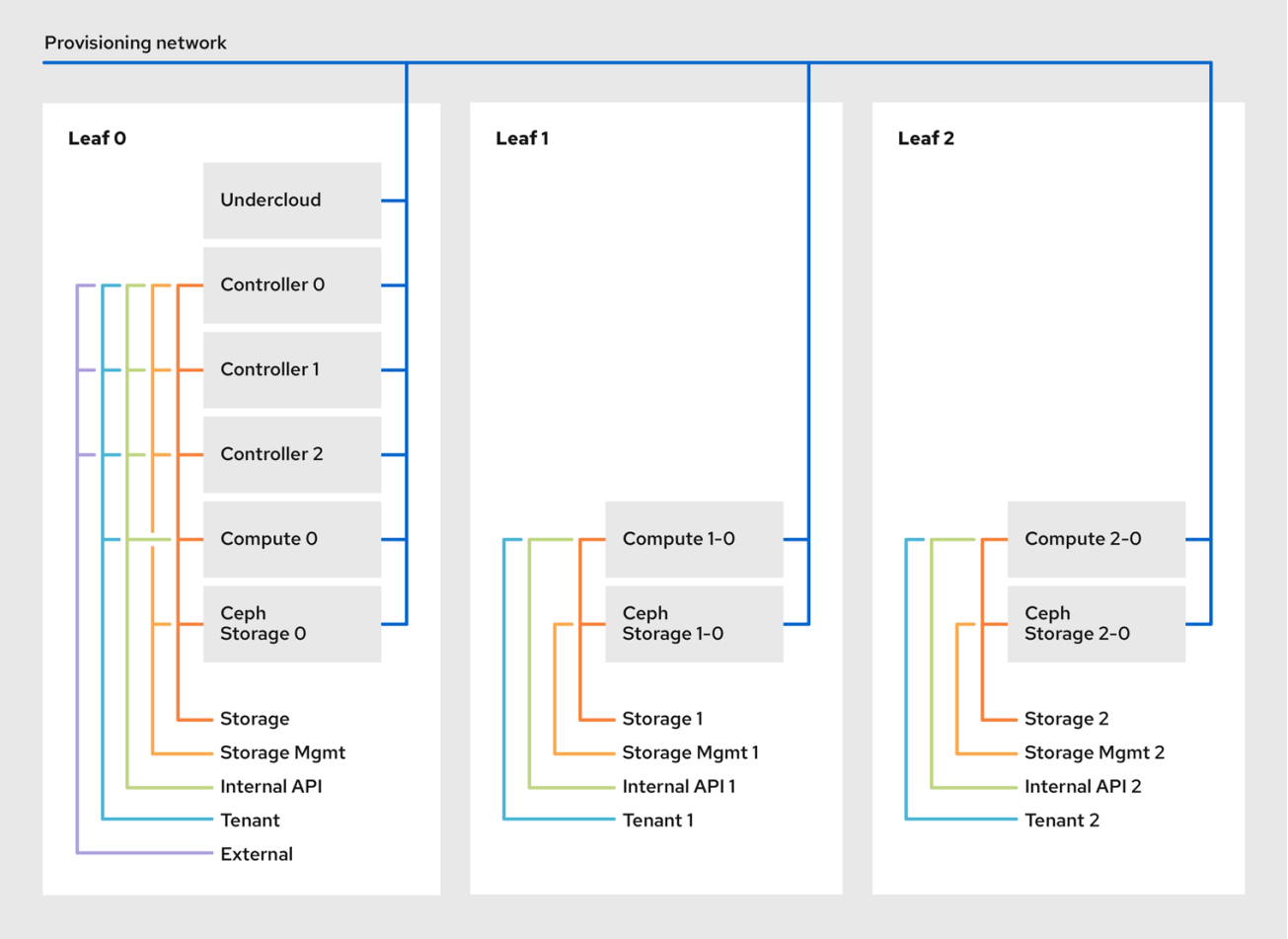
表 1.3. leaf 2 Networks

Network	附加的角色	子网
Provisioning / Ctlplane / Leaf2	ComputeLeaf2, CephStorageLeaf2	192.168.12.0/24
StorageLeaf2	ComputeLeaf2, CephStorageLeaf2	172.16.2.0/24
StorageMgmtLeaf2	CephStorageLeaf2	172.17.2.0/24

Network	附加的角色	子网
InternalApiLeaf2	ComputeLeaf2	172.18.2.0/24
TenantLeaf2 [1]	ComputeLeaf2	172.19.2.0/24

[1] 租户网络也称为项目网络。

图 1.2. spine-leaf 网络拓扑



249_OpenStack_0522

1.3. SPINE-LEAF 要求

要在带有 L3 路由架构的网络中部署 overcloud，请完成以下步骤：

layer-3 路由

配置网络基础架构的路由，以启用不同 L2 网段之间的流量。您可以静态或动态配置此路由。

DHCP-Relay

每个不是 undercloud 本地的 L2 片段必须提供 **dhcp-relay**。您必须在连接的 undercloud 的置备网络片段上将 DHCP 请求转发到 undercloud。

注意

undercloud 使用两个 DHCP 服务器。一个用于裸机节点内省，另一个用于部署 overcloud 节点。确保您读取 DHCP 转发配置，以了解配置 **dhcp-relay** 时的要求。

1.4. SPINE-LEAF 限制

- 一些角色（如 Controller 角色）使用虚拟 IP 地址和集群。此功能背后的机制需要这些节点之间的 L2 网络连接。您必须将这些节点放在同一个叶型中。
- 类似的限制适用于 Networker 节点。网络服务使用虚拟路由器冗余协议(VRRP)在网络中实施高可用性默认路径。由于 VRRP 使用虚拟路由器 IP 地址，您必须将 master 和备份节点连接到相同的 L2 网络段。
- 当您将租户或提供商网络与 VLAN 分段搭配使用时，您必须在所有 Networker 和 Compute 节点之间共享特定的 VLAN。

注意

可以使用多组 Networker 节点配置网络服务。每个 Networker 节点都为其网络共享路由，VRRP 在每个组 Networker 节点内提供高可用性默认路径。在这种配置中，共享网络的所有网络节点都必须位于同一 L2 网络段中。

第 2 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。

2.1. 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：

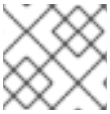
```
[stack@director ~]$ openstack undercloud install
```

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。

2.2. 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。

2.3. 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：

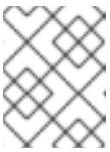
```
- name: <role>
  count: 1
  defaults:
    resource_class: baremetal.<ROLE>
```

- 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。

2.4. 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
```

2.5. 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

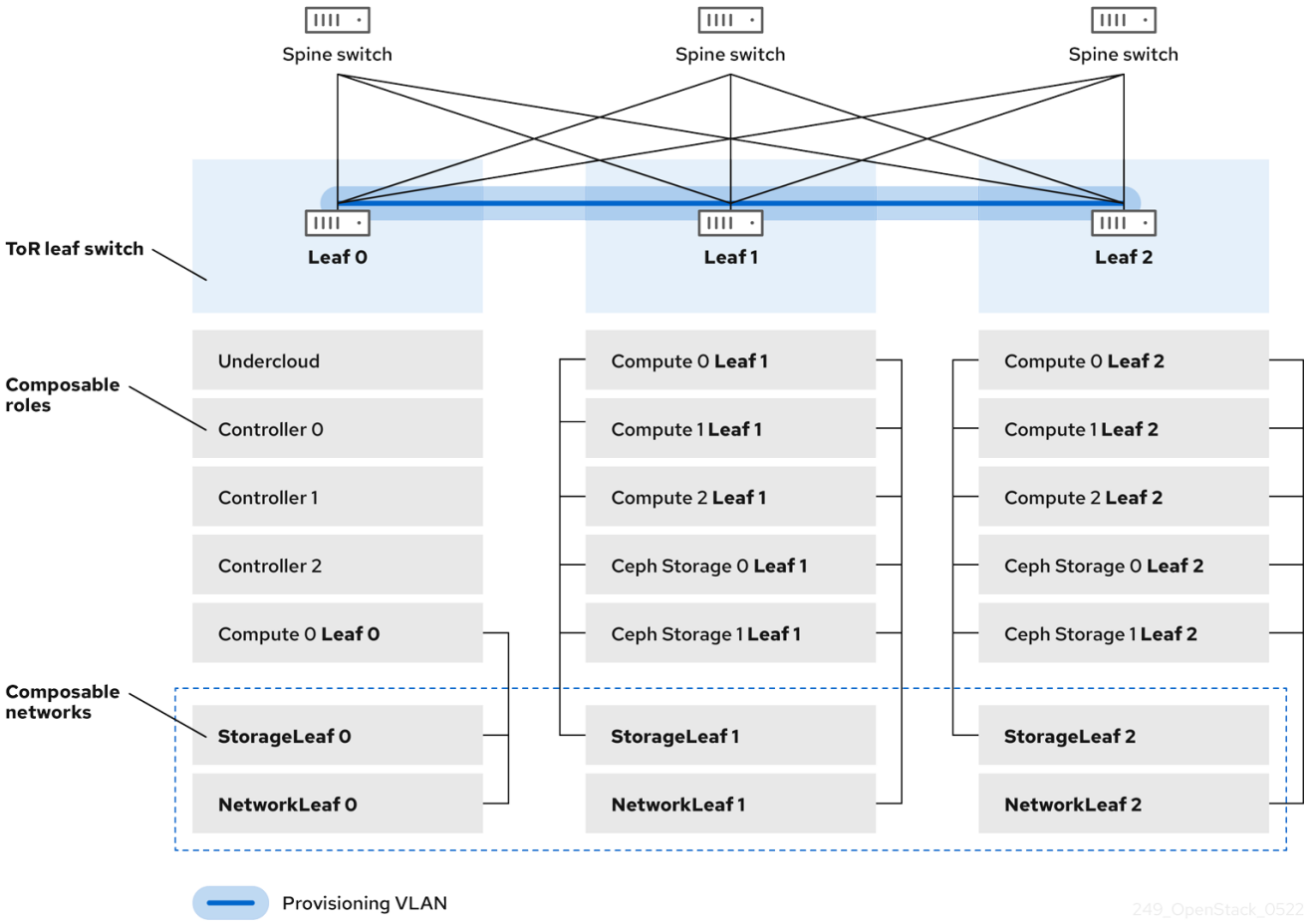
第 3 章 其他置备网络方法

本节介绍可用于配置 provisioning 网络的其他方法，以使用可组合网络容纳路由的 spine-leaf。

3.1. VLAN PROVISIONING 网络

在本例中，director 通过 provisioning 网络部署新的 overcloud 节点，并在 L3 拓扑中使用 VLAN 隧道。如需更多信息，请参阅 图 3.1、"VLAN 置备网络拓扑"。如果您使用 VLAN 置备网络，则 director DHCP 服务器可以将 **DHCPOFFER** 广播发送到任何叶。要建立此隧道，在 Top-of-Rack (ToR) leaf 交换机之间中继一个 VLAN。在下图中，Ceph 存储和 Compute 节点中可以看见 **StorageLeaf** 网；**NetworkLeaf** 代表您要编写的任何网络的示例。

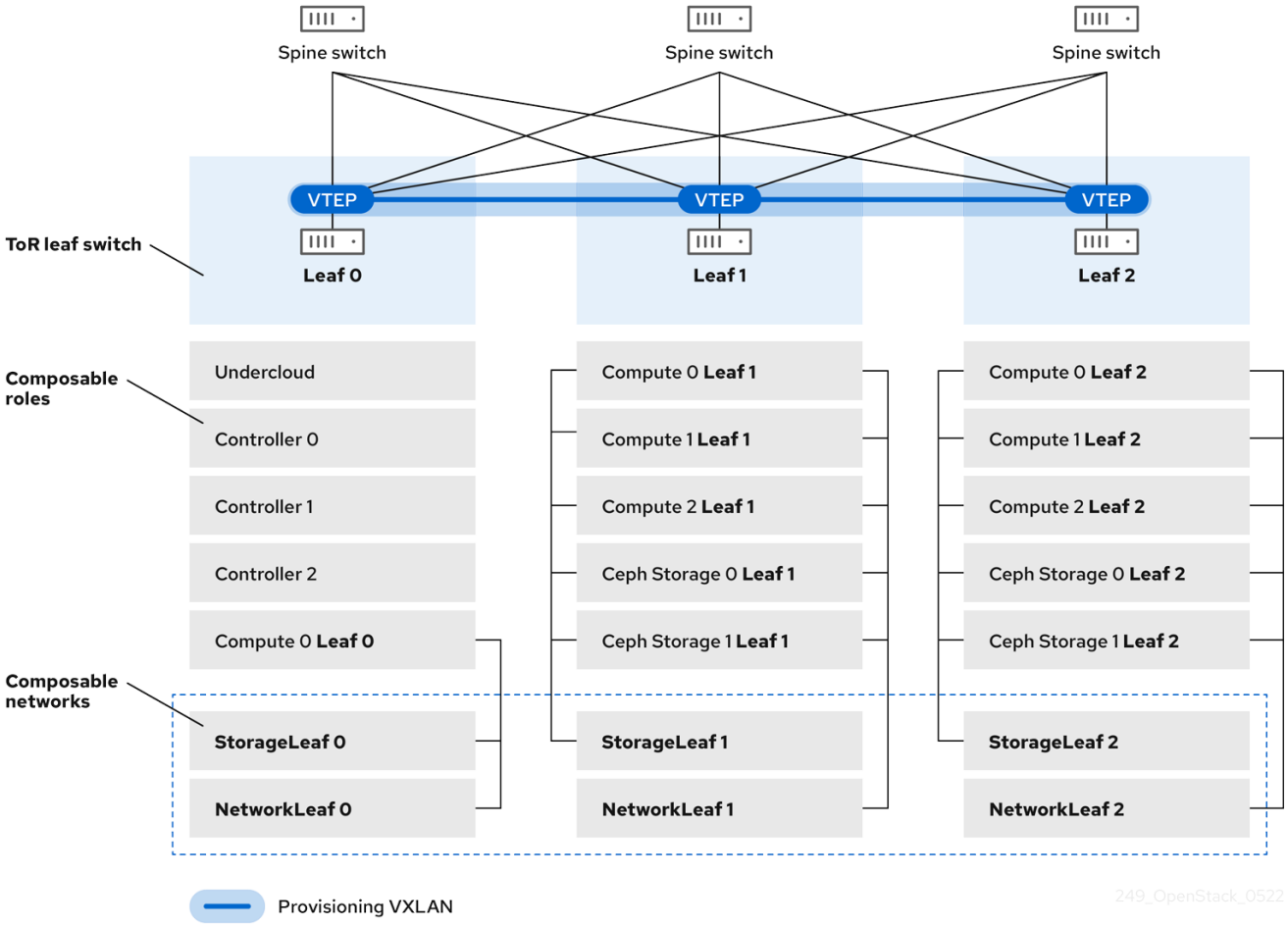
图 3.1. VLAN 置备网络拓扑



3.2. VXLAN 置备网络

在本例中，director 通过 provisioning 网络部署新的 overcloud 节点，并使用 VXLAN 隧道跨第 3 层拓扑来跨越第 3 层拓扑。如需更多信息，请参阅 图 3.2、"VXLAN 置备网络拓扑"。如果您使用 VXLAN 置备网络，则 director DHCP 服务器可以将 **DHCPOFFER** 广播发送到任何叶。要建立此隧道，请在 Top-of-Rack (ToR) leaf 交换机上配置 VXLAN 端点。

图 3.2. VXLAN 置备网络拓扑



第 4 章 配置 OVERCLOUD

使用 Red Hat OpenStack Platform (RHOSP) director 在 RHOSP overcloud 中安装和配置 spine leaf 网络。高级步骤有：

1. 为每个叶定义 overcloud 网络。
2. 为每个叶叶创建一个可组合角色，并将可组合网络附加到每个对应的角色。
3. 为每个角色创建一个唯一的 NIC 配置。
4. 更改网桥映射，以便每个叶组都通过该叶上的特定网桥或 VLAN 路由流量。
5. 为您的 overcloud 端点定义虚拟 IP (VIP)，并确定每个 VIP 的子网 。
6. 置备 overcloud 网络和 overcloud VIP 。
7. 在 overcloud 中注册裸机节点。

注意

如果您使用预置备节点，请跳过第 7 步、8 和 9 步。

8. 内省 overcloud 中的裸机节点。
9. 置备裸机节点。
10. 使用您在前面的步骤中设置的配置部署 overcloud。

4.1. 定义叶网络

Red Hat OpenStack Platform (RHOSP) director 从您构造的 YAML 格式的自定义网络创建 overcloud leaf 网络。此自定义网络定义文件列出了每个可组合网络及其属性，也定义每个叶网络所需的子网。

完成以下步骤，创建一个 YAML 格式的自定义网络定义文件，该文件包含 overcloud 上 spine-leaf 网络的规格。之后，置备过程会从部署 RHOSP overcloud 时包含的网络定义文件创建一个 heat 环境文件。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。

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** 下创建一个 **templates** 目录：

```
$ mkdir /home/stack/templates
```

4. 将默认网络定义模板 **routed-networks.yaml** 复制到您的自定义 **templates** 目录中：

示例

```
$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/\
routed-networks.yaml \
/home/stack/templates/spine-leaf-networks-data.yaml
```

5. 编辑网络定义模板的副本，将每个基本网络和每个关联的叶子网定义为可组合网络项。

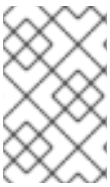
提示

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 网络定义文件配置选项。

示例

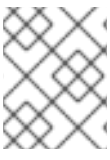
以下示例演示了如何定义内部 API 网络及其叶网络：

```
- name: InternalApi
  name_lower: internal_api
  vip: true
  mtu: 1500
  subnets:
    internal_api_subnet:
      ip_subnet: 172.16.32.0/24
      gateway_ip: 172.16.32.1
      allocation_pools:
        - start: 172.16.32.4
          end: 172.16.32.250
      vlan: 20
    internal_api_leaf1_subnet:
      ip_subnet: 172.16.33.0/24
      gateway_ip: 172.16.33.1
      allocation_pools:
        - start: 172.16.33.4
          end: 172.16.33.250
      vlan: 30
    internal_api_leaf2_subnet:
      ip_subnet: 172.16.34.0/24
      gateway_ip: 172.16.34.1
      allocation_pools:
        - start: 172.16.34.4
          end: 172.16.34.250
      vlan: 40
```



注意

不要在自定义网络定义模板中定义 Control Plane 网络，因为 undercloud 已创建了这些网络。但是，您必须手动设置参数，以便 overcloud 能够相应地配置 NIC。如需更多信息，请参阅在 undercloud 中配置路由 spine-leaf。



注意

RHOSP 不对网络子网和 **allocation_pools** 值执行自动验证。确保以统一方式定义这些值，并且它们不会与现有网络冲突。

注意

添加 **vip** 参数，并为托管基于 Controller 的服务的网络将值设为 **true**。在本例中，**InternalApi** 网络包含这些服务。

后续步骤

1. 注意您创建的自定义网络定义文件的路径和文件名。稍后为 RHOSP overcloud 置备网络时，需要此信息。
2. 继续执行下一步，定义叶角色并附加网络。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 网络定义文件配置选项

4.2. 定义叶角色和附加网络

Red Hat OpenStack Platform (RHOSP) director 为每个叶组创建一个可组合角色，并将可组合网络附加到您构建的 roles 模板中的每个对应的角色。首先，从 director 核心模板复制默认的 Controller、Compute 和 Ceph Storage 角色，并进行修改以满足您的环境的需求。创建所有单独的角色后，您将运行 **openstack overcloud roles generate** 命令，将它们串联为一个大型自定义角色数据文件。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。

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. 将 RHOSP 附带的 Controller、Compute 和 Ceph Storage 角色的默认角色复制到 **stack** 用户的主目录。重命名文件以指示它们是叶的 0 ：

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Controller.yaml \
~/roles/Controller0.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Compute.yaml \
~/roles/Compute0.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/CephStorage.yaml \
~/roles/CephStorage0.yaml
```

4. 复制 leaf 0 文件以创建您的叶 1 和叶 2 文件：

```
$ cp ~/roles/Compute0.yaml ~/roles/Compute1.yaml
$ cp ~/roles/Compute0.yaml ~/roles/Compute2.yaml
$ cp ~/roles/CephStorage0.yaml ~/roles/CephStorage1.yaml
$ cp ~/roles/CephStorage0.yaml ~/roles/CephStorage2.yaml
```

5. 编辑每个文件中的参数，使其与对应的叶参数保持一致。

提示

有关角色数据模板中的各种参数的信息，请参阅自定义 *Red Hat OpenStack Platform 部署* 指南中的 检查 角色参数。

示例 - ComputeLeaf0

```
- name: ComputeLeaf0
  HostnameFormatDefault: '%stackname%-compute-leaf0-%index%'
```

示例 - CephStorageLeaf0

```
- name: CephStorageLeaf0
  HostnameFormatDefault: '%stackname%-cephstorage-leaf0-%index%'
```

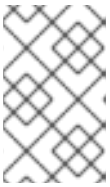
6. 编辑 leaf 1 和 leaf 2 文件中的 **network** 参数，以便它们与对应的 leaf network 参数保持一致。

示例 - ComputeLeaf1

```
- name: ComputeLeaf1
  networks:
    InternalApi:
      subnet: internal_api_leaf1
    Tenant:
      subnet: tenant_leaf1
    Storage:
      subnet: storage_leaf1
```

示例 - CephStorageLeaf1

```
- name: CephStorageLeaf1
  networks:
    Storage:
      subnet: storage_leaf1
    StorageMgmt:
      subnet: storage_mgmt_leaf1
```



注意

这只适用于 leaf 1 和 leaf 2。leaf 0 的 **network** 参数保留基本子网值，这些值是每个子网的小写名称以及 **_subnet** 后缀。例如，leaf 0 的内部 API 是 **internal_api_subnet**。

7. 在每个 Controller、Compute 和 （如果存在）Networker 角色文件中，将 OVN BGP 代理添加到 **ServicesDefault** 参数下的服务列表中：

示例

```
- name: ControllerRack1
  ...
  ServicesDefault:
    ...
```

```
    - OS::TripleO::Services::Frr
    - OS::TripleO::Services::OVNBgpAgent
    ...
```

8. 角色配置完成后，运行 **overcloud 角色 generate** 命令来生成完整的角色数据文件。

   示例

```
$ openstack overcloud roles generate --roles-path ~/roles \
-o spine-leaf-roles-data.yaml Controller Compute Compute1 Compute2 \
CephStorage CephStorage1 CephStorage2
```

   这会创建一个自定义角色数据文件，其中包含每个对应叶网络的所有自定义角色。

后续步骤

1. 注意 **overcloud 角色生成命令创建的自定义角色数据文件的路径和文件名**。在稍后部署 overcloud 时会使用此路径。
2. 继续下一步，为叶角色创建自定义 NIC 配置。

其他资源

- 在自定义 *Red Hat OpenStack Platform 部署* 指南中的 检查角色参数

4.3. 为叶角色创建自定义 NIC 配置

Red Hat OpenStack Platform (RHOSP) director 创建的每个角色都需要一个唯一的 NIC 配置。完成以下步骤以创建自定义 NIC 模板和一个自定义环境文件，将自定义模板映射到对应的角色。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。
- 您有一个自定义网络定义文件。
- 您有一个自定义角色数据文件。

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. 从其中一个默认 NIC 模板复制内容，以便为 NIC 配置创建自定义模板。

   示例

   在本例中，**single-nic-vlans** NIC 模板被复制，用于 NIC 配置的自定义模板：

```
$ cp -r /usr/share/ansible/roles/tripleo_network_config/\
templates/single-nic-vlans/* /home/stack/templates/spine-leaf-nics/.
```

4. 在您在上一步中创建的每个 NIC 模板中，更改 NIC 配置以匹配 spine-leaf 拓扑的具体内容。

示例

```
{% set mtu_list = [ctlplane_mtu] %}
{% for network in role_networks %}
{{ mtu_list.append(lookup('vars', networks_lower[network] ~ '_mtu')) }}
{%- endfor %}
{% set min_viable_mtu = mtu_list | max %}
network_config:
- type: ovs_bridge
  name: {{ neutron_physical_bridge_name }}
  mtu: {{ min_viable_mtu }}
  use_dhcp: false
  dns_servers: {{ ctlplane_dns_nameservers }}
  domain: {{ dns_search_domains }}
  addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  routes: {{ ctlplane_host_routes }}
  members:
  - type: interface
    name: nic1
    mtu: {{ min_viable_mtu }}
    # force the MAC address of the bridge to this interface
    primary: true
{% for network in role_networks %}
  - type: vlan
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    vlan_id: {{ lookup('vars', networks_lower[network] ~ '_vlan_id') }}
    addresses:
    - ip_netmask:
        {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars',
networks_lower[network] ~ '_cidr') }}
    routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endfor %}
```

提示

如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform 指南中的定义自定义网络接口模板*。

5. 创建自定义环境文件，如 **spine-leaf-nic-roles-map.yaml**，其中包含一个 **parameter_defaults** 部分，它将自定义 NIC 模板映射到每个自定义角色。

```
parameter_defaults:
  %%ROLE%%NetworkConfigTemplate: <path_to_ansible_jinja2_nic_config_file>
```

示例

```
parameter_defaults:
  Controller0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-
vlans.j2'
  Controller1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-
vlans.j2'
```

```
  Controller2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
```

后续步骤

1. 注意自定义 NIC 模板的路径和文件名以及将自定义 NIC 模板映射到每个自定义角色的自定义环境文件。在稍后部署 overcloud 时会使用此路径。
2. 继续执行下一步 配置叶型网络。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 定义自定义网络接口模板

4.4. 配置叶网络

在 spine leaf architecture 中，每个叶架构都通过该叶上的特定网桥或 VLAN 路由流量，这通常是边缘计算场景。因此，您必须更改 Red Hat OpenStack Platform (RHOSP) Controller 和 Compute 网络配置使用 **br-ex** 网桥的默认映射。

RHOSP director 在创建 undercloud 过程中创建 control plane 网络。但是，overcloud 需要访问每个叶的 control plane。要启用此访问权限，您必须在部署中定义附加参数。

完成以下步骤以创建自定义网络环境文件，其中包含单独的网络映射，并为 overcloud 设置对 control plane 网络的访问。

先决条件

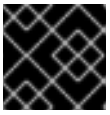
- 访问 **stack** 用户的 undercloud 主机和凭据。

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. 在新的自定义环境文件中，如 **spine-leaf-ctlplane.yaml**，创建一个 **parameter_defaults** 部分，并为使用默认 **br-ex** 网桥的每个叶设置 **NeutronBridgeMappings** 参数。



重要

您创建的自定义环境文件名称必须以 **.yaml** 或 **.template** 结尾。

- 对于扁平网络映射，列出 **NeutronFlatNetworks** 参数中的每个 leaf，并为每个叶设置 **NeutronBridgeMappings** 参数：

  示例

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"

  Compute2Parameters:
    NeutronBridgeMappings: "leaf2:br-ex"
```

  提示

  如需更多信息，请参阅 Chapter 17。networking (neutron)参数（*Overcloud 参数* 指南）

- 对于 VLAN 网络映射，将 **vlan** 添加到 **NeutronNetworkType**，并使用 **NeutronNetworkVLANRanges**，映射叶网络的 VLAN：

  示例

```
parameter_defaults:
  NeutronNetworkType: 'geneve,vlan'
  NeutronNetworkVLANRanges: 'leaf0:1:1000,leaf1:1:1000,leaf2:1:1000'

  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
```

```
Compute1Parameters:
  NeutronBridgeMappings: "leaf1:br-ex"

Compute2Parameters:
  NeutronBridgeMappings: "leaf2:br-ex"
```

注意

您可以在 spine-leaf topology 中使用扁平网络和 VLAN。

4. 使用 < **role>ControlPlaneSubnet** 参数为每个 spine-leaf 网络添加 control plane 子网映射：

示例

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    ControllerControlPlaneSubnet: leaf0
  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Controller1ControlPlaneSubnet: leaf0
  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Controller2ControlPlaneSubnet: leaf0
  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Compute0ControlPlaneSubnet: leaf0
  CephStorage0Parameters:
    CephStorage0ControlPlaneSubnet: leaf0
  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"
    Compute1ControlPlaneSubnet: leaf1
  CephStorage1Parameters:
    CephStorage1ControlPlaneSubnet: leaf1
  Compute2Parameters:
    NeutronBridgeMappings: "leaf2:br-ex"
    Compute2ControlPlaneSubnet: leaf2
  CephStorage2Parameters:
    CephStorage2ControlPlaneSubnet: leaf2
```

后续步骤

1. 请注意您创建的自定义网络环境文件的路径和文件名。部署 overcloud 时，稍后您将需要此信息。
2. 继续执行下一步，为虚拟 IP 地址 设置子网。

其他资源

- 第 17 章.networking (neutron)参数（*Overcloud 参数* 指南）

4.5. 为虚拟 IP 地址设置子网

默认情况下，Red Hat Openstack Platform (RHOSP) Controller 角色为每个网络托管虚拟 IP (VIP)地址。RHOSP overcloud 从每个网络的基本子网中获取 VIP，但 control plane 除外。control plane 使用 **ctlplane-subnet**，这是在标准 undercloud 安装过程中创建的默认子网名称。

在本文档中使用的 spine-leaf 示例中，默认基础调配网络为 **leaf0** 而不是 **ctlplane-subnet**。这意味着，您必须将值对 **subnet: leaf0** 添加到 **network:ctlplane** 参数，以便将子网映射到 **leaf0**。

完成以下步骤，创建一个 YAML 格式的自定义网络 VIP 定义文件，该文件包含 overcloud 上 VIP 的覆盖。之后，置备过程会从部署 RHOSP overcloud 时包含的网络 VIP 定义文件创建一个 heat 环境文件。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。

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. 在新的自定义网络 VIP 定义模板中，如 **spine-leaf-vip-data.yaml**，列出需要在控制器节点使用的特定子网上创建的虚拟 IP 地址。

   示例

```
- network: storage_mgmt
  subnet: storage_mgmt_subnet_leaf1
- network: internal_api
  subnet: internal_api_subnet_leaf1
- network: storage
  subnet: storage_subnet_leaf1
- network: external
  subnet: external_subnet_leaf1
  ip_address: 172.20.11.50
- network: ctlplane
  subnet: leaf0
- network: oc_provisioning
  subnet: oc_provisioning_subnet_leaf1
- network: storage_nfs
  subnet: storage_nfs_subnet_leaf1
```

   您可以在 **spine-leaf-vip-data.yaml** 文件中使用以下参数：

   network

   设置 neutron 网络名称。这是唯一必需的参数。

   ip_address

   设置 VIP 的 IP 地址。

   子网

   设置 neutron 子网名称。在创建虚拟 IP neutron 端口时，使用 指定子网。当部署使用路由网络时，需要此参数。

   dns_name

设置 FQDN （完全限定域名）。

name

设置虚拟 IP 名称。

提示

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 添加可组合网络。

后续步骤

1. 请注意您创建的自定义网络 VIP 定义模板的路径和文件名。您稍后会为 RHOSP overcloud 置备网络 VIP 时使用此路径。
2. 继续执行 overcloud 的下一步 Provisioning 网络和 VIP。

4.6. 为 OVERCLOUD 置备网络和 VIP

Red Hat OpenStack Platform (RHOSP)置备过程使用您的网络定义文件来创建一个包含您的网络规格的新 heat 环境文件。如果您的部署使用 VIP，RHOSP 会从 VIP 定义文件中创建一个新的 heat 环境文件。置备网络和 VIP 后，您稍后有两个用于部署 overcloud 的 heat 环境文件。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。
- 您有一个网络配置模板。
- 如果您使用 VIP，则有一个 VIP 定义模板。

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. 调配 overcloud 网络。
使用 **overcloud network provision** 命令，并提供之前创建的网络定义文件的路径。

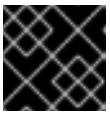
提示

如需更多信息，请参阅 *使用 director 安装和管理 Red Hat OpenStack Platform 指南* 中的配置和管理 overcloud 网络定义。

示例

在本例中，路径为 **/home/stack/templates/spine-leaf-networks-data.yaml**。使用 **--output** 参数为 命令创建的文件命名。

```
$ openstack overcloud network provision \
  --output spine-leaf-networks-provisioned.yaml \
 /home/stack/templates/spine-leaf-networks-data.yaml
```



重要

您指定的输出文件的名称必须以 **.yaml** 或 **.template** 结尾。

4. 置备 overcloud VIP。
使用 **overcloud network vip provision** 命令及 **--stack** 参数，将之前创建的 VIP 定义文件命名为之前创建的 VIP 定义文件。使用 **--output** 参数为 命令创建的文件命名。

提示

如需更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 配置和置备网络 VIP。

```
$ openstack overcloud network vip provision \
  --stack spine-leaf-overcloud \
 --output spine-leaf-vips-provisioned.yaml \
 /home/stack/templates/spine-leaf-vip-data.yaml
```

重要

您指定的输出文件的名称必须以 **.yaml** 或 **.template** 结尾。

5. 请注意生成的输出文件的路径和文件名。您稍后会在部署 overcloud 时使用此信息。

验证

- 您可以使用以下命令确认命令创建了 overcloud 网络和子网：

```
$ openstack network list
$ openstack subnet list
$ openstack network show <network>
$ openstack subnet show <subnet>
$ openstack port list
$ openstack port show <port>
```

将 <network>、<subnet> 和 <port> 替换为您要检查的网络、子网和端口的名称或 UUID。

后续步骤

1. 如果您使用预置备节点，请跳至 运行 overcloud 部署命令。
2. 否则，继续在 overcloud 上注册裸机节点。

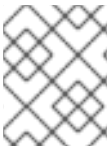
其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 配置和置备 overcloud 网络定义

- *使用 director 安装和管理 Red Hat OpenStack Platform 指南* 中的为 overcloud 配置和置备网络 VIP。
- *命令行界面参考中的* overcloud 网络置备
- *命令行界面参考中的* overcloud 网络 vip 置备

4.7. 在 OVERCLOUD 上注册裸机节点

Red Hat OpenStack Platform (RHOSP) director 需要自定义节点定义模板，用于指定物理机的硬件和电源管理详情。您可以使用 JSON 或 YAML 格式创建此模板。在将物理计算机注册为裸机节点后，您要内省它们，最后进行调配。



注意

如果您使用预置备节点，您可以跳过注册、内省和调配裸机节点，并前往 部署启用了 spine-leaf 的 overcloud。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。

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. 创建新节点定义模板，如 **baremetal-nodes.yaml**。添加包含其硬件和电源管理详情的物理机列表。

示例

```
nodes:
  - name: "node01"
    ports:
      - address: "aa:aa:aa:aa:aa:aa"
        physical_network: ctlplane
        local_link_connection:
          switch_id: 52:54:00:00:00:00
          port_id: p0
    cpu: 4
    memory: 6144
    disk: 40
    arch: "x86_64"
    pm_type: "ipmi"
    pm_user: "admin"
    pm_password: "p@55w0rd!"
    pm_addr: "192.168.24.205"
  - name: "node02"
    ports:
      - address: "bb:bb:bb:bb:bb:bb"
```

```
      physical_network: ctlplane
      local_link_connection:
        switch_id: 52:54:00:00:00:00
        port_id: p0
cpu: 4
memory: 6144
disk: 40
arch: "x86_64"
pm_type: "ipmi"
pm_user: "admin"
pm_password: "p@55w0rd!"
pm_addr: "192.168.24.206"
```

提示

如需有关模板参数值和 JSON 示例的更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 注册节点。

4. 验证模板格式化和语法。

示例

```
$ openstack overcloud node import --validate-only ~/templates/\
baremetal-nodes.yaml
```

5. 更正任何错误并保存节点定义模板。

6. 将节点定义模板导入到 RHOSP director，从模板将每个节点注册到 director：

示例

```
$ openstack overcloud node import ~/baremetal-nodes.yaml
```

验证

- 节点注册和配置完成后，确认 director 已成功注册节点：

```
$ openstack baremetal node list
```

baremetal node list 命令应包含导入的节点，并且状态应该可以 **管理**。

后续步骤

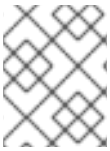
- 继续执行下一步，在 overcloud 上检查裸机节点。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 注册节点。
- *命令行界面参考中的* overcloud 节点导入

4.8. 内省 OVERCLOUD 上的裸机节点

将物理机注册为裸机节点后，您可以使用 OpenStack Platform (RHOSP) director 内省来自动添加节点的硬件详情，并为其每个以太网 MAC 地址创建端口。在裸机节点上执行内省后，最后一步是置备它们。



注意

如果您使用预置备节点，您可以跳过内省和内省裸机节点，并前往 部署启用了 spine-leaf 的 overcloud。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。
- 您已使用 RHOSP 为 overcloud 注册了裸机节点。

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. 运行 pre-introspection 验证组来检查内省要求：

```
$ validation run --group pre-introspection
```

4. 查看验证报告的结果。
5. （可选）查看特定验证的详细输出：

```
$ validation history get --full <UUID>
```

将 <UUID> 替换为您要查看的报告中特定验证的 UUID。

重要

FAILED 验证不会阻止您部署或运行 RHOSP。但是，**FAILED** 验证可能会显示生产环境中潜在的问题。

6. 检查所有节点的硬件属性：

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

提示

如需更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform 指南* 中的使用 director 内省来收集裸机节点硬件信息。

在一个单独的终端窗口中监控内省进度日志：

```
$ sudo tail -f /var/log/containers/ironic-inspector/ironic-inspector.log
```

验证

- 内省完成后，所有节点都会变为 available 状态。

后续步骤

- 继续执行下一步，为 overcloud 置备裸机节点。

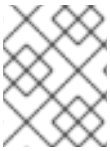
其他资源

- 使用 director 内省来收集裸机节点硬件信息，请参阅 *安装和管理 Red Hat OpenStack Platform* 指南
- *命令行界面参考*中的overcloud 节点内省

4.9. 为 OVERCLOUD 置备裸机节点

要为 Red Hat OpenStack Platform (RHOSP)置备裸机节点，您可以定义您要部署和为这些节点部署并分配 overcloud 角色的裸机节点的数量和属性。您还定义节点的网络布局。您可以使用 YAML 格式在节点定义文件中添加所有这些信息。

置备过程会从节点定义文件创建一个 heat 环境文件。此 heat 环境文件包含您在节点定义文件中配置的节点规格，包括节点数、预测节点放置、自定义镜像和自定义 NIC。部署 overcloud 时，请将此 heat 环境文件包含在部署命令中。置备过程还会为每个节点定义的所有网络置备端口资源，或者在节点定义文件中角色。



注意

如果您使用预置备节点，您可以跳过置备裸机节点，并进入 部署启用了 spine-leaf 的 overcloud。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。
- 裸机节点已注册、内省并可用于调配和部署。

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. 创建裸机节点定义文件，如 **spine-leaf-baremetal-nodes.yaml**，并为您要置备的每个角色定义节点数。

示例

```
- name: Controller
  count: 3
  defaults:
    networks:
```

```
    - network: ctlplane
      vif: true
    - 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: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
      default_route_network:
      - external
- name: Compute0
  count: 1
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
      subnet: tenant_subnet02
    - network: storage
      subnet: storage_subnet02
    network_config:
      template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
- name: Compute1
...
```

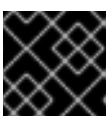
提示

有关您可以设置裸机节点定义文件的属性的更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 置备裸机节点。

4. 使用 **overcloud node provision** 命令置备 overcloud 裸机节点。

示例

```
$ openstack overcloud node provision \
 --stack spine_leaf_overcloud \
 --network-config \
 --output spine-leaf-baremetal-nodes-provisioned.yaml \
 /home/stack/templates/spine-leaf-baremetal-nodes.yaml
```



重要

您指定的输出文件的名称必须以 **.yaml** 或 **.template** 结尾。

5. 在一个单独的终端中监控置备进度。当置备成功时，节点状态将从 **available** 变为 **active** ：

```
$ watch openstack baremetal node list
```

6. 使用 **metalsmith** 工具获取节点的统一视图，包括分配和端口：

```
$ metalsmith list
```

7. 请注意生成的输出文件的路径和文件名。部署 overcloud 时，需要此路径。

验证

- 确认节点与主机名关联：

```
$ openstack baremetal allocation list
```

后续步骤

- 继续执行下一步，部署启用了 spine-leaf 的 overcloud。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 为 overcloud 置备裸机节点。

4.10. 部署启用了 SPINE-LEAF 的 OVERCLOUD

部署 Red Hat OpenStack Platform (RHOSP) overcloud 的最后一步是运行 **overcloud deploy** 命令。命令的输入包括您构建的所有 overcloud 模板和环境文件。RHOSP director 使用这些模板和文件，作为如何安装和配置 overcloud 的计划。

先决条件

- 访问 **stack** 用户的 undercloud 主机和凭据。
- 您已执行了本节前面流程中列出的所有步骤，并编译了所有各种 heat 模板和环境文件，以用作 **overcloud deploy** 命令的输入。

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. 更正 overcloud 环境所需的自定义环境文件和自定义模板。此列表包含 director 安装提供的未编辑 heat 模板文件，以及您创建的自定义文件。确定您有到以下文件的路径：
   - 您的自定义网络定义文件包含 overcloud 上 spine-leaf 网络的规格，如 **spine-leaf-networks-data.yaml**。
     如需更多信息，请参阅 定义叶网络。
   - 您的自定义角色数据文件，该文件为每个叶定义角色。
     示例： **spine-leaf-roles.yaml**。

如需更多信息，请参阅 定义叶角色和附加网络

- 包含每个角色的角色和自定义 NIC 模板映射的自定义环境文件。
  示例： **spine-leaf-nic-roles-map.yaml**。

  如需更多信息，请参阅为叶角色创建自定义 NIC 配置。

- 包含单独的网络映射的自定义网络环境文件，并为 overcloud 设置对 control plane 网络的访问。
  示例： **spine-leaf-ctlplane.yaml**

  如需更多信息，请参阅配置叶网络。

- 置备 overcloud 网络的输出文件。
  示例： **spine-leaf-networks-provisioned.yaml**

  有关更多信息，请参阅为 overcloud 置备网络和 VIP。

- 置备 overcloud VIP 的输出文件。
  示例： **spine-leaf-vips-provisioned.yaml**

  有关更多信息，请参阅为 overcloud 置备网络和 VIP。

- 如果您不使用预置备节点，则置备裸机节点的输出文件。
  示例： **spine-leaf-baremetal-nodes-provisioned.yaml**。

  有关更多信息，请参阅为 overcloud 置备裸机节点。

- 任何其它自定义环境文件。

4. 通过仔细排序为命令输入的自定义环境文件和自定义模板，输入 **overcloud deploy** 命令。
   常规规则是首先指定任何未编辑的 heat 模板文件，后跟包含自定义配置的自定义环境文件和自定义模板，如覆盖默认属性。

   按照以下顺序列出 **overcloud deploy** 命令的输入：

   a. 包含您的自定义环境文件，其中包含映射到每个角色的自定义 NIC 模板。
      示例：在 **network-environment.yaml** 后 **spine-leaf-nic-roles-map.yaml**。

      network-environment.yaml 文件为可组合网络参数提供默认网络配置，您的映射文件覆盖。请注意，director 从 **network-environment.j2.yaml** Jinja2 模板呈现此文件。

   b. 如果您创建了任何其他 spine leaf network 环境文件，请在 roles-NIC 模板映射文件后包含这些环境文件。

   c. 添加任何其他环境文件。例如，包含容器镜像位置或 Ceph 集群配置的环境文件。

      示例

      示例 **overcloud deploy** 命令摘录显示了命令输入的正确排序：

```
$ openstack overcloud deploy --templates \
  -n /home/stack/templates/spine-leaf-networks-data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml
\
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/frr.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ovn-bgp-
```

```
agent.yaml \
  -e /home/stack/templates/spine-leaf-nic-roles-map.yaml \
  -e /home/stack/templates/spine-leaf-ctlplane.yaml \
  -e /home/stack/templates/spine-leaf-baremetal-provisioned.yaml \
  -e /home/stack/templates/spine-leaf-networks-provisioned.yaml \
  -e /home/stack/templates/spine-leaf-vips-provisioned.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /home/stack/inject-trust-anchor-hiera.yaml \
  -r /home/stack/templates/spine-leaf-roles-data.yaml
  ...
```

提示

如需更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的创建 overcloud。

5. 运行 **overcloud deploy** 命令。
完成 overcloud 创建后，RHOSP director 会提供帮助您访问 overcloud 的详细信息。

验证

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的执行验证 overcloud 部署 中的步骤。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的创建 overcloud
- *命令行界面参考*中的 overcloud 部署

4.11. 向 SPINE-LEAF 部署中添加一个新的 LEAF

在增加网络容量或添加新的物理站点时，您可能需要向 Red Hat OpenStack Platform (RHOSP) spine-leaf 网络添加新的叶。

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. 打开您的网络定义模板，例如 **/home/stack/templates/spine-leaf-networks-data.yaml**。在适当的基本网络中，添加一个叶子网作为您要添加的新叶的可组合网络项。

示例

在本例中，添加了新的 leaf (**leaf3**)的子网条目：

```
- name: InternalApi
  name_lower: internal_api
  vip: true
  vlan: 10
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
  gateway_ip: '172.18.0.1'
  subnets:
    internal_api_leaf1:
      vlan: 11
      ip_subnet: '172.18.1.0/24'
      allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
      gateway_ip: '172.18.1.1'
    internal_api_leaf2:
      vlan: 12
      ip_subnet: '172.18.2.0/24'
      allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]
      gateway_ip: '172.18.2.1'
    internal_api_leaf3:
      vlan: 13
      ip_subnet: '172.18.3.0/24'
      allocation_pools: [{'start': '172.18.3.4', 'end': '172.18.3.250'}]
      gateway_ip: '172.18.3.1'
```

4. 为您要添加的新 leaf 创建一个角色数据文件。

   a. 为您要添加的新叶叶复制 leaf Compute 和 leaf Ceph Storage 文件。

      示例

      在本例中，**Compute1.yaml** 和 **CephStorage1.yaml** 分别被复制用于新的 leaf, **Compute3.yaml** 和 **CephStorage3.yaml** ：

      ```
      $ cp ~/roles/Compute1.yaml ~/roles/Compute3.yaml
      $ cp ~/roles/CephStorage1.yaml ~/roles/CephStorage3.yaml
      ```

   b. 编辑新 leaf 文件中的 **name** 和 **HostnameFormatDefault** 参数，以便它们与对应的 leaf 参数保持一致。

      示例

      例如，Leaf 1 Compute 文件中的参数具有以下值：

      ```
      - name: ComputeLeaf1
        HostnameFormatDefault: '%stackname%-compute-leaf1-%index%'
      ```

      示例

      Leaf 1 Ceph Storage 参数具有以下值：

      ```
      - name: CephStorageLeaf1
        HostnameFormatDefault: '%stackname%-cephstorage-leaf1-%index%'
      ```

   c. 编辑新 leaf 文件中的 network 参数，以便它们与相应的 Leaf 网络参数保持一致。

      示例

例如，Leaf 1 Compute 文件中的参数具有以下值：

```
- name: ComputeLeaf1
  networks:
    InternalApi:
      subnet: internal_api_leaf1
    Tenant:
      subnet: tenant_leaf1
    Storage:
      subnet: storage_leaf1
```

示例

Leaf 1 Ceph Storage 参数具有以下值：

```
- name: CephStorageLeaf1
  networks:
    Storage:
      subnet: storage_leaf1
    StorageMgmt:
      subnet: storage_mgmt_leaf1
```

d. 角色配置完成后，运行以下命令来生成完整的角色数据文件。在您的网络中包含所有叶叶以及您要添加的新叶叶。

示例

在本例中，leaf3 添加到 leaf0, leaf1, 和 leaf2 中：

```
$ openstack overcloud roles generate --roles-path ~/roles -o roles_data_spine_leaf.yaml
Controller Controller1 Controller2 Compute Compute1 Compute2 Compute3
CephStorage CephStorage1 CephStorage2 CephStorage3
```

这会创建一个完整的 **roles_data_spine_leaf.yaml** 文件，其中包含每个对应叶网络的所有自定义角色。

5. 为要添加的叶创建一个自定义 NIC 配置。

a. 为您要添加的新叶型复制 leaf Compute 和 leaf Ceph Storage NIC 配置文件。

示例

在本例中，**computeleaf1.yaml** 和 **ceph-storageleaf1.yaml** 会分别复制到新的 leaf, **computeleaf3.yaml** 和 **ceph-storageleaf3.yaml** ：

```
$ cp ~/templates/spine-leaf-nics/computeleaf1.yaml ~/templates/spine-leaf-
nics/computeleaf3.yaml
$ cp ~/templates/spine-leaf-nics/ceph-storageleaf1.yaml ~/templates/spine-leaf-
nics/ceph-storageleaf3.yaml
```

6. 打开自定义环境文件，其中包含每个角色的角色和自定义 NIC 模板映射，例如 spine-leaf-nic-roles-map.yaml。为您要添加的新 leaf 的每个角色插入一个条目。

```
parameter_defaults:
  %%ROLE%%NetworkConfigTemplate: <path_to_ansible_jinja2_nic_config_file>
```

示例

在本例中，添加了 **ComputeLeaf3NetworkConfigTemplate** 和 **CephStorage3NetworkConfigTemplate** 条目：

```
parameter_defaults:
  Controller0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  Controller1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  Controller2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  ComputeLeaf3NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage0NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage1NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage2NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
  CephStorage3NetworkConfigTemplate: '/home/stack/templates/spine-leaf-nics/single-nic-vlans.j2'
```

7. 打开自定义网络环境文件，其中包含单独的网络映射，并设置 overcloud 的 control plane 网络的访问权限，如 **spine-leaf-ctlplane.yaml** 并更新 control plane 参数。
在 **parameter_defaults** 部分下，为新 leaf 网络添加 control plane 子网映射。此外，也包含新叶网络的外部网络映射。

   - 对于扁平网络映射，列出 **NeutronFlatNetworks** 参数中的新 leaf (**leaf3**)，并为新的叶设置 **NeutronBridgeMappings** 参数：

     ```
     parameter_defaults:
       NeutronFlatNetworks: leaf0,leaf1,leaf2,leaf3
       Controller0Parameters:
         NeutronBridgeMappings: "leaf0:br-ex"
       Compute0Parameters:
         NeutronBridgeMappings: "leaf0:br-ex"
       Compute1Parameters:
         NeutronBridgeMappings: "leaf1:br-ex"
       Compute2Parameters:
         NeutronBridgeMappings: "leaf2:br-ex"
       Compute3Parameters:
         NeutronBridgeMappings: "leaf3:br-ex"
     ```

   - 对于 VLAN 网络映射，还要将 **NeutronNetworkVLANRanges** 设置为映射新叶(**leaf3**)网络的 VLAN：

     ```
     NeutronNetworkType: 'geneve,vlan'
     NeutronNetworkVLANRanges: 'leaf0:1:1000,leaf1:1:1000,leaf2:1:1000,leaf3:1:1000'
     ```

示例

在本例中，使用了扁平网络映射，并且添加了新的叶(**leaf3**)条目：

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2,leaf3
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  ControllerControlPlaneSubnet: leaf0
  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  Controller1ControlPlaneSubnet: leaf0
  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  Controller2ControlPlaneSubnet: leaf0
  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  Compute0ControlPlaneSubnet: leaf0
  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"
  Compute1ControlPlaneSubnet: leaf1
  Compute2Parameters:
    NeutronBridgeMappings: "leaf2:br-ex"
  Compute2ControlPlaneSubnet: leaf2
  Compute3Parameters:
    NeutronBridgeMappings: "leaf3:br-ex"
  Compute3ControlPlaneSubnet: leaf3
```

8. 置备您修改的网络。
   如需更多信息，请参阅置备 overcloud 网络和 overcloud VIP。

9. 在之前创建的裸机节点定义文件中，例如 **spine-leaf-baremetal-nodes.yaml**，请确保 **network_config_update** 变量设为 **true**。

   示例

```
- name: Controller
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - 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: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
```

```
    default_route_network:
    - external
  network_config_update: true
```

10. 置备修改后的节点。
    如需更多信息，请参阅 置备裸机节点。

11. 按照 部署 spine-leaf enabled overcloud 中的步骤重新部署您的 spine-leaf enabled overcloud 。