Red Hat OpenStack Platform 16.1

Spine Leaf Networking

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

Last Updated: 2023-11-07

Red Hat OpenStack Platform 16.1 Spine Leaf Networking

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。

使用直接文档反馈(DDF)功能

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

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

第 1 章 简介

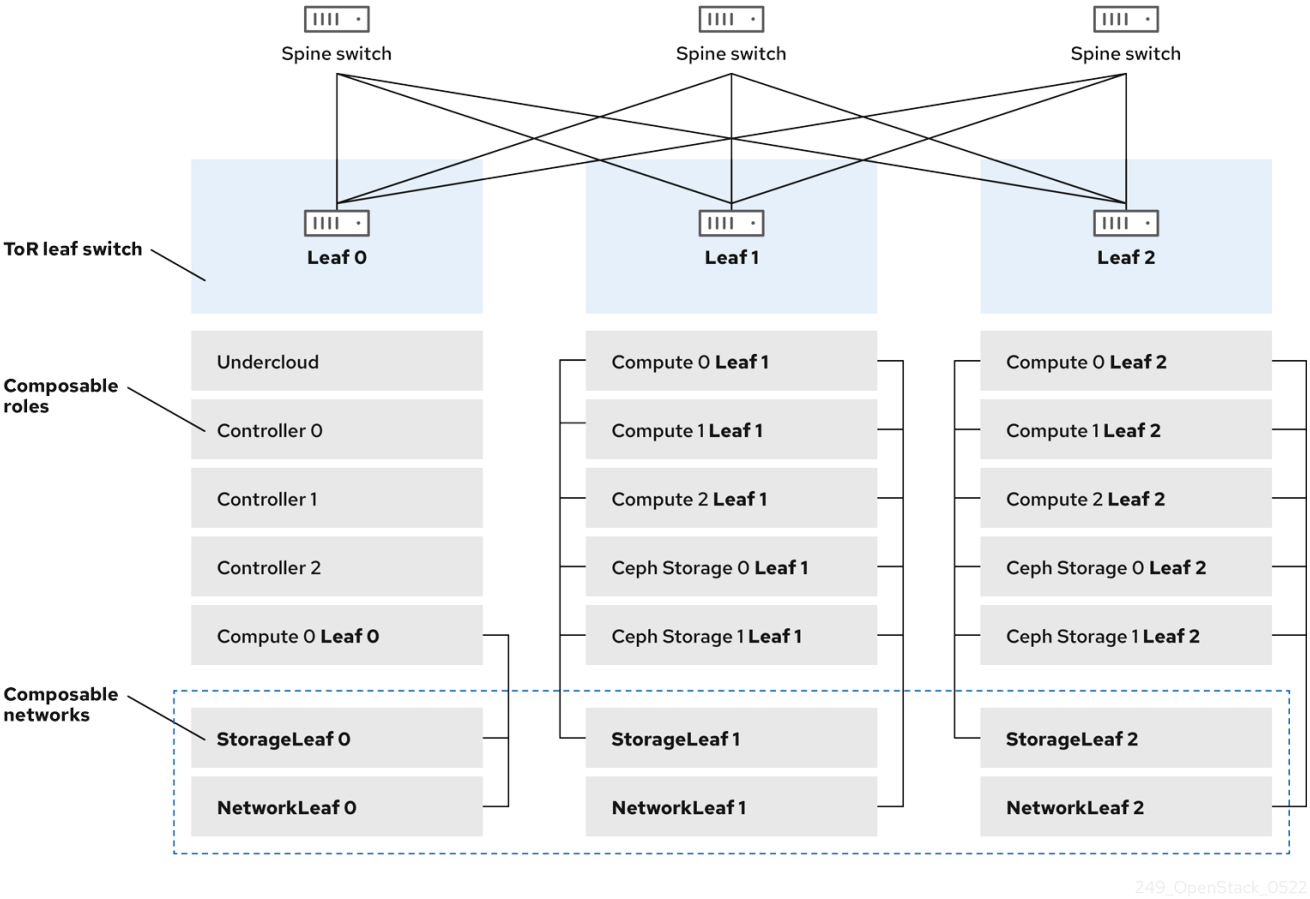
本指南提供有关为您的 Red Hat OpenStack Platform 环境构建 spine-leaf 网络拓扑的信息。这包括完整的端到端场景和示例文件，以帮助在您自己的环境中复制更广泛的网络拓扑。

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。

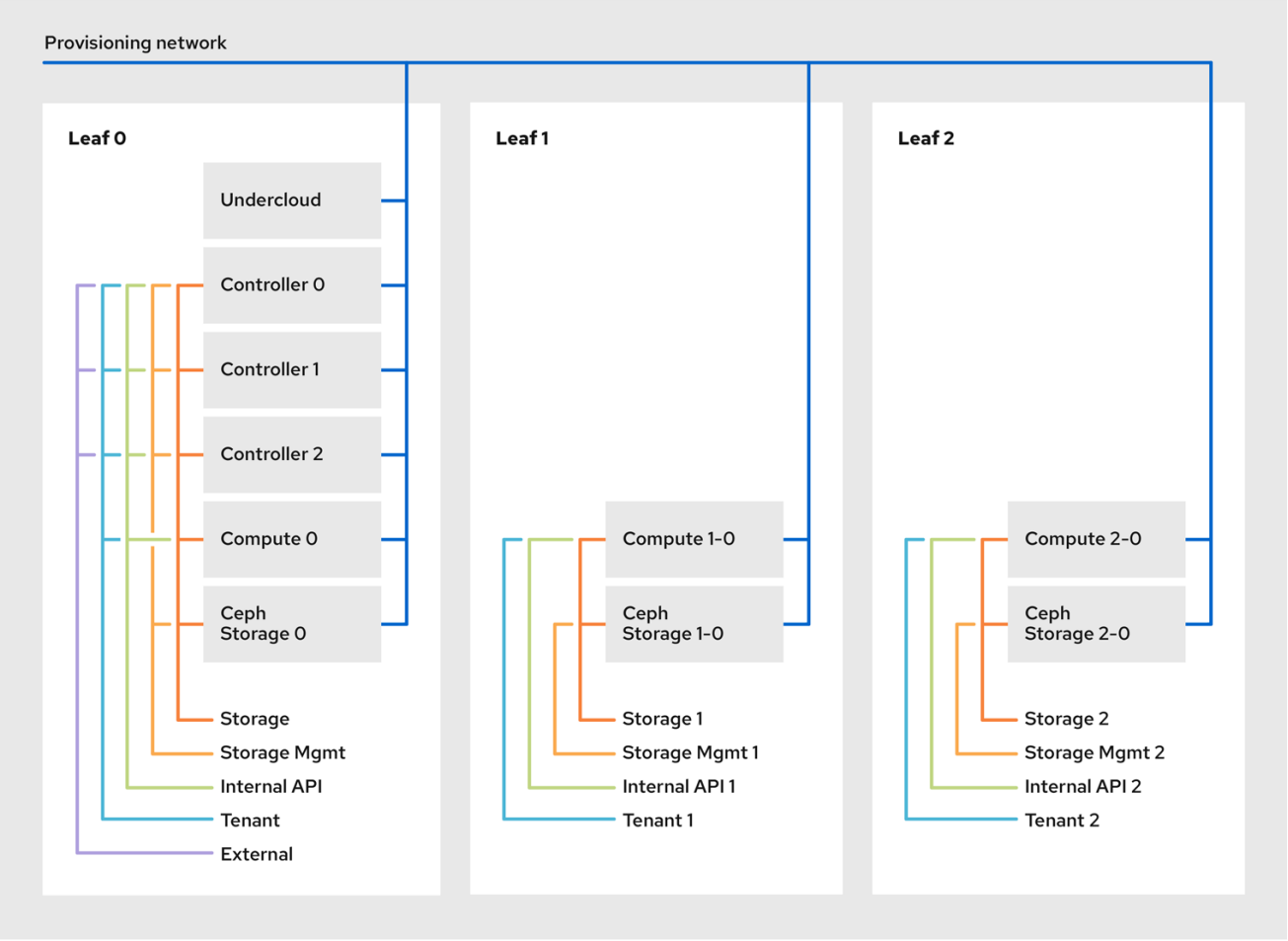
表 1.3. leaf 2 Networks

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：

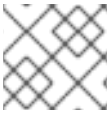
```
[stack@director ~]$ openstack undercloud install
```

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. 为叶网络创建类别和标记节点

每个叶网络中的每个角色都需要一个类别和角色分配，以便您可以将节点标记为对应的 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. 为每个自定义角色创建类别：

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

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

3. 检索节点列表来识别它们的 UUID：

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

4. 使用自定义资源类将每个裸机节点标记为 leaf network 和 role：

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

将 **<node>** 替换为裸机节点的 ID。

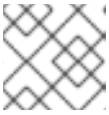
例如，输入以下命令将 UUID **58c3d07e-24f2-48a7-bbb6-6843f0e8ee13** 的节点标记为 Leaf2 上的 Compute 角色：

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

5. 将每个叶网络角色类别与自定义资源类关联：

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

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



注意

类别只能请求一个裸机资源类实例。

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

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

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 **enroll** 或 **manageable** 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. 提供 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

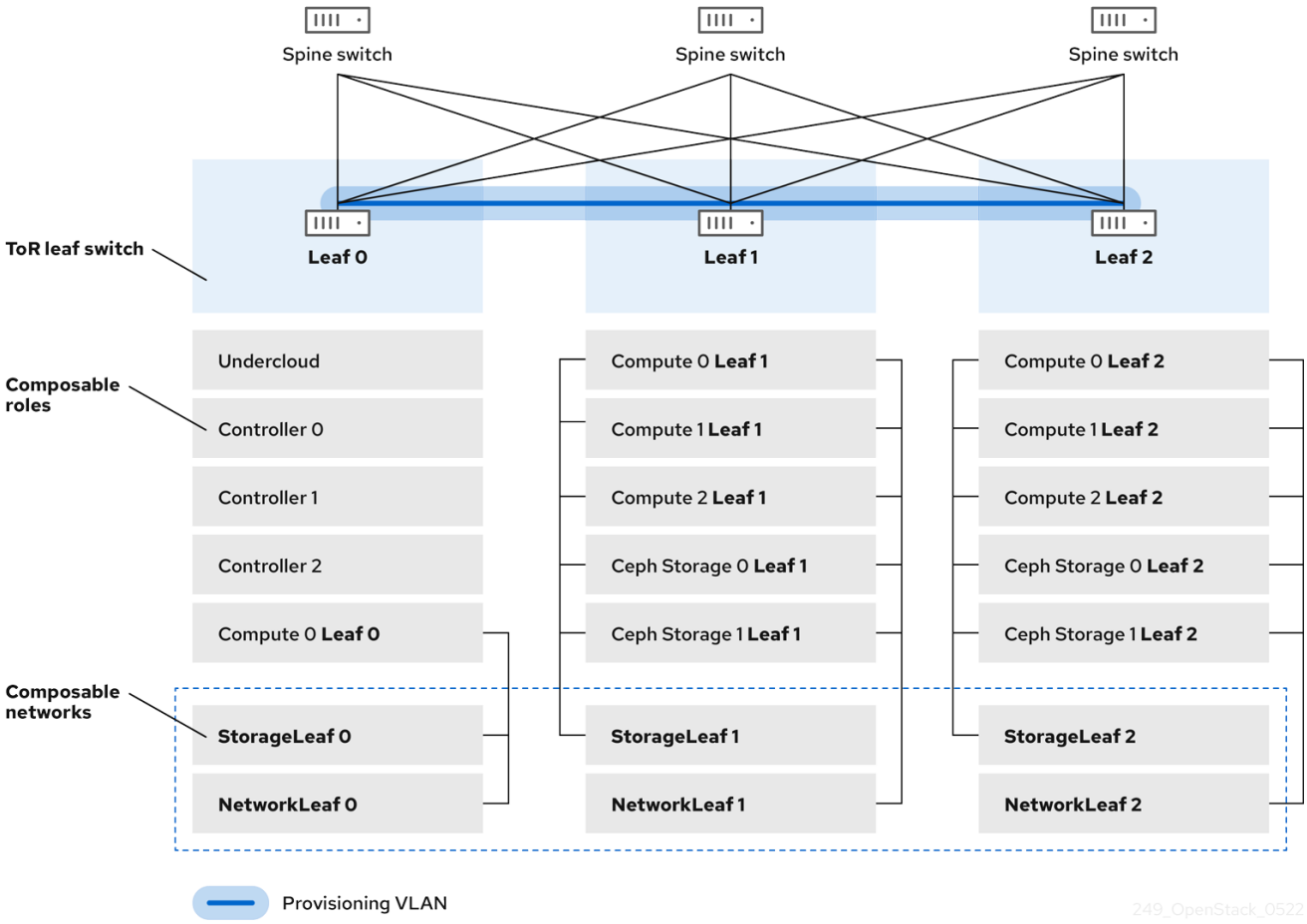
第 3 章 其他置备网络方法

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。

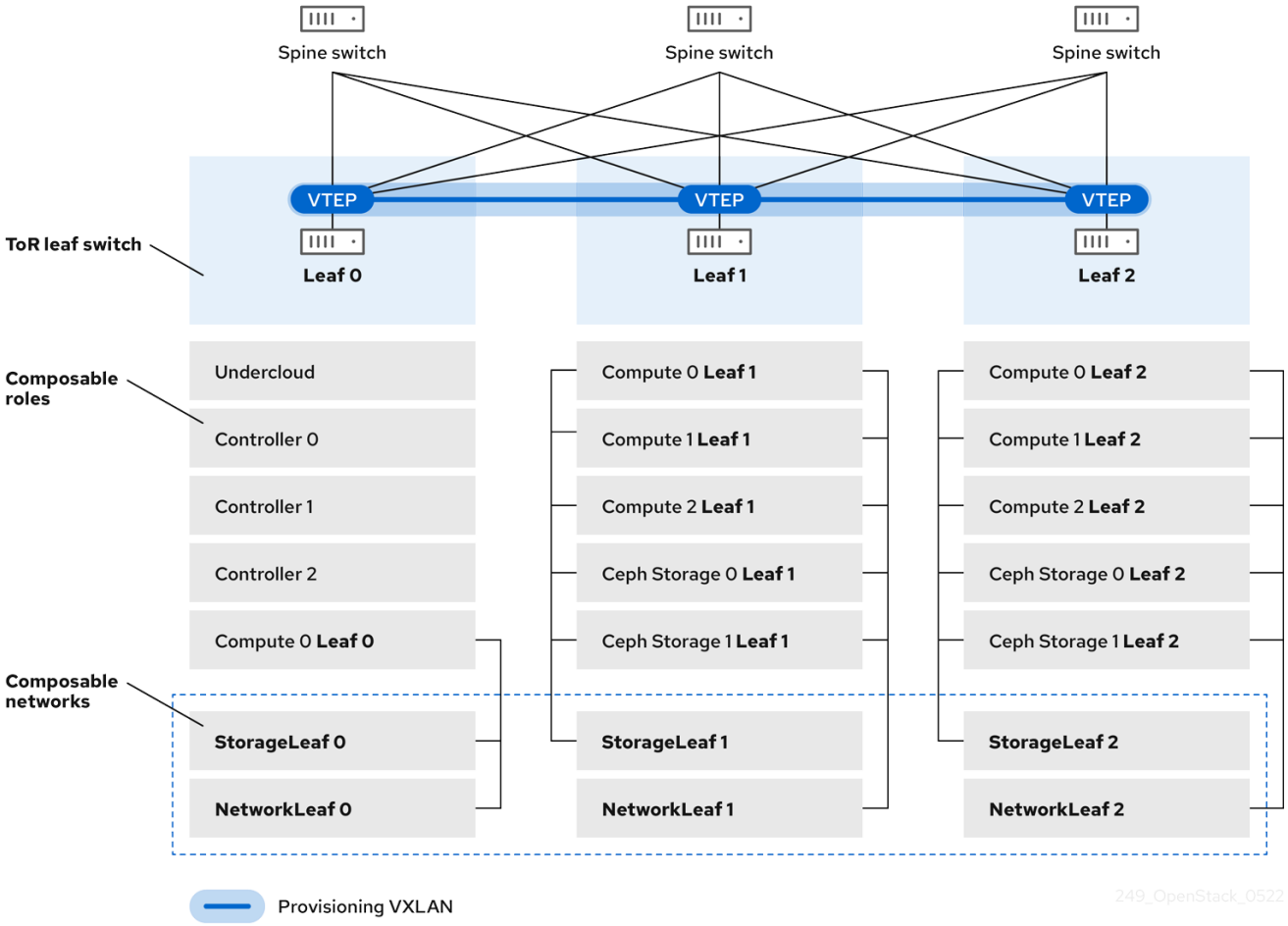
图 3.1. VLAN 置备网络拓扑



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

配置 undercloud 后，您可以使用一系列配置文件配置剩余的 overcloud 叶网络。配置剩余的 overcloud 叶网络并部署 overcloud 后，生成的部署具有多个可用的网络集合。

4.1. 创建网络数据文件

要定义叶网络，请创建一个网络数据文件，其中包含每个可组合网络及其属性的 YAML 格式列表。使用 **subnets** 参数定义额外的 Leaf 子网。

流程

1. 在 **stack** 用户的主目录中创建一个新的 **network_data_spine_leaf.yaml** 文件。使用默认 **network_data_subnets_routed.yaml** 文件作为基础：

```
$ cp /usr/share/openstack-tripleo-heat-templates/network_data_subnets_routed.yaml
/home/stack/network_data_spine_leaf.yaml
```

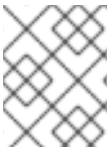
2. 在 **network_data_spine_leaf.yaml** 文件中，编辑 YAML 列表，将每个基本网络和对应的 leaf 子网定义为可组合网络项。使用以下示例语法定义基础 leaf 和两个 leaf 子网：

```
- name: <base_name>
  name_lower: <lowercase_name>
  vip: <true/false>
  vlan: '<vlan_id>'
  ip_subnet: '<network_address>/<prefix>'
  allocation_pools: [{'start': '<start_address>', 'end': '<end_address>'}]
  gateway_ip: '<router_ip_address>'
  subnets:
    <leaf_subnet_name>:
      vlan: '<vlan_id>'
      ip_subnet: '<network_address>/<prefix>'
      allocation_pools: [{'start': '<start_address>', 'end': '<end_address>'}]
      gateway_ip: '<router_ip_address>'
    <leaf_subnet_name>:
      vlan: '<vlan_id>'
      ip_subnet: '<network_address>/<prefix>'
      allocation_pools: [{'start': '<start_address>', 'end': '<end_address>'}]
      gateway_ip: '<router_ip_address>'
```

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
  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'
```



注意

您不会在网络数据文件中定义 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。

注意

为包含基于 Controller 的服务的网络定义 **vip: true**。在本例中，**InternalApiLeaf0** 包含这些服务。

4.2. 创建角色数据文件

要为每个叶定义每个可组合角色，并将可组合网络附加到每个对应的角色，请完成以下步骤。

流程

1. 在 **stack** 用户的主目录中创建自定义角色目录：

```
$ mkdir ~/roles
```

2. **将默认的 Controller、Compute 和 Ceph Storage 角色从 director 核心模板集合复制到 roles 目录。重命名 Compute 和 Ceph Storage 的文件以适合 Leaf 0 ：**

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Controller.yaml ~/roles/Controller.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
```

3. **复制 Leaf 0 Compute 和 Ceph Storage 文件作为您的 Leaf 1 和 Leaf 2 文件的基础：**

```
$ 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
```

4. **编辑 Leaf 0, Leaf 1, 和 Leaf 2 文件中的 name, HostnameFormatDefault, and deprecated_nic_config_name 参数，以便它们与对应的 Leaf 参数一致。例如，Leaf 0 Compute 文件中的参数具有以下值：**

```
- name: ComputeLeaf0
  HostnameFormatDefault: '%stackname%-compute-leaf0-%index%'
  deprecated_nic_config_name: 'computeleaf0.yaml'
```

Leaf 0 Ceph Storage 参数具有以下值：

```
- name: CephStorageLeaf0
  HostnameFormatDefault: '%stackname%-cephstorage-leaf0-%index%'
  deprecated_nic_config_name: 'ceph-strorageleaf0.yaml'
```

5. **编辑 Leaf 1 和 Leaf 2 文件中的 network 参数，以便它们与相应的 Leaf 网络参数保持一致。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
```

Leaf 1 Ceph Storage 参数具有以下值：

```
- 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再加上一个 _subnet 后缀。例如，Leaf 0 的内部 API 是 internal_api_subnet。

6. **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：**

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

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。

每个角色都有自己的 NIC 配置。在配置 spine-leaf 配置前，您必须创建一个基本 NIC 模板集合来适合您的当前 NIC 配置。

4.3. 创建自定义 NIC 配置

每个角色都需要一个唯一的 NIC 配置。完成以下步骤以创建基本 NIC 模板集合的副本，并将新模板映射到相应的 NIC 配置资源。

流程

1. 进入核心 heat 模板目录：

```
$ cd /usr/share/openstack-tripleo-heat-templates
```

2. 使用 tools/process-templates.py 脚本、自定义 network_data 文件和自定义 roles_data 文件呈现 Jinja2 模板：

```
$ tools/process-templates.py \
    -n /home/stack/network_data_spine_leaf.yaml \
    -r /home/stack/roles_data_spine_leaf.yaml \
    -o /home/stack/openstack-tripleo-heat-templates-spine-leaf
```

3. 进入主目录：

```
$ cd /home/stack
```

4. 从其中一个默认 NIC 模板复制内容，以用作您的 spine-leaf 模板的基础。例如，复制 single-nic-vlans NIC 模板：

```
$ cp -r openstack-tripleo-heat-templates-spine-leaf/network/config/single-nic-vlans/*
/home/stack/templates/spine-leaf-nics/.
```

5. 编辑 /home/stack/templates/spine-leaf-nics/ 中的每个 NIC 配置，并将配置脚本的位置更改为绝对位置。滚动到网络配置部分，类似于以下片断：

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: ../../scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
```

将脚本的位置改为绝对路径：

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-
config.sh
          params:
            $network_config:
              network_config:
```

在每个文件中为每个 Leaf 进行这个更改并保存更改。

注意

有关进一步的 NIC 更改，*请参阅高级 Overcloud 自定义指南中的自定义网络接口模板*。https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/16.1/html/advanced_overcloud_customization/assembly_custom-network-interface-templates

6. **创建名为 spine-leaf-nics.yaml 的文件，并编辑该文件。**

7. **在文件中创建一个 resource_registry 部分，并添加一组映射到相应 NIC 模板的 ::Net::SoftwareConfig 资源：**

```
resource_registry:
```

```
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/controller.yaml
  OS::TripleO::ComputeLeaf0::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf0.yaml
  OS::TripleO::ComputeLeaf1::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf1.yaml
  OS::TripleO::ComputeLeaf2::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf2.yaml
  OS::TripleO::CephStorageLeaf0::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf0.yaml
  OS::TripleO::CephStorageLeaf1::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf1.yaml
  OS::TripleO::CephStorageLeaf2::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf2.yaml
```

这些资源映射会在部署过程中覆盖默认资源映射。

8. 保存 **spine-leaf-nics.yaml** 文件。

9. 删除渲染的模板目录：

```
$ rm -rf openstack-tripleo-heat-templates-spine-leaf
```

因此，您现在有一组 **NIC** 模板和一个环境文件，它将所需的 **::Net::SoftwareConfig** 资源映射到它们。

10. 最终运行 **openstack overcloud deploy** 命令时，请确保按以下顺序包含环境文件：

    a. **/usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml**，它可启用网络隔离。请注意，**director** 从 **network-isolation.j2.yaml Jinja2** 模板呈现此文件。

    b. **/usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml**，它是默认的网络环境文件，包括默认的 **NIC** 资源映射。请注意，**director** 从 **network-environment.j2.yaml Jinja2** 模板呈现此文件。

    c. **/home/stack/templates/spine-leaf-nics.yaml**，其中包含您的自定义 **NIC** 资源映射并覆盖默认 **NIC** 资源映射。

以下命令片断演示了排序：

```
$ openstack overcloud deploy --templates
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-
environment.yaml \
  -e /home/stack/templates/spine-leaf-nics.yaml \
  ...
```

11. 完成以下部分中的步骤，在网络环境文件中添加详情，并定义 spine leaf 架构的某些方面。完成此配置后，请将此文件包含在 openstack overcloud deploy 命令中。

其他资源

- *高级 Overcloud 自定义指南中的自定义网络接口模板* https://access.redhat.com/documentation/zh-cn/red_hat_openstack_platform/16.1/html/advanced_overcloud_customization/assembly_custom-network-interface-templates

4.4. 设置 CONTROL PLANE 参数

您通常使用 network_data 文件为隔离的 spine-leaf 网络定义网络详情。例外是 undercloud 创建的 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。

在本例中，在 Leaf 0 上为相应 Control Plane 网络定义 IP、子网和默认路由。

流程

1. 创建名为 spine-leaf-ctlplane.yaml 的文件，并编辑该文件。

2. 在文件中创建一个 parameter_defaults 部分，并为每个 spine-leaf 网络添加 control plane 子网映射：

```
parameter_defaults:
  ...
  ControllerControlPlaneSubnet: leaf0
  Compute0ControlPlaneSubnet: leaf0
  Compute1ControlPlaneSubnet: leaf1
  Compute2ControlPlaneSubnet: leaf2
```

```
CephStorage0ControlPlaneSubnet: leaf0
CephStorage1ControlPlaneSubnet: leaf1
CephStorage2ControlPlaneSubnet: leaf2
```

3. 保存 **spine-leaf-ctlplane.yaml** 文件。

4.5. 为虚拟 IP 地址设置子网

Controller 角色通常为每个网络托管虚拟 IP (VIP)地址。默认情况下，overcloud 从每个网络的基本子网获取 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。

在这种 spine leaf 场景中，默认基础调配网络为 leaf0，而不是 ctlplane-subnet。这意味着，您必须在 VipSubnetMap 参数中添加覆盖值，以更改 control plane VIP 使用的子网。

另外，如果每个网络的 VIP 没有使用一个或多个网络的基本子网，您必须将额外的覆盖添加到 VipSubnetMap 参数，以确保 director 在连接到 Controller 节点的 L2 网络段的子网中创建了 VIP。

流程：

1. 创建名为 **spine-leaf-vips.yaml** 的文件，并编辑该文件。

2. 在文件中创建一个 **parameter_defaults** 部分，并根据您的要求添加 **VipSubnetMap** 参数：

   - 如果将 **leaf0** 用于 **provisioning / control plane** 网络，请将 **ctlplane VIP** 重新映射设置为 **leaf0** ：

     ```
     parameter_defaults:
       VipSubnetMap:
         ctlplane: leaf0
     ```

   - 如果您将不同的 **Leaf** 用于多个 **VIP**，请设置 **VIP** 重新映射以满足这些要求。例如，使用以下代码片段将 **VipSubnetMap** 参数配置为对所有 **VIP** 使用 **leaf1** ：

     ```
     parameter_defaults:
       VipSubnetMap:
         ctlplane: leaf1
         redis: internal_api_leaf1
     ```

```
  InternalApi: internal_api_leaf1
  Storage: storage_leaf1
  StorageMgmt: storage_mgmt_leaf1
```

3. 保存 **spine-leaf-vips.yaml** 文件。

4.6. 映射单独的网络

默认情况下，OpenStack Platform 使用 Open Virtual Network (OVN)，这需要所有 Controller 和 Compute 节点都连接到单个 L2 网络来访问外部网络。这意味着，控制器和计算网络配置都使用 br-ex 网桥，director 默认映射到 overcloud 中的 datacentre 网络。此映射通常是扁平网络映射或 VLAN 网络映射。在 spine leaf architecture 中，您可以更改这些映射，以便每个 Leaf 通过该 Leaf 上的特定网桥或 VLAN 路由流量，这通常是边缘计算场景的情况。

流程

1. 创建名为 **spine-leaf-separate.yaml** 的文件，并编辑该文件。

2. 在 **spine-leaf-separate.yaml** 文件中创建一个 **parameter_defaults** 部分，并为每个 spine-leaf 网络包含外部网络映射：

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

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

   - 对于 VLAN 网络映射，还要将 **NeutronNetworkVLANRanges** 设置为为所有三个 Leaf 网络映射 VLAN：

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

3. 保存 spine-leaf-separate.yaml 文件。

4.7. 部署启用了 SPINE-LEAF 的 OVERCLOUD

完成 spine-leaf overcloud 配置后，完成以下步骤来查看每个文件，然后运行部署命令：

流程

1. 检查 /home/stack/template/network_data_spine_leaf.yaml 文件，并确保它包含每个叶的每个网络和子网。

注意

目前，网络子网和 allocation_pools 值没有自动验证。确保以统一方式定义这些值，并且现有网络没有冲突。

2. 查看 /home/stack/templates/roles_data_spine_leaf.yaml 值，并确保为每个叶定义角色。

3. 查看 ~/templates/spine-leaf-nics/ 目录中的 NIC 模板，并确保正确为每个叶角色定义接口。

4. 查看自定义 spine-leaf-nics.yaml 环境文件，并确保它包含引用每个角色自定义 NIC 模板的 resource_registry 部分。

5. 检查 /home/stack/templates/nodes_data.yaml 文件，并确保所有角色都有分配的类别和节点数。另外，也请检查您是否已正确标记每个叶的所有节点。

6. 运行 openstack overcloud deploy 命令以应用 spine leaf 配置。例如：

```
$ openstack overcloud deploy --templates \
   -n /home/stack/templates/network_data_spine_leaf.yaml \
   -r /home/stack/templates/roles_data_spine_leaf.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
   -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml \
   -e /home/stack/templates/spine-leaf-nics.yaml \
   -e /home/stack/templates/spine-leaf-ctlplane.yaml \
   -e /home/stack/templates/spine-leaf-vips.yaml \
```

```
  -e /home/stack/templates/spine-leaf-separate.yaml \
  -e /home/stack/templates/nodes_data.yaml \
  -e [OTHER ENVIRONMENT FILES]
```

- **network-isolation.yaml 是 Jinja2 文件在同一位置(network-isolation.j2.yaml)的呈现名称。在部署命令中包括此文件，以确保 director 将每个网络隔离到正确的叶位置。这样可确保在 overcloud 创建过程中动态创建网络。**

- **在 network-isolation.yaml 后面包括 network-environment.yaml 文件。network-environment.yaml 文件为可组合网络参数提供默认网络配置。**

- **在 network-environment.yaml 后面包括 spine-leaf-nics.yaml 文件。spine-leaf-nics.yaml 文件会覆盖来自 network-environment.yaml 文件的默认 NIC 模板映射。**

- **如果您创建了任何其他 spine leaf network 环境文件，请在 spine-leaf-nics.yaml 文件后包括这些环境文件。**

- **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。**

7. **等待启用了 spine-leaf 的 overcloud 部署。**

4.8. 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，您可能需要在 Red Hat OpenStack Platform (RHOSP) spine-leaf 网络中有一个新的叶型。

前提条件

- **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. **提供 undercloud 凭证文件：**

-

```
$ source ~/stackrc
```

3. **在 /usr/share/openstack-tripleo-heat-templates/network_data_spine_leaf.yaml 文件中，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。**

Example

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。**

Example

在本例中，Compute1.yaml 和 CephStorage1.yaml 分别从新 leaf, Compute3.yaml 和 CephStorage3.yaml 复制。

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

b.

编辑新 leaf 文件中的 name,HostnameFormatDefault, 和 deprecated_nic_config_name 参数，以便它们与相应的 Leaf 参数保持一致。

Example

例如，Leaf 1 Compute 文件中的参数具有以下值：

```
- name: ComputeLeaf1
  HostnameFormatDefault: '%stackname%-compute-leaf1-%index%'
  deprecated_nic_config_name: 'computeleaf1.yaml'
```

示例

Leaf 1 Ceph Storage 参数具有以下值：

```
- name: CephStorageLeaf1
  HostnameFormatDefault: '%stackname%-cephstorage-leaf1-%index%'
  deprecated_nic_config_name: 'ceph-strorageleaf1.yaml'
```

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 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。

Example

在本例中，computeleaf1.yaml 和 ceph-storageleaf1.yaml 复制到新的 leaf, computeleaf3.yaml 和 ceph-storageleaf3.yaml，代表：

```
$ 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
```

b.

在 /usr/share/openstack-tripleo-heat-templates/network_data_spine_leaf.yaml 中，在文件中的 resource_registry 部分下，添加一组 ::Net::SoftwareConfig 资源，映射到对应的 NIC 模板：

Example

在本例中，添加了新的 leaf NIC 配置文件(computeleaf3.yaml 和 ceph-storageleaf3.yaml)：

```
resource_registry:
  OS::TripleO::Controller::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/controller.yaml
  OS::TripleO::ComputeLeaf0::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf0.yaml
  OS::TripleO::ComputeLeaf1::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf1.yaml
  OS::TripleO::ComputeLeaf2::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf2.yaml
  OS::TripleO::ComputeLeaf3::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/computeleaf3.yaml
  OS::TripleO::CephStorageLeaf0::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf0.yaml
  OS::TripleO::CephStorageLeaf1::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf1.yaml
  OS::TripleO::CephStorageLeaf2::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf2.yaml
  OS::TripleO::CephStorageLeaf3::Net::SoftwareConfig: /home/stack/templates/spine-leaf-nics/ceph-storageleaf3.yaml
```

这些资源映射会在部署过程中覆盖默认资源映射。

因此，您现在有一组 NIC 模板和一个环境文件，它将所需的 ::Net::SoftwareConfig 资源映射到它们。最终运行 openstack overcloud deploy 命令时，请确保按以下顺序包含环境文件：

c. /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml，它可启用网络隔离。

请注意，director 从 network-isolation.j2.yaml Jinja2 模板呈现此文件。

d. /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml，它是默认的网络环境文件，包括默认的 NIC 资源映射。

请注意，director 从 network-environment.j2.yaml Jinja2 模板呈现此文件。

e. /home/stack/templates/spine-leaf-nics.yaml，其中包含您的自定义 NIC 资源映射并覆盖默认 NIC 资源映射。

以下命令片断演示了排序：

```
$ openstack overcloud deploy --templates
```

```
    ...
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/network-
environment.yaml \
    -e /home/stack/templates/spine-leaf-nics.yaml \
    ...
```

6. **更新 control plane 参数。**

在 ~/templates/spine-leaf-ctlplane.yaml 中，在 parameter_defaults 部分下为新的 leaf 网络添加 control plane 子网映射：

Example

在本例中，添加了新的 leaf (leaf3)条目：

```
parameter_defaults:
  ...
  ControllerControlPlaneSubnet: leaf0
  Compute0ControlPlaneSubnet: leaf0
  Compute1ControlPlaneSubnet: leaf1
  Compute2ControlPlaneSubnet: leaf2
  Compute3ControlPlaneSubnet: leaf3
  CephStorage0ControlPlaneSubnet: leaf0
  CephStorage1ControlPlaneSubnet: leaf1
  CephStorage2ControlPlaneSubnet: leaf2
  CephStorage3ControlPlaneSubnet: leaf3
```

7. **映射新的叶网络。**

在 ~/templates/spine-leaf-separate.yaml 中，在 parameter_defaults 部分下，包括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'
```

8. 按照 第 4.7 节 “部署启用了 spine-leaf 的 overcloud” 中的步骤重新部署启用了 spine-leaf 的 overcloud。

其他资源

- 将新的 leaf 添加到 spine-leaf provisioning 网络