Red Hat OpenStack Platform 17.1

在 Red Hat OpenStack Platform 中配置动态路由

使用 director 配置 FRRouting 和 OVN BGP 代理，以便在 Red Hat OpenStack Platform 中实现动态路由

Last Updated: 2024-05-23

Red Hat OpenStack Platform 17.1 在 Red Hat OpenStack Platform 中配置动态路由

使用 director 配置 FRRouting 和 OVN BGP 代理，以便在 Red Hat OpenStack Platform 中实现动态路由

OpenStack Team
rhos-docs@redhat.com

法律通告

摘要

在 Red Hat OpenStack Platform 中安装、配置、操作和故障排除 BGP 动态路由的指南。

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 RHOSP 动态路由简介

Red Hat OpenStack Platform (RHOSP)支持使用边框网关协议(BGP)的动态路由。

本节中包含的主题有：

- 关于 RHOSP 动态路由
- RHOSP 动态路由中使用的 BGP 组件
- BGP 广告和流量重定向

1.1. 关于 RHOSP 动态路由

Red Hat OpenStack Platform (RHOSP)支持使用 control 和 data plane 中的边框网关协议(BGP)的 ML2/OVN 动态路由。在纯层 3 (L3)数据中心中部署集群解决了传统第 2 层(L2)基础架构的扩展问题，如大型故障域、高卷广播流量或故障恢复期间的长期聚合时间。

RHOSP 动态路由提供了一种负载平衡和高可用性的机制，它与大多数互联网服务提供商采用的传统方法不同。使用 RHOSP 动态路由，您可以使用 Controller 节点上的共享 VIP 的 L3 路由来提高高可用性。在跨可用区部署的 control plane 服务器上，您可以维护独立的 L2 片段、物理站点和电源电路。

使用 RHOSP 动态路由，您可以在创建和启动期间或与浮动 IP 地址关联时为虚拟机和负载均衡器公开 IP 地址。当设置了特殊标志时，项目网络上提供相同的功能。

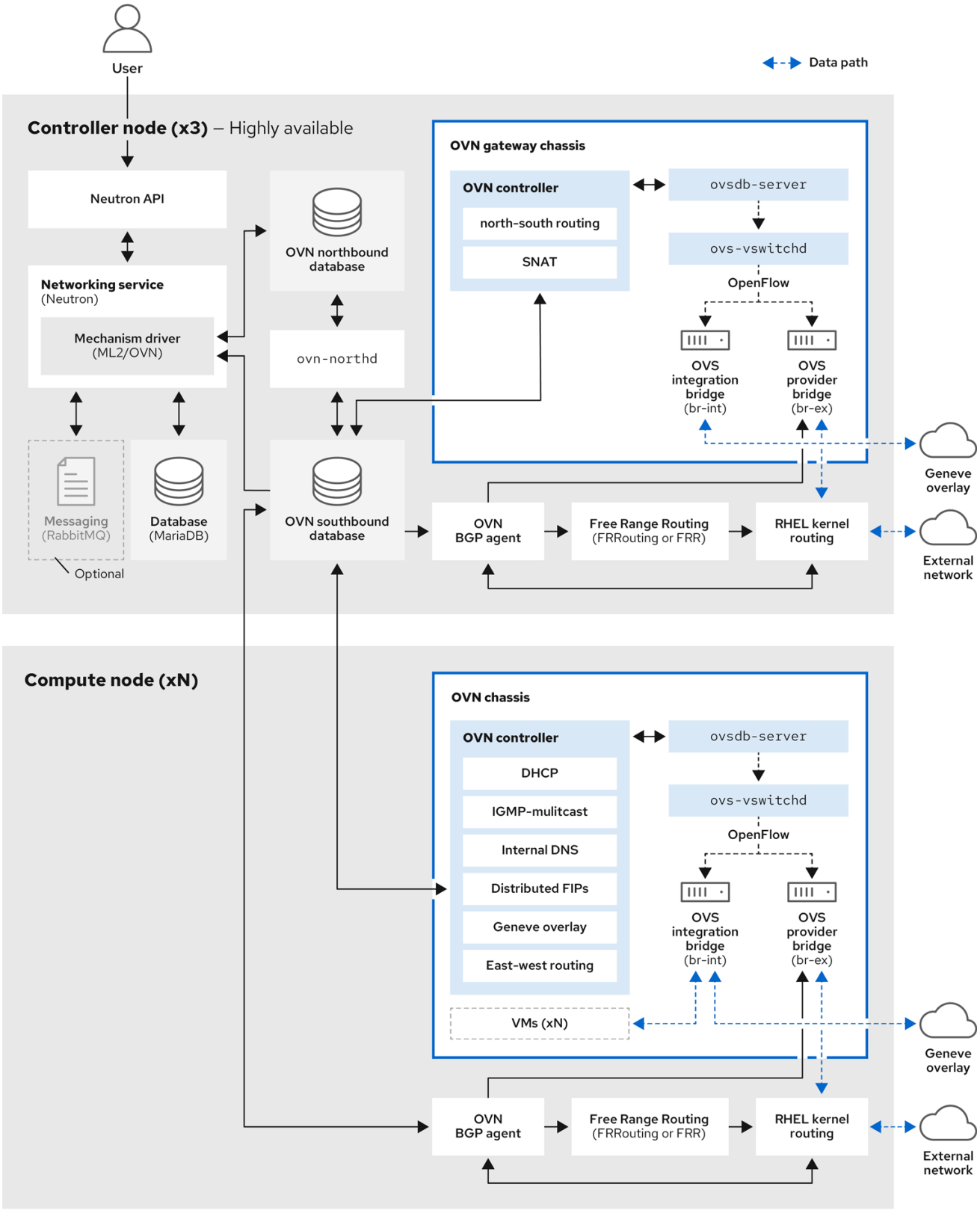
RHOSP 动态路由还提供以下优点：

- 改进了数据平面流量的管理。
- 简化了角色之间差异的配置。
- 跨 L3 边界的分布式 L2 提供程序 VLAN 和浮动 IP (FIP)子网不需要跨越机架之间的 VLAN，仅用于非重叠的 CIDR。
- 数据中心和边缘站点中的机架和 L3 边界之间的分布式控制器。
- 将整个子网从一个站点故障转移到另一个站点，以进行公共提供程序 IP 或 FIP。
- 下一代数据中心和超大规模结构支持。

1.2. RHOSP 动态路由中使用的 BGP 组件

Red Hat OpenStack Platform (RHOSP)依赖于几个组件来提供到第 3 层数据中心的动态路由。

图 1.1. RHOSP 动态路由组件



OVN BGP 代理(ovn-bgp-agent 容器)

在每个 RHOSP Controller 和 Compute 节点上的 **ovn-controller** 容器中运行的基于 Python 的守护进程。代理监控特定虚拟机和浮动 IP (FIP)事件的 Open Virtual Network (OVN)南向数据库。当发生这些事件时，代理会通知 FRR BGP 守护进程 (**bgpd**) 公告与虚拟机关联的 IP 地址或 FIP。代理还触发将

外部流量路由到 OVN 覆盖的操作。由于代理使用多驱动程序实现，您可以为在 OVN 上运行的特定基础架构（如 RHOSP 或 Red Hat OpenShift Platform）配置代理。

自由范围路由(FRRouting 或 FRR) (frr 容器)

在所有可组合角色上的 **frr** 容器中运行的守护进程的 IP 路由协议套件，并一起构建路由表。FRR 支持 equal-cost 多路径路由(ECMP)，但每个协议守护进程使用不同的方法来管理 ECMP 策略。由 Red Hat Enterprise Linux (RHEL)提供的 FRR 套件提供多个守护进程。RHOSP 主要使用 FRR **bgpd**、**bfdd** 和 Zebra 守护进程。

BGP 守护进程(frr 容器)

在 **frr** 容器中运行的守护进程(**bgpd**)，以实施边框网关协议(BGP)的版本 4。**bgpd** 守护进程使用能力协商来检测远程对等的功能。如果对等点专门配置为 IPv4 单播邻居，则 **bgpd** 不会发送功能协商数据包。BGP 守护进程通过 Zebra 守护进程与内核路由表通信。

BFD 守护进程(frr 容器)

实现双向转发检测(BFD)的 FRR 套件中的守护进程(**bfdd**)。此协议提供相邻转发引擎之间的故障检测。

zebra 守护进程(frr 容器)

一个守护进程，协调各种 FRR 守护进程的信息，并将路由决策直接发送到内核路由表。

VTY shell (frr 容器)

VTY shell (**vtysh**)的 shell 聚合了每个守护进程中定义的所有 CLI 命令，并在单个 shell 中显示它们。

其他资源

- FRRouting 文档

1.3. BGP 广告和流量重定向

在使用 Red Hat OpenStack Platform (RHOSP)动态路由的部署中，使用公告的路由到虚拟机、负载均衡器和浮动 IP (FIP)的网络流量流。在流量到达节点后，OVN BGP 代理添加 IP 规则、路由和 OVS 流规则，以使用 Red Hat Enterprise Linux (RHEL)内核网络将流量重定向到 OVS 提供程序网桥(**br-ex**)。

广告网络路由的过程从 OVN BGP 代理触发自由范围路由(FRRouting 或 FRR)来公告和撤回直接连接的路由。OVN BGP 代理执行这些步骤来正确配置 FRR，以确保每当 IP 地址添加到 **bgp-nic** 接口时公告它们：

1. FRR 启动 VTY shell 以连接到 FRR 套接字：

```
$ vtysh --vty_socket -c <command_file>
```

2. VTY shell 通过包含以下内容的文件：

```
LEAK_VRF_TEMPLATE = '''
router bgp {{ bgp_as }}
  address-family ipv4 unicast
    import vrf {{ vrf_name }}
  exit-address-family

  address-family ipv6 unicast
    import vrf {{ vrf_name }}
  exit-address-family

router bgp {{ bgp_as }} vrf {{ vrf_name }}
  bgp router-id {{ bgp_router_id }}
```

```
 address-family ipv4 unicast
  redistribute connected
 exit-address-family

 address-family ipv6 unicast
  redistribute connected
 exit-address-family
...
```

VTY shell 通过的命令，执行以下操作：

a. 默认情况下，创建一个名为 **bgp_vrf** 的 VRF。

b. 将 dummy 接口类型与 VRF 关联。
默认情况下，dummy 接口命名为 **bgp-nic**。

c. 向 OVS 提供程序网桥添加 IP 地址，以确保启用了地址解析协议(ARP)和邻居发现协议(NDP)。

3. Zebra 守护进程监控虚拟机和负载均衡器的 IP 地址，因为它们在任何本地接口上添加和删除，Zebra 公告和撤回路由。
因为 FRR 被配置为启用 **重新分发连接** 的选项，因此广告和撤回路由由从 dummy 接口 **bgp-nic** 公开或删除路由组成。

注意

默认情况下，禁用连接到租户网络的虚拟机。如果在 RHOSP 配置中启用了它，OVN BGP 代理会公开 neutron 路由器网关端口。OVN BGP 代理通过托管 **机箱重定向** 逻辑路由器端口(**CR-LRP**)的节点将流到租户网络上虚拟机的流量注入到 OVN 覆盖中。

4. FRR 在托管虚拟机或负载均衡器或包含 OVN 路由器网关端口的节点上公开 IP 地址。

OVN BGP 代理执行必要的配置，以使用 RHEL 内核网络和 OVS 将流量重定向到 OVN 覆盖，然后 FRR 在正确的节点上公开 IP 地址。

当 OVN BGP 代理启动时，它会执行以下操作：

1. 添加 IP 地址到 OVS 提供程序网桥，以启用 ARP 和 NDP。

2. 向 **/etc/iproute2/rt_tables** 中每个 OVS 提供程序网桥的路由表中添加一个条目。

注意

在 RHEL 内核中，路由表的最大数量为 252。这会将提供商网络数量限制为 252。

3. 将 VLAN 设备连接到网桥，并启用 ARP 和 NDP （仅限 VLAN 提供商网络）。

4. 清理 OVS 提供程序网桥上的任何额外的 OVS 流。

在常规重新同步事件或启动过程中，OVN BGP 代理执行以下操作：

1. 添加 IP 地址规则，以将特定路由应用到路由表。
在以下示例中，此规则与 OVS 提供程序网桥关联：

```
$ ip rule

0:      from all lookup local
1000:   from all lookup [l3mdev-table]
*32000:  from all to IP lookup br-ex*  # br-ex is the OVS provider bridge
*32000:  from all to CIDR lookup br-ex*  # for VMs in tenant networks
32766:  from all lookup main
32767:  from all lookup default
```

2. 添加 IP 地址路由到 OVS 提供程序网桥路由表，将流量路由到 OVS 提供程序网桥设备：

```
$ ip route show table br-ex

default dev br-ex scope link
*CIDR via CR-LRP_IP dev br-ex*  # for VMs in tenant networks
*CR-LRP_IP dev br-ex scope link*  # for the VM in tenant network redirection
*IP dev br-ex scope link*  # IPs on provider or FIPs
```

3. 根据是否使用 IPv4 或 IPv6，通过 OVS 提供程序网桥(**br-ex**)将流量路由到 OVN：

   a. 对于 IPv4，为 OVN 路由器网关端口 **CR-LRP** 添加静态 ARP 条目，因为 OVN 不会响应 L2 网络外部的 ARP 请求：

   ```
   $ ip nei

   ...
   CR-LRP_IP dev br-ex lladdr CR-LRP_MAC PERMANENT
   ...
   ```

   b. 对于 IPv6，添加 NDP 代理：

   ```
   $ ip -6 nei add proxy CR-LRP_IP dev br-ex
   ```

4. 通过在 OVS 提供程序网桥上添加新流，将来自 OVN 覆盖的流量发送到内核网络，使目标 MAC 地址更改为 OVS 提供程序网桥的 MAC 地址 (**actions=mod_dl_dst:OVN_PROVIDER_BRIDGE_MAC,NORMAL**)：

```
$ sudo ovs-ofctl dump-flows br-ex

cookie=0x3e7, duration=77.949s, table=0, n_packets=0, n_bytes=0, priority=
900,ip,in_port="patch-provnet-1" actions=mod_dl_dst:3a:f7:e9:54:e8:4d,NORMAL
cookie=0x3e7, duration=77.937s, table=0, n_packets=0, n_bytes=0, priority=
900,ipv6,in_port="patch-provnet-1" actions=mod_dl_dst:3a:f7:e9:54:e8:4d,NORMAL
```

第 2 章 规划使用 RHOSP 动态路由的部署

当您计划在 Red Hat OpenStack Platform 环境中实施动态路由时，评估支持的功能、依赖项、限制和必要的网络拓扑。

本节中包含的主题有：

- RHOSP 动态路由支持列表
- RHOSP 动态路由的要求
- RHOSP 动态路由的限制
- RHOSP 动态路由拓扑示例

2.1. RHOSP 动态路由支持列表

下表列出了 Red Hat OpenStack Platform (RHOSP) 17.1 中支持的动态路由功能。

注意

如果没有列出该功能，则 RHOSP 17.1 不支持该功能。

表 2.1. RHOSP 动态路由功能支持列表

功能	RHOSP 17.1 支持？
内核路由	是
IPv6	是
EVPN	否
OVS-DPDK 路由	否
SR-IOV 路由	否
重叠的 CIDR	否
选择浮动 IP 的暴露	否

2.2. RHOSP 动态路由的要求

Red Hat OpenStack Platform (RHOSP)的动态路由需要以下网络拓扑和软件：

- spine-leaf 网络拓扑。
- 使用以下内容运行 RHOSP 版本 17.0 或更高版本的环境：
  - ML2/OVN 机制驱动程序。

- 在回环接口上的 RHOSP undercloud 上配置边界网关协议(BGP)和 VIP。
- 在 RHOSP overcloud 上部署的 OVN BGP 代理。

- 与 BGP 对等的网络设备必须安装有 BGP 的实现。BGP 在大多数设备上是标准的，由设备厂商提供。

2.3. RHOSP 动态路由的限制

在 Red Hat OpenStack Platform (RHOSP)环境中计划动态路由时，请考虑以下限制：

- 您无法控制公开哪些虚拟机和负载均衡器(LB)。公开所有位于提供商网络或具有Floating IP 的虚拟机和 LB。另外，如果启用了 **expose_tenant_network** 标志，则项目网络中的虚拟机会公开。
- 您必须使用地址范围和子网池，因为不支持重叠 CIDR。
- BGP 通过使用内核路由通过 IP 路由和规则来窃取网络流量。因此，不支持不使用内核空间的 OVS-DPDK。
- 在 RHOSP 网络中，对于将负载均衡器成员连接到提供商网络的路由器，北向供应商上的 OVN-octavia VIP 的流量或与项目网络上的 VIP 关联的 FIP 必须经过托管 neutron 路由器网关的网络节点。

  注意

  这些节点上的端口被称为 **chassisredirect** 逻辑路由器端口(**cr-lrp**)。

  因此，进入 OVN 覆盖的入口点必须是这些网络节点之一，因此 VIP 和 FIP 会通过这些节点公开。从网络节点，流量遵循正常的隧道路径(Geneve 隧道)到所选成员所在的 RHOSP Compute 节点。

- 目前存在一个已知问题：转发到浮动 IP (FIP)地址上的端口会失败。相反，用于 FIP 的网络流量被转发到租户端口 IP 地址，该 IP 地址包含在配置为执行端口转发的端口列表中。此失败的原因是 OVN BGP 代理没有向 FIP 公开路由。目前，还没有临时解决方案。如需更多信息，请参阅 BZ 2160481。
- 目前，Red Hat OpenStack Platform Compute 服务无法路由发送到多播 IP 地址目的地的数据包。因此，订阅多播组的虚拟机实例无法接收发送到它们的数据包。其原因是 overcloud 节点上没有正确配置 BGP 多播路由。目前，还没有临时解决方案。如需更多信息，请参阅 BZ 2163477。
- 在两个 RHOSP 17.1 版本之间的更新过程中，会出现连接停机时间，因为每个节点上的 FRR 组件必须重启，并发生以下事件：

  1. 每个节点都使用其对等路由器重新建立 BGP 会话，这会影响 control plane 和数据平面流量。
  2. 重新建立 BGP 会话后，FRR 会获取到节点或从节点恢复路由，这意味着路由在几秒内不可用。
  3. 最后，**ovn-bgp-agent** 将 VRF 配置添加到运行的 FRR 服务中，以使用 BGP 公告数据平面流量的路由。

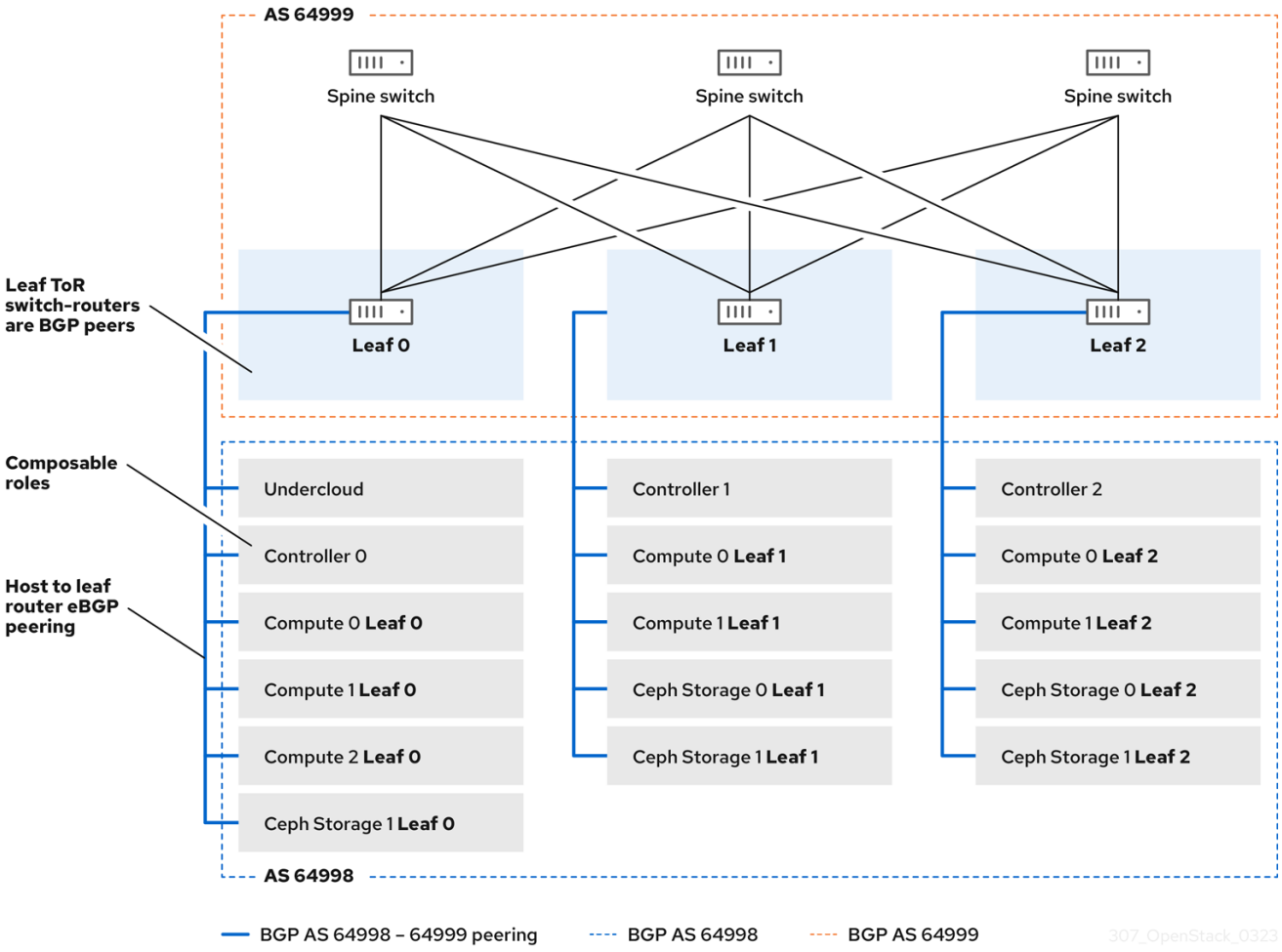
- 在 RHOSP 动态环境中添加存储节点时，必须使用 provisioning 网络。RHOSP director 需要在创建建立连接的路由器之前存在 Red Hat Ceph Storage。

2.4. RHOSP 动态路由拓扑示例

下图演示了在 Red Hat OpenStack Platform (RHOSP) ML2/OVN 环境中使用动态路由的网络拓扑示例。

此示例网络拓扑展示了一个由三个离开而成的。rack switch-routers 的顶部是 BGP 对等点，与 spine 上的交换机相关联。

图 2.1. overcloud 主机的对叶路由器的 BGP 对等



第 3 章 为 RHOSP 动态路由部署 UNDERCLOUD

undercloud 是控制最终 Red Hat OpenStack Platform (RHOSP) 环境（称为 overcloud）的配置、安装和管理的节点。undercloud 使用 OpenStack Platform 组件服务，包括在容器中运行的 OVN BGP 代理。这些容器化服务包括一个称为 RHOSP director 的工具，用于创建和管理 overcloud。

本节中包含的主题为：

- 为 RHOSP 动态路由安装和配置 undercloud

3.1. 为 RHOSP 动态路由安装和配置 UNDERCLOUD

您可以使用 Red Hat OpenStack Platform (RHOSP) director 在 RHOSP undercloud 中安装和配置动态路由。高级步骤有：

1. （可选）在 **frr-parameters.yaml** 中为 undercloud 设置 BGP 配置值。
2. 在 **undercloud.conf** 中为 undercloud 设置 spine-leaf 网络拓扑配置值。
3. 运行 **openstack undercloud install** 命令。

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. 如果您计划使用 BGP 访问其他机架和 overcloud 节点，请通过将以下参数添加到自定义 heat 环境文件 **/home/stack/templates/frr-parameters.yaml** 来配置 FRRouting (FRR)。

注意

请记住此路径。后续步骤中需要用到它。

示例

```
parameter_defaults:
  ContainerFrrImage: registry.redhat.io/rhosp-17.1/openstack-frr-rhel9:17.1.1
  FrrBfdEnabled: true
  FrrBgpEnabled: true
  FrrBgpAsn: 64999
  FrrBgpUplinks: ['nic2', 'nic3']
  FrrBgpUplinksScope: internal
  FrrLogLevel: debugging
  FrrBgpRouterID: 172.30.4.1
  FrrBgpIpv4SrcIp: 172.30.4.1
  FrrBgpIpv6SrcIp: fe80::5054:ff:fe74:73ce
```

提示

有关更多信息，请参阅 *Overcloud 参数指南*中的 Networking (neutron) 参数。

FrrBfdEnabled

为 **true** 时，启用双向转发检测(BFD)。默认值为 **false**。

FrrBgpEnabled

为 **true** 时，启用边框网关协议(BGP)。默认值是 **true**。

FrrBgpAsn

FRRouting 中使用的默认 ASN。默认值为 **65000**。**FrrBgpAsn** 可以设置为使用的每个角色的不同值。

FrrBgpUplinks

以逗号分隔的 uplink 网络接口列表。默认值为 **['nic1', 'nic2']**。

FrrBgpUplinksScope

与内部(iBGP)或外部(eBGP)邻居的对等点。默认值为 **internal**。

FrrLogLevel

使用以下值指定 FRR 日志级别：**emergencies**, **alerts**, **critical**, **errors**, **warnings**, **notifications**, **informational**, **debugging**.默认为 **informational**。

FrrBgpRouterID

FRR 要使用的 BGP **router_id**。

FrrBgpIpv4SrcIp

IPv4 网络流量的源 IP 地址。

FrrBgpIpv6SrcIp

IPv6 网络流量的源 IP 地址。

tripleo_frr_bgp_peers

用于指定特定于角色的参数，用来为 Free Range Routing (FRR)指定要对等的 IP 地址或主机名。

tripleo_frr_ovn_bgp_agent_enable

特定于角色的参数，用于在没有数据平面路由的 RHOSP 节点上启用或禁用 OVN BGP 代理。默认值为 **true**。

4. 如果您还没有 **undercloud.conf** 文件，请复制示例模板文件：

```
$ cp /usr/share/python-tripleoclient/undercloud.conf.sample \
~/templates/undercloud.conf
```

5. 在 **[DEFAULT]** 部分中，设置以下常规参数值：

示例

```
[DEFAULT]
# General
cleanup = false
container_images_file=/home/stack/templates/
\containers-prepare-parameter.yaml
overcloud_domain_name = {{ cloud_domain }}
undercloud_timezone = UTC
undercloud_hostname = undercloud-0.{{ cloud_domain }}

# BGP on undercloud
...
```

```
# TLS-e
...

# Networking
...

# Subnets
...
```

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 Undercloud 配置参数。

overcloud_domain_name

指定部署 overcloud 时要使用的 DNS 域名。在后续步骤中，您必须确保这个值与 overcloud **CloudDomain** 参数的值匹配。

cleanup

删除临时文件。把它设置为 **false** 以保留部署期间使用的临时文件。如果出现错误，临时文件可帮助您调试部署。

container_images_file

使用容器镜像信息指定 Heat 环境文件。

container_insecure_registries

供 **podman** 使用的不安全 registry 列表。如果您想从其他来源（如私有容器 registry）拉取镜像，则使用此参数。

custom_env_files

要添加到 undercloud 安装中的其他环境文件。

undercloud_hostname

定义 undercloud 的完全限定主机名。如果设置，undercloud 安装将配置所有系统主机名设置。如果保留未设置，undercloud 将使用当前的主机名，但您必须相应地配置所有主机名设置。

undercloud_timezone

undercloud 的主机时区。如果未指定时区，director 将使用现有时区配置。

6. 如果要在 undercloud 上安装 BGP，在 **[DEFAULT]** 部分中，在 undercloud 上启用 FRR，并指向您在前面步骤中设置 FRR 参数值的自定义环境文件。

示例

```
[DEFAULT]
# General
...

# BGP on undercloud
enable_frr=true
custom_env_files=/home/stack/templates/frr-parameters.yaml

# TLS-e
...
```

```
# Networking
...

# Subnets
...
```

7. 如果您使用 TLS-everywhere，然后在 **[DEFAULT]** 部分中设置以下 TLS-everywhere 参数值：

示例

```
[DEFAULT]
# General
...

# BGP on undercloud
...

# TLS-e
enable_novajoin = False
undercloud_nameservers = {{ freeipa_ip }}
generate_service_certificate = True
ipa_otp = {{ undercloud_otp }}

# Networking
...

# Subnets
...
```

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 Undercloud 配置参数。

enable_novajoin

为 **true** 时，启用 novajoin 服务来部署 TLS。

undercloud_nameservers

指定 undercloud 名称服务器的 DNS 服务器的当前 IP 地址。您可以在 **/etc/resolv.conf** 中找到此信息。

generate_service_certificate

定义 undercloud 安装期间是否生成 SSL/TLS 证书，此证书用于 **undercloud_service_certificate** 参数。

ipa_otp

设置 FreeIPA OTP 事实。

8. 在 **[DEFAULT]** 部分中，设置以下网络参数值：

示例

```
[DEFAULT]
# General
...
```

```
# BGP on undercloud
...

# TLS-e
...

# Networking
local_interface = eth0
local_ip = {{ undercloud_ctlplane }}/24
undercloud_public_host = {{ undercloud_public_host }}
undercloud_admin_host = {{ undercloud_admin_host }}

# Subnets
...
```

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 Undercloud 配置参数。

local_interface

用于本地网络的桥接接口。

local_ip

leaf0 上的 undercloud 的 IP 地址。

undercloud_public_host

用于外部的 undercloud 的 IP 地址。

undercloud_admin_host

undercloud 的管理 IP 地址。此 IP 地址通常位于 leaf0 上。

9. 为您之前在 **subnets** 参数中定义的每个子网创建一个新部分。

重要

director 在创建子网后无法更改子网的 IP 地址。

示例

```
[DEFAULT]
# General
...

# BGP on undercloud
...

# TLS-e
...

# Networking
...
```

```
# Subnets
[r1]
# This subnet is used for overcloud nodes deployed on rack1.
cidr = 192.168.1.0/24
dhcp_start = 192.168.1.150
dhcp_end = 192.168.1.170
inspection_iprange = 192.168.1.171,192.168.1.185
gateway = 192.168.1.1
masquerade = False
[r2]
# This subnet is used for overcloud nodes deployed on rack2.
cidr = 192.168.2.0/24
dhcp_start = 192.168.2.150
dhcp_end = 192.168.2.170
inspection_iprange = 192.168.2.171,192.168.2.185
gateway = 192.168.2.1
masquerade = False
[r3]
# This subnet is used for overcloud nodes deployed on rack3.
cidr = 192.168.3.0/24
dhcp_start = 192.168.3.150
dhcp_end = 192.168.3.170
inspection_iprange = 192.168.3.171,192.168.3.185
gateway = 192.168.3.1
masquerade = False
[r4]
# This subnet is used for the underloud node and potentially FreeIPA
# that are deployed on rack4.
cidr = 192.168.4.0/24
dhcp_start = {{ undercloud_dhcp_start }}
dhcp_end = 192.168.4.170
inspection_iprange = 192.168.4.171,192.168.4.185
gateway = 192.168.4.1
masquerade = False
```

提示

有关更多信息，*请参阅使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 子网。

cidr

director 用来管理 overcloud 实例的网络。这是 undercloud **neutron** 服务管理的 Provisioning 网络。保留其默认值 **192.168.24.0/24**，除非您需要 Provisioning 网络使用其他子网。

masquerade

定义是否伪装 **cidr** 中定义的用于外部访问的网络。这为 Provisioning 网络提供了网络地址转换(NAT)，使 Provisioning 网络能够通过 director 进行外部访问。

注意

director 配置还使用相关的 sysctl 内核参数自动启用 IP 转发。

dhcp_start 和 dhcp_end

overcloud 节点 DHCP 分配范围的开始值和终止值。请确保此范围可以为节点提供足够的 IP 地址。

dhcp_exclude

DHCP 分配范围中排除的 IP 地址。

dns_nameservers

特定于子网的 DNS 名称服务器。如果没有为子网定义名称服务器，子网将使用 **undercloud_nameservers** 参数中定义的名称服务器。

gateway

overcloud 实例的网关。它是 undercloud 主机，会把网络流量转发到外部网络。保留其默认值 **192.168.24.1**，除非您需要 director 使用其他 IP 地址，或想直接使用外部网关。

10. 运行 install 命令。

```
$ openstack undercloud install
```

11. 确保您的 undercloud 具有正确的网络配置，包括用于访问每个叶和机架的额外网络路由。如需更多信息，请参阅 *安装和管理 Red Hat OpenStack Platform 指南* 中的 Director 配置参数。

验证

1. director 配置脚本会自动启动所有服务。确认 RHOSP 服务容器正在运行：

```
$ sudo podman ps -a --format "{{.Names}} {{.Status}}"
```

输出示例

您应该看到类似如下的输出，这表示 RHOSP 服务容器为 **Up** ：

```
memcached Up 3 hours (healthy)
haproxy Up 3 hours
rabbitmq Up 3 hours (healthy)
mysql Up 3 hours (healthy)
iscsid Up 3 hours (healthy)
keystone Up 3 hours (healthy)
keystone_cron Up 3 hours (healthy)
neutron_api Up 3 hours (healthy)
logrotate_crond Up 3 hours (healthy)
neutron_dhcp Up 3 hours (healthy)
neutron_l3_agent Up 3 hours (healthy)
neutron_ovs_agent Up 3 hours (healthy)
ironic_api Up 3 hours (healthy)
ironic_conductor Up 3 hours (healthy)
ironic_neutron_agent Up 3 hours (healthy)
ironic_pxe_tftp Up 3 hours (healthy)
ironic_pxe_http Up 3 hours (unhealthy)
ironic_inspector Up 3 hours (healthy)
ironic_inspector_dnsmasq Up 3 hours (healthy)
neutron-dnsmasq-qdhcp-30d628e6-45e6-499d-8003-28c0bc066487 Up 3 hours
...
```

2. 确认您可以初始化 **stack** 用户以使用命令行工具：

```
$ source ~/stackrc
```

如果提示符显示 **(undercloud)**，这表示 OpenStack 命令对 undercloud 进行身份验证并执行：

输出示例

```
(undercloud) [stack@director ~]$
```

director 的安装已完成。您现在可以使用 director 命令行工具了。

其他资源

- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 undercloud 配置参数。
- *使用 director 安装和管理 Red Hat OpenStack Platform 中的* 子网
- *Overcloud 参数指南*中的 networking(neutron) 参数。
- *使用 director 安装和管理 Red Hat OpenStack Platform* 指南中的 部署命令选项

第 4 章 为 RHOSP 动态路由部署 OVERCLOUD

使用 Red Hat OpenStack Platform (RHOSP) director 在 overcloud 中安装和配置 RHOSP 动态路由。高级步骤有：

1. 为每个叶定义 overcloud 网络。
2. 创建一个可组合角色 - 包括每个叶的 Free Range Routing (FRR)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. 在 动态路由环境中部署 Ceph。
11. 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。

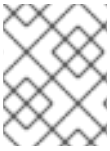
示例

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
```

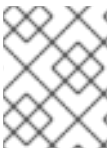
注意

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，请参阅为 RHOSP 动态路由部署 undercloud。



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/Controller0.yaml ~/roles/Controller1.yaml
$ cp ~/roles/Controller0.yaml ~/roles/Controller2.yaml
```

```
$ 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)控制器和计算网络配置使用 OVS 供应商网桥(**br-ex**)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。

您必须设置一些基本的 FRRouting 和 OVN BGP 代理配置。

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

- 您必须是有权访问 RHOSP undercloud 的 **stack** 用户。
- 已安装 undercloud。

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 OVS 提供程序网桥(**br-ex**)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。

示例

```
parameter_defaults:
  NeutronFlatNetworks: provider1
  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

提示

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

4. 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: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

```
ControllerRack3Parameters:
  NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

ComputeRack1Parameters:
  NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

ComputeRack2Parameters:
  NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

ComputeRack3Parameters:
  NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

注意

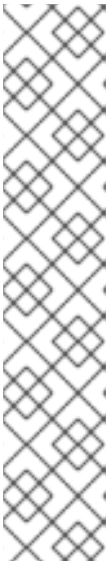
您可以在 spine-leaf topology 中使用扁平网络和 VLAN。

5. 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:
  NeutronNetworkType: 'geneve,vlan'
  NeutronNetworkVLANRanges: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r1

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r2

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r3

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

6. 为每个叶设置 OVN BGP 代理、FRRouting 和 CMS 选项。



注意

FRR 服务在所有 RHOSP 节点上运行，以提供在数据平面不同节点上运行的 control plane 和服务之间的连接。但是，您必须仅在所有 Compute 节点上运行 OVN BGP 代理，并在配置了 **enable-chassis-as-gw** 的节点上运行。

对于没有公开数据平面路由的 RHOSP 节点，请通过将 **tripleo_frr_ovn_bgp_agent_enable** 参数设置为 **false** 来禁用这些角色的 OVN BGP 代理。默认值是 **true**。

示例

```
parameter_defaults:
  DatabaseRack1ExtraGroupVars:
    tripleo_frr_ovn_bgp_agent_enable: false
```

示例

```
parameter_defaults:
  NeutronNetworkType: 'geneve,vlan'
  NeutronNetworkVLANRanges: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r1
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r2
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r3
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
```

提示

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

OVNCMSOptions

在 OVSDB 中配置的 CMS 选项。

FrrOvnBgpAgentReconcileInterval

定义检查状态的频率，以确保在正确的位置仅公开正确的 IP。默认： 120。

FrrOvnBgpAgentOvsdbConnection

原生 OVSDB 后端的连接字符串。使用 **tcp:<IP_address>:<port>** 进行 TCP 连接。默认：*tcp:127.0.0.1:6640*。

FrrOvnBgpAgentExposeTenantNetworks

通过 MP-BGP IPv4 和 IPv6 单播在租户网络上公开虚拟机 IP。需要 BGP 驱动程序（请参阅 THT 参数 FrrOvnBgpAgentDriver）。默认：false。

FrrOvnBgpAgentDriver

配置如何通过 BGP 公告虚拟机 IP。EVPN 驱动程序通过 MP-BGP IPv4 和 IPv6 单播公开提供商网络上的虚拟机 IP 和 FIP。BGP 驱动程序通过 MP-BGP EVPN VXLAN 在租户网络上公开虚拟机 IP。默认： **ovn_evpn_driver**。

FrrOvnBgpAgentAsn

在 BGP 模式中运行时，代理使用的自主系统编号(ASN)。默认：64999。**FrrOvnBgpAgentAsn** 可以设置为使用的每个角色的不同值。

FrrLogLevel

日志级别。默认： *信息性*。

FrrBgpAsn

FRR 中使用的默认 ASN。默认： 65000。**FrrBgpAsn** 可以设置为使用的每个角色的不同值。

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，需要此路径。
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

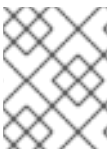
- 继续执行下一步，在 overcloud 上检查裸机节点。

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: ControllerRack1
  count: 1
  hostname_format: ctrl-1-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r1.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network1
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: ctrl-1-0
    name: ctrl-1-0
    capabilities:
      node: ctrl-1-0
```

```
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.1.2
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.2
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.1
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:f00d:0001
      subnet: main_network_ipv6_r1
- name: ComputeRack1
  count: 2
  hostname_format: cmp-1-%index%
  defaults:
    network_config:
      default_route_network:
        - ctlplane
      template: /home/stack/tht/nics_r1.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network1
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: cmp-1-0
    name: cmp-1-0
    capabilities:
      node: cmp-1-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.1.6
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.6
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.2
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:f00d:0004
      subnet: main_network_ipv6_r1
  - hostname: cmp-1-1
    name: cmp-1-1
    capabilities:
      node: cmp-1-1
    networks:
    - network: ctlplane
```

```
      vif: true
    - network: left_network
      fixed_ip: 100.65.1.10
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.10
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.3
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0005
      subnet: main_network_ipv6_r1
- name: ControllerRack2
  count: 1
  hostname_format: ctrl-2-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r2.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network2
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: ctrl-2-0
    name: ctrl-2-0
    capabilities:
      node: ctrl-2-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.2
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.2
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.1
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0002
      subnet: main_network_ipv6_r1
- name: ComputeRack2
  count: 2
  hostname_format: cmp-2-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r2.yaml
```

```
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
    - network: right_network2
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: cmp-2-0
    name: cmp-2-0
    capabilities:
      node: cmp-2-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.6
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.6
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.2
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0006
      subnet: main_network_ipv6_r1
  - hostname: cmp-2-1
    name: cmp-2-1
    capabilities:
      node: cmp-2-1
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.10
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.10
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.3
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0007
      subnet: main_network_ipv6_r1
- name: ControllerRack3
  count: 1
  hostname_format: ctrl-3-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r3.yaml
    networks:
    - network: ctlplane
```

```
      vif: true
    - network: left_network
    - network: right_network3
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: ctrl-3-0
    name: ctrl-3-0
    capabilities:
      node: ctrl-3-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.2
      subnet: left_network_r3
    - network: right_network3
      fixed_ip: 100.64.0.2
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.1
      subnet: main_network_r3
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0003
      subnet: main_network_ipv6_r1
- name: ComputeRack3
  count: 2
  hostname_format: cmp-3-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r3.yaml
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
    - network: right_network3
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: cmp-3-0
    name: cmp-3-0
    capabilities:
      node: cmp-3-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.6
      subnet: left_network_r3
    - network: right_network3
      fixed_ip: 100.64.0.6
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.2
```

```
      subnet: main_network_r3
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0008
      subnet: main_network_ipv6_r1
  - hostname: cmp-3-1
    name: cmp-3-1
    capabilities:
      node: cmp-3-1
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.10
      subnet: left_networ10_r3
    - network: right_network3
      fixed_ip: 100.64.0.10
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.3
      subnet: main_network_r3
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0009
      subnet: main_network_ipv6_r1
```

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

- 在 动态路由环境中，继续执行部署 Ceph 的下一步。

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. 在动态路由环境中部署 CEPH

通过对常规配置进行一些调整，您可以在使用动态路由的 Red Hat OpenStack Platform (RHOSP)环境中部署 Red Hat Ceph Storage。

注意

在使用动态路由的 Red Hat OpenStack Platform 环境中安装 Red Hat Ceph Storage 时，您必须在部署 overcloud 前安装 Ceph。在以下示例配置中，您可以使用 provisioning 网络来部署 Ceph。为获得最佳性能，建议您在 RHOSP 动态路由环境中部署 Ceph Storage 时使用专用 NIC 和网络硬件。

流程

1. 按照说明如何在 *部署 Red Hat Ceph Storage 和 Red Hat OpenStack Platform 中与 director 一起安装 Red Hat Ceph Storage*。在配置 Ceph Storage 集群的步骤中，请按照以下步骤操作：
2. 创建 Ceph 配置文件，再添加名为 **[global]** 的部分。

   示例

   在本例中，Ceph 配置文件名为 **initial-ceph.conf** ：

   ```
   $ echo "[global]" > initial-ceph.conf
   ```

3. 在 **[global]** 部分中，包含 **public_network** 和 **cluster_network** 参数，并为每个参数添加 **undercloud.conf** 文件中列出的子网 CIDR。仅包含与 overcloud 节点对应的子网 CIDR。

   提示

   要添加的子网 CIDR 是为 RHOSP 动态路由安装和配置 undercloud 中描述的子网 CIDR。

   示例

   在本例中，与 overcloud 节点对应的 **undercloud.conf** 文件中的子网添加到 **public_network** 和 **cluster_network** 参数中：

```
[global]
public_network="192.168.1.0/24,192.168.2.0/24,192.168.3.0/24"
cluster_network="192.168.1.0/24,192.168.2.0/24,192.168.3.0/24"
```

4. 当您准备好部署 Ceph 时，请确保在 **overcloud ceph deploy** 命令中包含 **initial-ceph.conf**。

   示例

```
$ openstack overcloud ceph deploy --config initial-ceph.conf \
<other_arguments> \
/home/stack/templates/overcloud-baremetal-deployed.yaml
```

重要

动态路由尚不可用。因此，如果 Ceph 节点需要路由来访问 NTP 服务器，则 Ceph 节点的 NTP 配置可能会延迟。

如果您的站点使用 NTP 服务器，请将 **--skip-ntp** 添加到 **openstack overcloud ceph deploy** 命令中。

在建立 BGP 路由前，不要将 Ceph 集群放入生产环境中，以便 Ceph 可以访问它所需的 NTP 服务器。在 overcloud 部署并配置了 NTP 之前，没有 NTP 的 Ceph 集群可能会导致一些异常情况，如守护进程忽略收到的消息、过期时间戳和超时，当消息没有及时或太晚时被触发。

5. 注意从运行 **overcloud ceph deploy** 命令生成的输出文件的路径和文件名。在本例中，**/home/stack/templates/overcloud-baremetal-deployed.yaml**。部署 overcloud 时，需要此信息。

后续步骤

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

其他资源

- *将 Red Hat Ceph Storage 和 Red Hat OpenStack Platform 与 director 一起部署*

4.11. 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。
- **overcloud ceph deploy** 命令的输出文件。
  示例： **overcloud-baremetal-deployed.yaml**。

  有关更多信息，请参阅 在动态路由环境中部署 Ceph。
- 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/templates/overcloud-baremetal-deployed.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

第 5 章 RHOSP 动态路由故障排除

诊断使用动态路由的 Red Hat OpenStack Platform (RHOSP)环境中的问题，从检查适当的日志并使用 VTY shell 查询各种 FRRouting 组件。

本节中包含的主题有：

- OVN BGP 代理和 FRRouting 日志
- 使用 VTY shell 命令对 BGP 进行故障排除

5.1. OVN BGP 代理和 FRROUTING 日志

Red Hat OpenStack Platform (RHOSP) OVN BGP 代理在此位置在 Compute 和 Networker 节点上写入其日志： **/var/log/containers/stdouts/ovn_bgp_agent.log**。

Free Range Routing (FRRouting 或 FRR)组件（如 BGP、BFD 和 Zebra 守护进程）在此位置的所有 RHOSP 节点上写入其日志： **/var/log/containers/frr/frr.log**

5.2. 使用 VTY SHELL 命令对 BGP 进行故障排除

您可以使用虚拟终端接口(VTY shell)的 shell 与 自由范围路由(FRRouting 或 FRR)守护进程交互。在 Red Hat OpenStack Platform 中，Rrrrr 守护进程（如 **bgpd** ）在容器中运行。使用 VTY shell 可帮助您对 BGP 路由问题进行故障排除。

先决条件

- 您必须在要运行 VTY shell 命令的主机上具有 sudo 权限。

流程

1. 登录到您要对 BGP 守护进程 **bgpd** 进行故障排除的主机。通常， **bgpd** 在所有 overcloud 节点上运行。
2. 输入 FRR 容器。

```
$ sudo podman exec -it frr bash
```

3. 您有两个选项来运行 VTY shell 命令：
   - 交互模式
     键入 **sudo vtysh** once 以进入交互模式来运行多个 VTY shell 命令。

     示例

```
$ sudo vtysh
> show bgp summary
```

   - 直接（Direct）模式
     在每个 VTY shell 命令前加 **sudo vtysh -c**。

     示例

```
$ sudo vtysh -c 'show bgp summary'
```

4. 以下是几个 VTY shell BGP 守护进程故障排除命令：

提示

使用 IPv6 时省略 **ip** 参数。

显示特定的路由表或所有路由表：

```
> show ip bgp <IPv4_address> | all
> show bgp <IPv6_address> | all
```

广告到 BGP 对等点的输出路由

```
> show ip bgp neighbors <router-ID> <advertised-routes>
```

从 BGP peer 接收的输出路由

```
> show ip bgp neighbors <router-ID> <received-routes>
```

其他资源

- 在 *FRRouting 用户指南*中显示 BGP 信息