Red Hat OpenStack Platform 13

스파인 Leaf 네트워킹

Red Hat OpenStack Platform director를 사용하여 라우팅된 스파인-리프 네트워크 설정

Last Updated: 2023-05-25

Red Hat OpenStack Platform 13 스파인 Leaf 네트워킹

Red Hat OpenStack Platform director를 사용하여 라우팅된 스파인-리프 네트워크 설정

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

이 가이드에서는 오버클라우드에서 라우팅된 스파인-리프 네트워크를 구성하는 방법에 대한 기본적인 시나리오를 제공합니다. 여기에는 언더클라우드 설정, 기본 구성 파일 작성, 노드의 역할 생성이 포함됩니다.

차례

1장. 소개

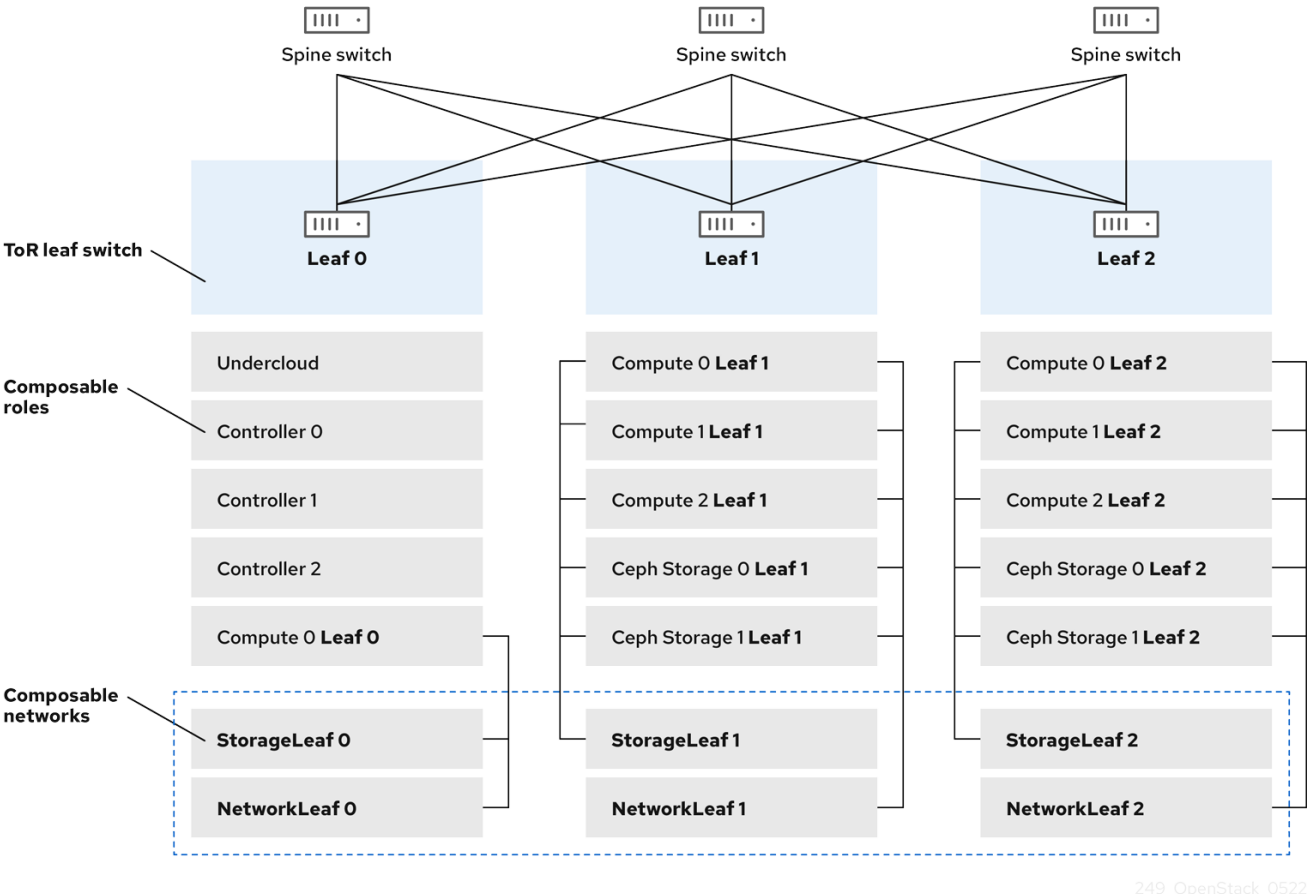
이 가이드에서는 Red Hat OpenStack Platform 환경에 스파인-리프형 네트워크 토폴로지를 구성하는 방법에 대한 정보를 제공합니다. 여기에는 사용자 환경 내에서 보다 광범위한 네트워크 토폴로지를 복제하는 데 도움이 되는 전체 엔드 투 엔드 시나리오 및 예제 파일이 포함됩니다.

1.1. 스파인-리프형 네트워킹

Red Hat OpenStack Platform의 구성 가능 네트워크 아키텍처를 사용하면 인기 있는 라우팅된 스파인-리프형 데이터 센터 토폴로지에 네트워킹을 조정할 수 있습니다. 라우팅된 스파인-리프의 실제 적용에서 리프는 일반적으로 그림 1.1. "라우팅된 스파인-leaf 예" 에 표시된 대로 데이터 센터 랙에서 구성 가능한 컴퓨팅 또는 스토리지 역할로 표시됩니다. *Leaf 0* 랙에는 언더클라우드 노드, 컨트롤러 및 계산 노드가 있습니다. 구성 가능 네트워크는 구성 가능 역할에 할당된 노드에 제공됩니다. 이 다이어그램에서 다음을 수행합니다.

- **StorageLeaf** 네트워크는 Ceph 스토리지 및 Compute 노드에 제공됩니다.
- **NetworkLeaf** 는 작성할 수 있는 네트워크의 예를 나타냅니다.

그림 1.1. 라우팅된 스파인-leaf 예



1.2. 네트워크 토폴로지

라우팅된 스파인-리프형 베어 메탈 환경에는 별도의 계층 2 브로드캐스트 도메인에서 격리된 VLAN 간에 트래픽을 라우팅하는 하나 이상의 계층 3 가능 스위치가 있습니다.

이 설계의 목적은 기능에 따라 트래픽을 분리하는 것입니다. 예를 들어, 컨트롤러 노드가 *내부* API 네트워크에서 API를 호스팅하는 경우, 계산 노드가 API에 액세스할 때 *내부 API* 네트워크의 자체 버전을 사용해

야 합니다. 이 라우팅이 작동하려면 *내부 API* 네트워크로 향하는 트래픽을 강제로 필요한 인터페이스를 사용하도록 라우팅이 필요합니다. 이는 *슈퍼넷* 경로를 사용하여 구성할 수 있습니다. 예를 들어 컨트롤러 노드의 *내부 API* 네트워크로 **172.18.0.0/24** 를 사용하는 경우 두 번째 *내부 API* 네트워크에 **172.18.1.0/24**, 세 번째에는 **172.18.2.0/24** 를 사용할 수 있습니다. 결과적으로 각 계층 2 도메인의 각 역할에 대해 로컬 *내부 API* 네트워크에서 게이트웨이 IP를 사용하는 더 큰 **172.18.0.0/16** 슈퍼넷을 가리키는 경로가 있을 수 있습니다.

이 시나리오에서는 다음 네트워크를 사용합니다.

표 1.1. 리프 0 네트워크

네트워크	연결된 역할	인터페이스	Bridge	서브넷
프로비저닝/컨트롤 플레인	All	nic1	br-ctlplane(undercloud)	192.168.10.0/24
스토리지	컨트롤러	nic2		172.16.0.0/24
스토리지 Mgmt	컨트롤러	nic3		172.17.0.0/24
내부 API	컨트롤러	nic4		172.18.0.0/24
테넌트	컨트롤러	nic5		172.19.0.0/24
외부	컨트롤러	nic6	br-ex	10.1.1.0/24

표 1.2. 리프 1 네트워크

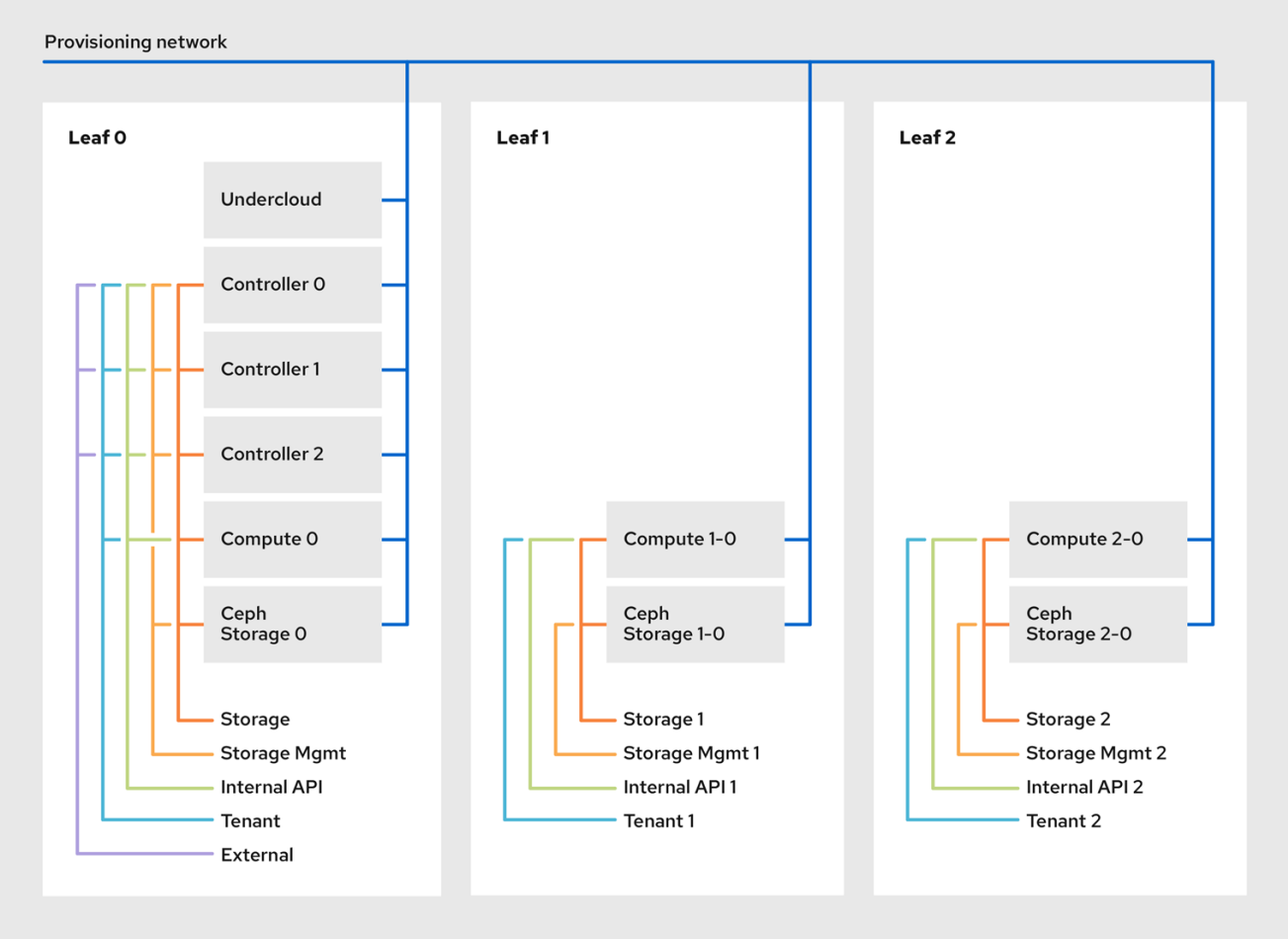
네트워크	연결된 역할	인터페이스	Bridge	서브넷
프로비저닝/컨트롤 플레인	All	nic1	br-ctlplane(undercloud)	192.168.11.0/24
Storage1	Compute1, Ceph1	nic2		172.16.1.0/24
스토리지 Mgmt1	Ceph1	nic3		172.17.1.0/24
내부 API1	Compute1	nic4		172.18.1.0/24
Tenant1	Compute1	nic5		172.19.1.0/24

표 1.3. 리프 2 네트워크

네트워크	연결된 역할	인터페이스	Bridge	서브넷
프로비저닝/컨트롤 플레인	All	nic1	br-ctlplane(undercloud)	192.168.12.0/24
Storage2	Compute2, Ceph2	nic2		172.16.2.0/24
스토리지 Mgmt2	Ceph2	nic3		172.17.2.0/24
내부 API2	Compute2	nic4		172.18.2.0/24
Tenant2	Compute2	nic5		172.19.2.0/24

표 1.4. 슈퍼넷 경로

네트워크	서브넷
스토리지	172.16.0.0/16
스토리지 Mgmt	172.17.0.0/16
내부 API	172.18.0.0/16
테넌트	172.19.0.0/16



249_OpenStack_0522

1.3. 스파인-리프 요구 사항

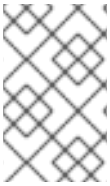
계층-3 라우팅 아키텍처가 있는 네트워크에 오버클라우드를 배포하려면 다음 요구 사항을 충족해야 합니다.

계층-3 라우팅

네트워크 인프라는 다른 계층 2 세그먼트 간 트래픽을 활성화하도록 라우팅이 구성되어 있어야 합니다. 이는 정적으로 또는 동적으로 구성할 수 있습니다.

DHCP-Relay

언더클라우드에 로컬로 표시되지 않는 각 계층 2 세그먼트는 **dhcp-relay** 를 제공해야 합니다. 언더클라우드가 연결된 프로비저닝 네트워크 세그먼트에서 DHCP 요청을 언더클라우드에 전달해야 합니다.



참고

언더클라우드는 두 개의 DHCP 서버를 사용합니다. 하나는 baremetal 노드 인트로스펙션이고 다른 하나는 오버클라우드 노드를 배포하기 위한 것입니다. **dhcp-relay** 를 구성할 때 요구 사항을 이해하려면 DHCP 릴레이 구성을 읽습니다.

1.4. 스파인-리프 제한

- Controller 역할과 같은 일부 역할은 가상 IP 주소 및 클러스터링을 사용합니다. 이 기능의 메커니즘에는 이러한 노드 간 계층 2 네트워크 연결이 필요합니다. 이러한 노드는 모두 동일한 리프 내에 배치됩니다.

- Networker 노드에도 유사한 제한 사항이 적용됩니다. 네트워크 서비스는 VRRP(Virtual Router Redundancy Protocol)를 사용하여 네트워크에서 고가용성 기본 경로를 구현합니다. VRRP는 가상 라우터 IP 주소를 사용하므로 마스터 및 백업 노드를 동일한 L2 네트워크 세그먼트에 연결해야 합니다.

- VLAN 분할과 함께 테넌트 또는 공급자 네트워크를 사용하는 경우 모든 Networker와 Compute 노드 간에 특정 VLAN을 공유해야 합니다.

참고

여러 Networker 노드 세트로 네트워크 서비스를 구성할 수 있습니다. 각 네트워크 공유 경로를 설정하고 VRRP는 각 Networker 노드 세트 내에서 고가용성 기본 경로를 제공합니다. 이러한 구성에서 네트워크를 공유하는 모든 Networker 노드는 동일한 L2 네트워크 세그먼트에 있어야 합니다.

2장. 언더클라우드 설정

이 섹션에서는 구성 가능한 네트워크를 사용하여 라우팅된 스파인-리프를 수용하도록 언더클라우드를 구성하는 방법에 대한 사용 사례에 대해 설명합니다.

2.1. 스파인 리프 프로비저닝 네트워크 구성

스파인 리프 인프라에 대한 프로비저닝 네트워크를 구성하려면 **undercloud.conf** 파일을 편집하고 다음 절차에 정의된 대로 관련 매개변수를 설정합니다.

절차

1. **stack** 사용자로 언더클라우드에 로그인합니다.
2. **undercloud.conf** 가 아직 없는 경우 샘플 템플릿 파일을 복사합니다.

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

3. **undercloud.conf** 를 편집합니다.
4. **[DEFAULT]** 섹션에서 다음을 수행합니다.
   a. **leaf0** 에서 **local_ip** 를 언더클라우드 IP로 설정합니다.

```
local_ip = 192.168.10.1/24
```

   b. **undercloud_public_vip** 를 외부에서 언더클라우드의 IP 주소로 설정합니다.

```
undercloud_public_vip = 10.1.1.1
```

   c. **undercloud_admin_vip** 를 언더클라우드의 관리 IP 주소로 설정합니다. 이 IP 주소는 일반적으로 leaf0에 있습니다.

```
undercloud_admin_vip = 192.168.10.2
```

   d. **local_interface** 를 로컬 네트워크의 브릿지로 인터페이스로 설정합니다.

```
local_interface = eth1
```

   e. **enable_routed_networks** 를 **true** 로 설정합니다.

```
enable_routed_networks = true
```

   f. **subnets** 매개 변수를 사용하여 서브넷 목록을 정의합니다. 라우팅된 스파인 및 리프에서 각 계층 2 세그먼트에 대해 하나의 서브넷을 정의합니다.

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

   g. **local_subnet** 매개변수를 사용하여 물리적 계층 2 세그먼트 로컬과 연결된 서브넷을 언더클라우드로 지정합니다.

```
local_subnet = leaf0
```

5. **subnets** 매개변수로 정의된 각 서브넷별로 새 섹션을 생성합니다.

```
[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. **undercloud.conf** 파일을 저장합니다.

7. 언더클라우드 설치 명령을 실행합니다.

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

이렇게 하면 provisioning 네트워크 / 컨트롤 플레인에 3개의 서브넷이 생성됩니다. 오버클라우드는 각 네트워크를 사용하여 각 리프 내의 시스템을 프로비저닝합니다.

언더클라우드에 DHCP 요청을 올바르게 릴레이하려면 DHCP 릴레이를 구성해야 할 수 있습니다. 다음 섹션에서는 DHCP 릴레이를 구성하는 방법에 대한 몇 가지 정보를 제공합니다.

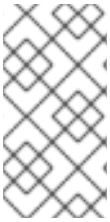
2.2. DHCP 릴레이 구성

언더클라우드는 provisioning 네트워크에서 두 개의 DHCP 서버를 사용합니다.

- 인트로스펙션 중 하나입니다.
- 하나는 프로비저닝에 사용됩니다.

DHCP 릴레이를 구성할 때 언더클라우드의 두 DHCP 서버로 DHCP 요청을 전달하도록 합니다.

이를 지원하는 장치와 함께 UDP 브로드캐스트를 사용하여 언더클라우드 프로비저닝 네트워크가 연결된 L2 네트워크 세그먼트에 DHCP 요청을 릴레이할 수 있습니다. 또는 DHCP 요청을 특정 IP 주소로 릴레이하는 UDP 유니캐스트를 사용할 수 있습니다.



참고

특정 장치 유형에 대한 DHCP 릴레이 구성은 이 문서의 범위를 벗어납니다. 참조로 이 문서는 ISC DHCP 소프트웨어 구현을 사용하는 DHCP 릴레이 구성 예제를 아래에서 제공합니다. 이 구현을 사용하는 방법에 대한 자세한 내용은 도움말 페이지 dhcrelay(8)를 참조하십시오.

브로드캐스트 DHCP 릴레이

이 메서드는 UDP 브로드캐스트 트래픽을 사용하여 DHCP 서버가 있는 L2 네트워크 세그먼트에 DHCP 요청을 릴레이합니다. 네트워크 세그먼트의 모든 장치는 브로드캐스트 트래픽을 수신합니다. UDP 브로드캐스트를 사용하면 언더클라우드의 두 DHCP 서버가 릴레이된 DHCP 요청을 수신합니다. 일반적으로 이 값은 인터페이스 또는 IP 네트워크 주소를 지정하여 구성됩니다.

인터페이스

DHCP 요청이 릴레이되는 L2 네트워크 세그먼트에 연결된 인터페이스를 지정합니다.

IP 네트워크 주소

DHCP 요청이 릴레이되는 IP 네트워크의 네트워크 주소를 지정합니다.

유니캐스트 DHCP 릴레이

이 방법은 UDP 유니캐스트 트래픽을 사용하여 특정 DHCP 서버로 DHCP 요청을 릴레이합니다. UDP 유니캐스트를 사용하는 경우, 언더클라우드에서 인트로스펙션에 사용되는 인터페이스에 할당된 IP 주소 및 OpenStack Networking(neutron) 서비스에서 생성된 네트워크 네임스페이스의 IP 주소 모두에 DHCP 요청을 릴레이하도록 DHCP 요청을 제공하는 장치를 구성하여 **ctlplane** 네트워크의 DHCP 서비스를 호스팅해야 합니다.

세부 검사에 사용되는 인터페이스는 **undercloud.conf** 에서 inspection_interface로 정의된 인터페이스입니다.

참고

인트로스펙션에 **br-ctlplane** 인터페이스를 사용하는 것이 일반적입니다.
undercloud.conf 에서 **local_ip** 로 정의된 IP 주소는 **br-ctlplane** 인터페이스에 있습니다.

Neutron DHCP 네임스페이스에 할당된 IP 주소는 undercloud.conf의 local_subnet에 구성된 IP 범위에서 사용 가능한 첫 번째 주소입니다. IP 범위의 첫 번째 주소는 구성에서 **dhcp_start** 로 정의된 주소입니다. 예를 들면 **192.168.10.10** 은 다음 구성이 사용되는 경우 IP 주소가 됩니다.

```
[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
```

주의

DHCP 네임스페이스의 IP 주소가 자동으로 할당됩니다. 대부분의 경우 IP 범위의 첫 번째 주소입니다. 언더클라우드에서 다음 명령을 실행하여 이 문제가 발생하는지 확인합니다.

```
$ 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 구성 예

다음 예에서 **dhcp** 패키지의 **dhcrelay** 명령은 다음 구성을 사용합니다.

- 들어오는 DHCP 요청을 릴레이하는 인터페이스: **eth1**,**eth2** 및 **eth3**.
- 네트워크 세그먼트의 언더클라우드 DHCP 서버가 **eth0** 에 연결되어 있습니다.
- 인트로스펙션에 사용되는 DHCP 서버는 IP 주소 '192.168.10.1에서 수신 대기 중입니다.
- 프로비저닝에 사용되는 DHCP 서버는 IP 주소 **192.168.10.10** 에서 수신 대기합니다.

그러면 다음과 같은 **dhcrelay** 명령이 생성됩니다.

```
$ sudo dhcrelay -d --no-pid 172.20.0.10 172.20.0.1 \
  -i eth0 -i eth1 -i eth2 -i eth3
```

Cisco IOS 라우팅 스위치 구성의 예

이 예에서는 다음 Cisco IOS 구성을 사용하여 다음 작업을 수행합니다.

- 프로비저닝 네트워크에 사용할 VLAN을 구성합니다.
- 리프의 IP 주소를 추가합니다.
- UDP 및 BOOTP 요청을 IP 주소에서 수신 대기하는 인트로스펙션 DHCP 서버로 전달합니다. **192.168.10.1**.
- UDP 및 BOOTP 요청을 IP 주소 **192.168.10.10** 에서 수신 대기하는 프로비저닝 DHCP 서버로 전달합니다.

```
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
!
```

이제 프로비저닝 네트워크를 설정했으므로 나머지 오버클라우드 리프 네트워크를 설정할 수 있습니다. 일련의 구성 파일을 사용하여 이 작업을 수행합니다.

2.3. 리프 네트워크에 대한 플레이버 생성 및 태그 노드 생성

각 리프 네트워크의 각 역할에는 각 리프에 노드를 태그할 수 있도록 플레이버 및 역할 할당이 필요합니다. 다음 절차에서는 각 플레이버를 생성하고 역할에 할당하는 방법을 설명합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 각 사용자 정의 역할에 대한 플레이버를 만듭니다.

```
$ ROLES="control0 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 4096 --disk 40 --vcpus 1 $ROLE ; done
$ for ROLE in $ROLES; do openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local" --property "capabilities:profile"="$ROLE" $ROLE ; done
```

3. 노드를 각 리프 네트워크에 태그합니다. 예를 들어 다음 명령을 실행하여 UUID **58c3d07e-24f2-48a7-bbb6-6843f0e8ee13** 로 노드를 Leaf2의 compute 역할에 태그합니다.

```
$ openstack baremetal node set --property capabilities='profile:compute_leaf2,boot_option:local' 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13
```

4. 플레이버를 역할에 매핑하는 환경 파일(~**templates/node-data.yaml**)을 생성합니다.

```
parameter_defaults:
  OvercloudController0Flavor: control0
  Controller0Count: 3
  OvercloudCompute0Flavor: compute_leaf0
  Compute0Count: 3
  OvercloudCompute1Flavor: compute_leaf1
  Compute1Count: 3
  OvercloudCompute2Flavor: compute_leaf2
  Compute2Count: 3
  OvercloudCephStorage0Flavor: ceph-storage_leaf0
  CephStorage0Count: 3
  OvercloudCephStorage1Flavor: ceph-storage_leaf1
  CephStorage1Count: 3
  OvercloudCephStorage2Flavor: ceph-storage_leaf2
  CephStorage2Count: 3
```

각 *Count' 매개변수를 사용하여 오버클라우드에 배포할 노드 수를 설정할 수도 있습니다.

2.4. 베어 메탈 노드 포트를 컨트롤 플레인 네트워크 세그먼트에 매핑

L3 라우팅 네트워크에 배포할 수 있도록 베어 메탈 포트에는 **physical_network** 필드가 구성되어 있어야 합니다. 각 베어 메탈 포트는 OpenStack Bare Metal(ironic) 서비스의 베어 메탈 노드와 연결됩니다. 물리적 네트워크 이름은 언더클라우드 구성의 **subnets** 옵션에서 사용하는 이름입니다.

참고

undercloud.conf 에서 **local_subnet** 으로 지정된 서브넷의 물리적 네트워크 이름은 특수합니다. 항상 **ctlplane** 이라고 합니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 베어 메탈 노드를 확인합니다.

```
$ openstack baremetal node list
```

3. 베어 메탈 노드가 **등록** 되거나 **manageable** 상태인지 확인합니다. 베어 메탈 노드가 이러한 상태 중 하나에 없는 경우 baremetal 포트에서 physical_network 속성을 설정하는 데 사용된 명령이 실패합니다. 모든 노드를 **manageable** 상태로 설정하려면 다음 명령을 실행합니다.

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

4. 어떤 baremetal 노드와 연결된 베어 메탈 포트를 확인합니다. 예를 들어 다음과 같습니다.

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

5. 포트의 **physical-network** 매개 변수를 설정합니다. 아래 예제에서 3개의 서브넷은 **leaf0**,**leaf1**, **leaf2**)에 정의되어 있습니다. local_subnet은 **leaf0** 입니다. **local_subnet** 의 물리적 네트워크는 항상 **ctlplane** 입니다. **leaf0** 에 연결된 baremetal 포트는 ctlplane를 사용합니다. 나머지 포트는 다른 리프 이름을 사용합니다.

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

6. 오버클라우드를 배포하기 전에 노드가 사용 가능 상태인지 확인합니다.

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

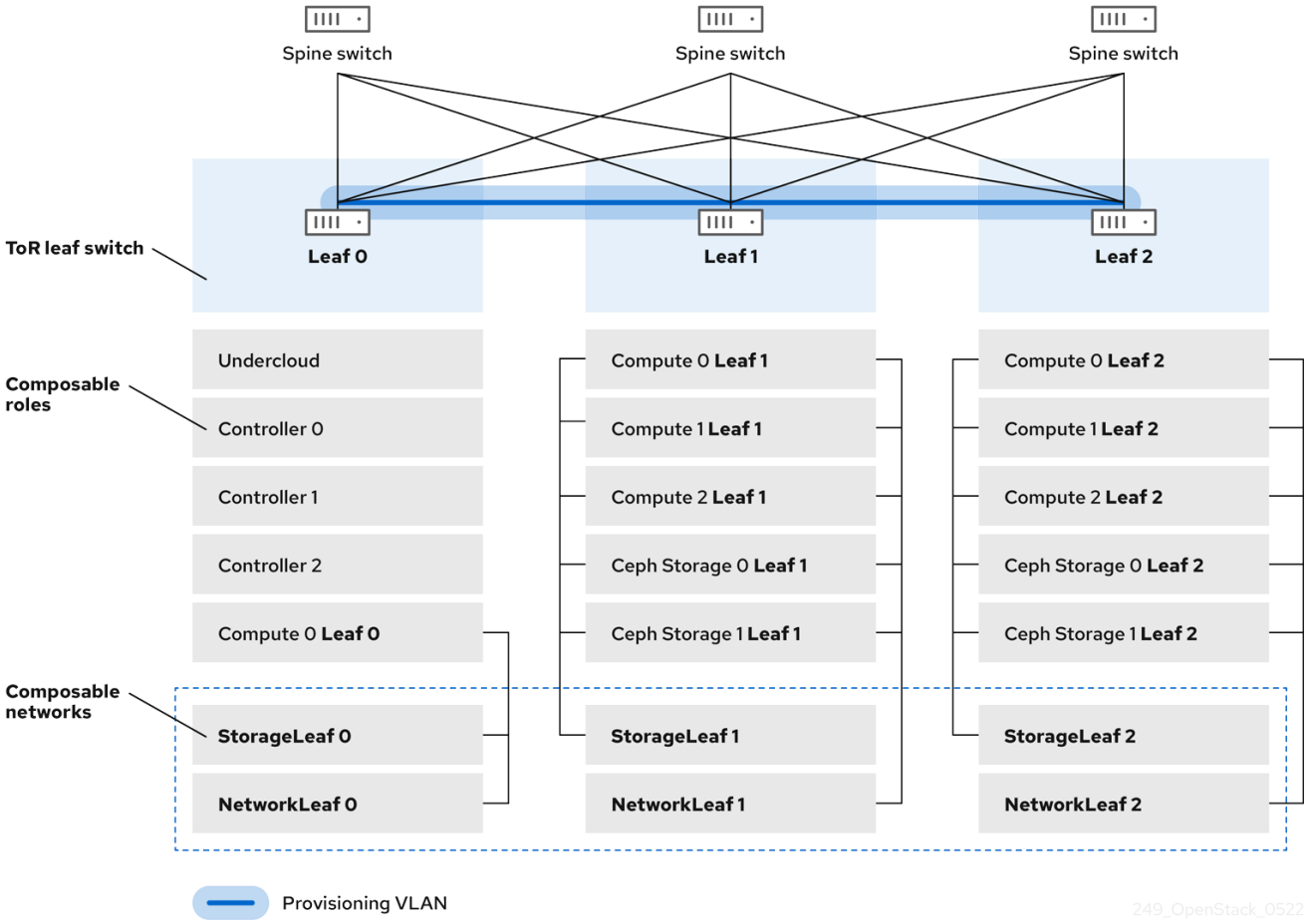
3장. 대체 프로비저닝 네트워크 방법

이 섹션에서는 구성 가능한 네트워크에서 라우팅된 스파인-리프를 수용하도록 프로비저닝 네트워크를 구성하는 다른 방법에 대해 설명합니다.

3.1. VLAN 프로비저닝 네트워크

이 예제에서 director는 provisioning 네트워크를 통해 새 오버클라우드 노드를 배포하고 계층 3 토폴로지에서 VLAN 터널을 사용합니다(그림 3.1. "VLAN 프로비저닝 네트워크 토폴로지" 참조). 이를 통해 director의 DHCP 서버에서 **DHCPOFFER** 브로드캐스트를 모든 리프에 보낼 수 있습니다. 이 터널을 설정하려면 top-of-Rack(ToR) 리프 스위치 간에 VLAN을 트렁크합니다. 이 다이어그램에서 **StorageLeaf** 네트워크는 Ceph 스토리지 및 Compute 노드에 표시됩니다. **NetworkLeaf** 는 구성할 수 있는 네트워크의 예를 나타냅니다.

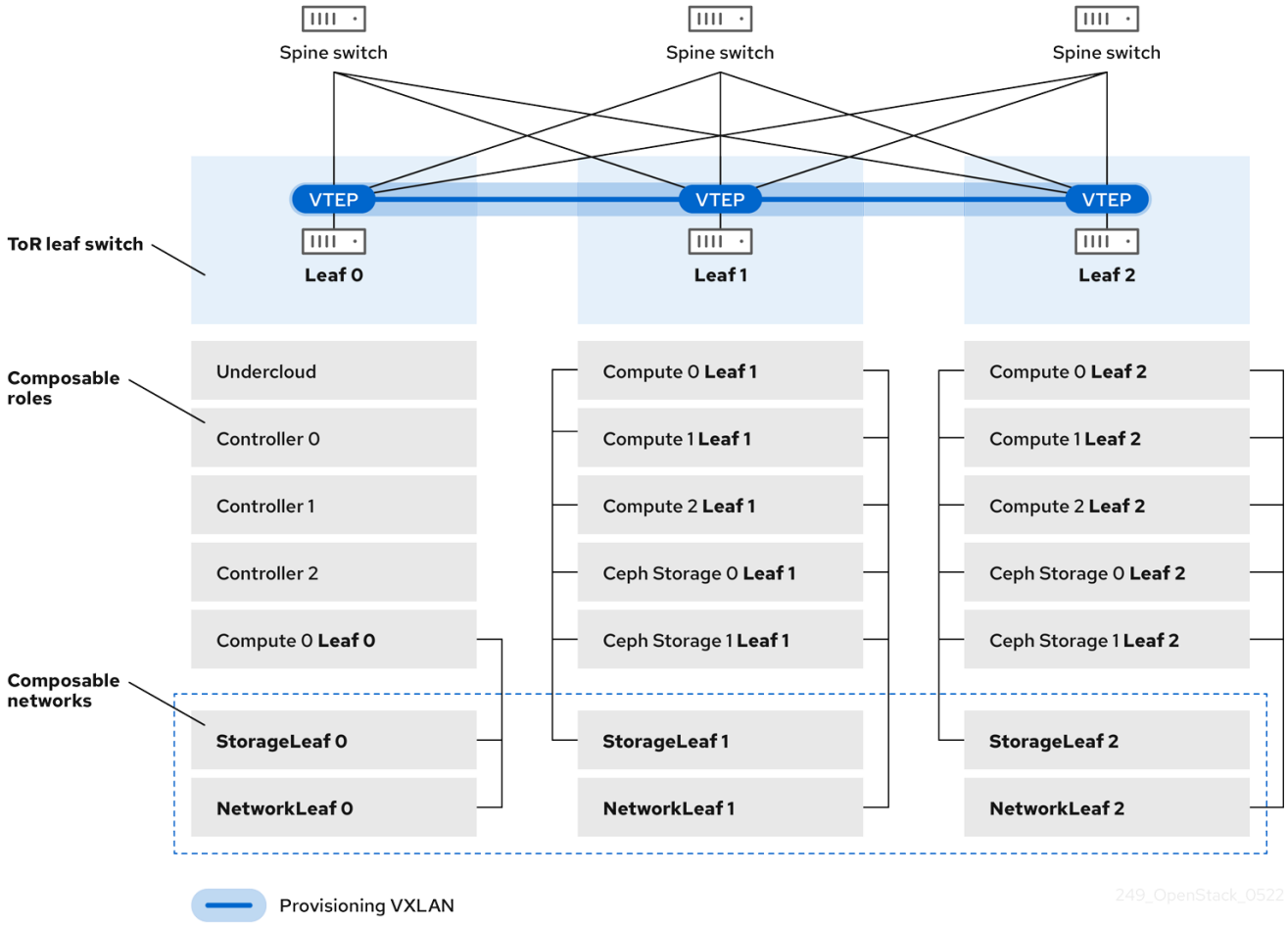
그림 3.1. VLAN 프로비저닝 네트워크 토폴로지



3.2. VXLAN 프로비저닝 네트워크

이 예제에서 director는 프로비저닝 네트워크를 통해 새 오버클라우드 노드를 배포하고 VXLAN 터널을 사용하여 계층 3 토폴로지에 걸쳐 있습니다(그림 3.2. "VXLAN 프로비저닝 네트워크 토폴로지" 참조). 이를 통해 director의 DHCP 서버에서 **DHCPOFFER** 브로드캐스트를 모든 리프에 보낼 수 있습니다. 이 터널을 설정하려면 top-of-Rack(ToR) 리프 스위치에서 VXLAN 엔드포인트를 구성합니다.

그림 3.2. VXLAN 프로비저닝 네트워크 토폴로지



4장. 오버클라우드 설정

이제 언더클라우드를 설정했으므로 나머지 오버클라우드 리프 네트워크를 설정할 수 있습니다. 일련의 구성 파일을 사용하여 이 작업을 수행합니다. 나중에 오버클라우드를 배포하고 결과 배포에는 라우팅이 있는 여러 네트워크 세트가 있습니다.

4.1. 네트워크 데이터 파일 생성

리프 네트워크를 정의하려면 구성 가능한 각 네트워크 및 해당 특성의 YAML 포맷 목록이 포함된 네트워크 데이터 파일을 생성합니다. 기본 네트워크 데이터는 **/usr/share/openstack-tripleo-heat-templates/network_data.yaml** 의 언더클라우드에 있습니다.

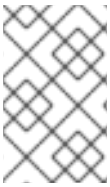
절차

1. **stack** 사용자의 로컬 디렉터리에 새 **network_data_spine_leaf.yaml** 파일을 만듭니다. 기본 **network_data** 파일을 기본으로 사용합니다.

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

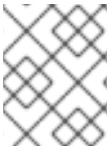
2. **network_data_spine_leaf.yaml** 파일에서 YAML 목록을 생성하여 각 네트워크 및 리프 네트워크를 구성 가능한 네트워크 항목으로 정의합니다. 예를 들어, 내부 API 네트워크 및 리프 네트워크는 다음 구문을 사용하여 정의됩니다.

```
# Internal API
- name: InternalApi
  name_lower: internal_api
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: false
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: false
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]
```



참고

언더클라우드는 이러한 네트워크를 이미 생성했기 때문에 네트워크 데이터 파일에 컨트롤 플레인 네트워크를 정의하지 않습니다. 그러나 오버클라우드에서 NIC를 적절하게 구성할 수 있도록 매개변수를 수동으로 설정해야 합니다.



참고

컨트롤러 기반 서비스가 포함된 네트워크에 대해 **vip: true** 를 정의합니다. 이 예에서 **InternalApi** 에는 이러한 서비스가 포함되어 있습니다.

모든 구성 가능 네트워크가 포함된 전체 예제는 부록 A. *network_data 파일 예* 를 참조하십시오.

4.2. 역할 데이터 파일 생성

이 섹션에서는 각 리프에 대한 각 구성 가능 역할을 정의하고 각 역할에 구성 가능 네트워크를 연결하는 방법을 보여줍니다.

절차

1. **stack** 사용자의 로컬 디렉터리에 사용자 지정 **역할** director를 생성합니다.

```
$ mkdir ~/roles
```

2. director의 코어 템플릿 컬렉션에서 기본 Controller, Compute, Ceph Storage 역할을 **~/roles** 디렉터리에 복사합니다. Leaf 1의 파일 이름을 바꿉니다.

```
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Controller.yaml ~/roles/Controller.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Compute.yaml ~/roles/Compute1.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/CephStorage.yaml
~/roles/CephStorage1.yaml
```

3. **Compute1.yaml** 파일을 편집합니다.

```
$ vi ~/roles/Compute1.yaml
```

4. Leaf 1 특정 매개변수와 일치하도록 이 파일에서 **이름**,**networks**, **HostnameFormatDefault** 매개변수를 편집합니다. 예를 들어 다음과 같습니다.

```
- name: Compute1
  ...
  networks:
    - InternalApi1
    - Tenant1
    - Storage1
  HostnameFormatDefault: '%stackname%-compute1-%index%'
```

   이 파일을 저장합니다.

5. **CephStorage1.yaml** 파일을 편집합니다.

```
$ vi ~/roles/CephStorage1.yaml
```

6. 이 파일의 **name** 및 **networks** 매개 변수를 편집하여 Leaf 1 특정 매개 변수와 일치하도록 합니다. 또한 **HostnameFormatDefault** 매개변수를 추가하고 Ceph Storage 노드의 Leaf 1 호스트 이름을 정의합니다. 예를 들어 다음과 같습니다.

```
- name: CephStorage1
  ...
  networks:
    - Storage1
    - StorageMgmt1
  HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
```

   이 파일을 저장합니다.

7. Leaf 2 및 Leaf 3 파일의 기반으로 Leaf 1 Compute 및 Ceph Storage 파일을 복사합니다.

```
$ cp ~/roles/Compute1.yaml ~/roles/Compute2.yaml
$ cp ~/roles/Compute1.yaml ~/roles/Compute3.yaml
$ cp ~/roles/CephStorage1.yaml ~/roles/CephStorage2.yaml
$ cp ~/roles/CephStorage1.yaml ~/roles/CephStorage3.yaml
```

8. Leaf 2 및 Leaf 3 파일에서 **이름**,**networks**, **HostnameFormatDefault** 매개변수를 편집하여 해당 Leaf 네트워크 매개 변수와 일치하도록 합니다. 예를 들어 Leaf 2 Compute 파일의 매개변수에는 다음과 같은 값이 있습니다.

```
- name: Compute2
  ...
  networks:
    - InternalApi2
    - Tenant2
    - Storage2
  HostnameFormatDefault: '%stackname%-compute2-%index%'
```

Leaf 2 Ceph Storage 매개변수에는 다음과 같은 값이 있습니다.

```
- name: CephStorage2
  ...
  networks:
    - Storage2
    - StorageMgmt2
  HostnameFormatDefault: '%stackname%-cephstorage2-%index%'
```

9. 역할이 준비되면 다음 명령을 사용하여 전체 역할 데이터 파일을 생성합니다.

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

이렇게 하면 각 리프 네트워크에 대한 모든 사용자 지정 역할을 포함하는 전체 **roles_data_spine_leaf.yaml** 파일이 생성됩니다.

이 파일의 전체 예는 부록 C. *roles_data 파일의 예* 를 참조하십시오.

각 역할에는 자체 NIC 구성이 있습니다. 스파인-리프형 구성을 구성하기 전에 현재 NIC 구성에 맞게 기본 NIC 템플릿 세트를 생성해야 합니다.

4.3. 사용자 정의 NIC 설정 생성

각 역할에는 자체 NIC 구성이 필요합니다. NIC 템플릿 기본 세트의 사본을 생성하고 현재 NIC 구성에 맞게 수정합니다.

절차

1. 코어 Heat 템플릿 디렉터리로 변경합니다.

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

2. **tools/process-templates.py** 스크립트, 사용자 지정 **network_data** 파일 및 사용자 지정 **roles_data** 파일을 사용하여 Jinja2 템플릿을 렌더링합니다.

```
$ 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. 홈 디렉터리로 변경합니다.

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

4. 스파인-리프형 템플릿의 기반으로 사용할 기본 NIC 템플릿 중 하나에서 콘텐츠를 복사합니다. 예를 들어 **single-nic-vlans** 를 복사합니다.

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

5. 렌더링된 템플릿 디렉터리를 제거합니다.

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

리소스

- NIC 템플릿 사용자 지정에 대한 자세한 내용은 *Advanced Overcloud Customization* 가이드의 "Custom Network Interface Templates" 를 참조하십시오.

4.4. 사용자 정의 컨트롤러 NIC 설정 편집

렌더링된 템플릿에는 스파인-리프 구성에 맞게 필요한 대부분의 콘텐츠가 포함되어 있습니다. 그러나 일부 추가 구성 변경이 필요합니다. Leaf0에서 컨트롤러 노드의 YAML 구조를 수정하려면 다음 절차를 따르십시오.

절차

1. 사용자 정의 NIC 디렉터리로 변경합니다.

```
$ cd ~/templates/spine-leaf-nics/
```

2. **controller0.yaml** 에 대한 템플릿을 편집합니다.

3. 매개변수 섹션의 **ControlPlaneSubnetCidr** 및 **ControlPlaneDefaultRoute 매개변수** 로 스크롤합니다. 이러한 매개변수는 다음 스니펫과 유사합니다.

```
ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
  type: string
ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: The default route of the control plane network.
  type: string
```

Leaf0에 맞게 이러한 매개변수를 수정합니다.

```
ControlPlane0SubnetCidr: # Override this via parameter_defaults
  default: '24'
```

```
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlane0DefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
```

4. 매개 변수 섹션의 **EC2MetadataIp 매개변수** 로 스크롤합니다. 이 매개변수는 다음 스니펫과 유사합니다.

```
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

Leaf0에 맞게 이 매개변수를 수정합니다.

```
  Leaf0EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

5. 네트워크 구성 섹션으로 스크롤합니다. 이 섹션에서는 다음 예와 유사합니다.

```
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:
```

스크립트의 위치를 절대 경로로 변경합니다.

```
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:
```

6. **network_config** 섹션에서 컨트롤 플레인 / 프로비저닝 인터페이스를 정의합니다. 예를 들어 다음과 같습니다.

```
              network_config:
```

```
- type: ovs_bridge
  name: bridge_name
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
  - ip_netmask:
      list_join:
      - /
      - - get_param: ControlPlaneIp
        - get_param: ControlPlane0SubnetCidr
  routes:
  - ip_netmask: 169.254.169.254/32
    next_hop:
      get_param: Leaf0EC2MetadataIp
  - ip_netmask: 192.168.10.0/24
    next_hop:
      get_param: ControlPlane0DefaultRoute
```

이 경우 사용되는 매개변수는 Leaf0: **ControlPlane0SubnetCidr**,**Leaf0EC2MetadataIp**, **ControlPlane0DefaultRoute** 에 고유합니다. 또한 프로비저닝 네트워크(192.168.10.0/24)의 Leaf0에 CIDR을 사용하여 경로로 사용됩니다.

7. **members** 섹션의 각 VLAN에는 관련 Leaf0 매개 변수가 포함되어 있습니다. 예를 들어 Storage 네트워크 VLAN 정보는 다음 스니펫과 유사해야 합니다.

```
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
      get_param: Storage0IpSubnet
```

라우팅 매개 변수를 정의하는 섹션을 추가합니다. 여기에는 슈퍼넷 경로(이 경우**StorageSupernet**)와 리프 기본 경로(이 경우**Storage0InterfaceDefaultRoute**)가 포함됩니다.

```
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
      get_param: Storage0IpSubnet
  routes:
  - ip_netmask:
      get_param: StorageSupernet
    next_hop:
      get_param: Storage0InterfaceDefaultRoute
```

스토리지 , **Storage Mgmt**,**InternalApi**, **Tenant** 등 컨트롤러 네트워크의 VLAN 구조 경로를 추가합니다.

8. 이 파일을 저장합니다.

4.5. 사용자 정의 컴퓨팅 NIC 구성 생성

이 절차에서는 Leaf0, Leaf1 및 Leaf2의 Compute 노드에 대한 YAML 구조를 생성합니다.

절차

1. 사용자 정의 NIC 디렉터리로 변경합니다.

```
$ cd ~/templates/spine-leaf-nics/
```

2. **compute0.yaml** 에 대한 템플릿을 편집합니다.

3. 매개변수 섹션의 **ControlPlaneSubnetCidr** 및 **ControlPlaneDefaultRoute 매개변수** 로 스크롤합니다. 이러한 매개변수는 다음 스니펫과 유사합니다.

```
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
```

   Leaf0에 맞게 이러한 매개변수를 수정합니다.

```
  ControlPlane0SubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlane0DefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
```

4. 매개 변수 섹션의 **EC2MetadataIp 매개변수** 로 스크롤합니다. 이 매개변수는 다음 스니펫과 유사합니다.

```
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

   Leaf0에 맞게 이 매개변수를 수정합니다.

```
  Leaf0EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

5. 네트워크 구성 섹션으로 스크롤합니다. 이 섹션은 다음 스니펫과 유사합니다.

```
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:
```

스크립트의 위치를 절대 경로로 변경합니다.

```
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:
```

6. **network_config** 섹션에서 컨트롤 플레인 / 프로비저닝 인터페이스를 정의합니다. 예를 들면 다음과 같습니다

```
            network_config:
            - type: interface
              name: nic1
              use_dhcp: false
              dns_servers:
                get_param: DnsServers
              addresses:
              - ip_netmask:
                  list_join:
                  - /
                  - - get_param: ControlPlaneIp
                    - get_param: ControlPlane0SubnetCidr
              routes:
              - ip_netmask: 169.254.169.254/32
                next_hop:
                  get_param: Leaf0EC2MetadataIp
              - ip_netmask: 192.168.10.0/24
                next_hop:
                  get_param: ControlPlane0DefaultRoute
```

이 경우 사용되는 매개변수는 Leaf0: **ControlPlane0SubnetCidr**,**Leaf0EC2MetadataIp**, **ControlPlane0DefaultRoute** 에 고유합니다. 또한 프로비저닝 네트워크(192.168.10.0/24)의 Leaf0에 CIDR을 사용하여 경로로 사용됩니다.

7. **members** 섹션의 각 VLAN에는 관련 Leaf0 매개 변수가 포함되어야 합니다. 예를 들어 Storage 네트워크 VLAN 정보는 다음 스니펫과 유사해야 합니다.

```
            - type: vlan
              vlan_id:
                get_param: Storage0NetworkVlanID
```

```
            addresses:
            - ip_netmask:
                get_param: Storage0IpSubnet
```

라우팅 매개 변수를 정의하는 섹션을 추가합니다. 여기에는 슈퍼넷 경로(이 경우**StorageSupernet**)와 리프 기본 경로(이 경우 **Storage0InterfaceDefaultRoute**)가 포함됩니다.

```
          - type: vlan
            vlan_id:
              get_param: Storage0NetworkVlanID
            addresses:
            - ip_netmask:
                get_param: Storage0IpSubnet
            routes:
            - ip_netmask:
                get_param: StorageSupernet
              next_hop:
                get_param: Storage0InterfaceDefaultRoute
```

다음 컨트롤러 네트워크에 대한 VLAN 구조를 추가합니다. 스토리지 **, InternalApi**, **Tenant**.

8. 이 파일을 저장합니다.
9. **compute1.yaml** 을 편집하고 동일한 단계를 수행합니다. 다음은 변경 사항 목록입니다.
   - **ControlPlaneSubnetCidr** 를 **ControlPlane1SubnetCidr** 로 변경합니다.
   - **ControlPlaneDefaultRoute** 를 **ControlPlane1DefaultRoute** 로 변경합니다.
   - **EC2MetadataIp** 를 **Leaf1EC2MetadataIp** 로 변경합니다.
   - 네트워크 구성 스크립트를 **../scripts/run-os-net-config.sh** 에서 **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh**.sh로 변경합니다.
   - Leaf1 매개변수를 사용하도록 컨트롤 플레인/프로비저닝 인터페이스 수정.
   - Leaf1 경로를 포함하도록 각 VLAN을 수정합니다.

   완료되면 이 파일을 저장합니다.
10. **compute2.yaml** 을 편집하고 동일한 단계를 수행합니다. 다음은 변경 사항 목록입니다.
    - **ControlPlaneSubnetCidr** 를 **ControlPlane2SubnetCidr** 로 변경합니다.
    - **ControlPlaneDefaultRoute** 를 **ControlPlane2DefaultRoute** 로 변경합니다.
    - **EC2MetadataIp** 를 **Leaf2EC2MetadataIp** 로 변경합니다.
    - 네트워크 구성 스크립트를 **../scripts/run-os-net-config.sh** 에서 **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh**.sh로 변경합니다.
    - Leaf2 매개변수를 사용하도록 컨트롤 플레인/프로비저닝 인터페이스를 수정합니다.
    - Leaf2 경로를 포함하도록 각 VLAN 수정.

    완료되면 이 파일을 저장합니다.

4.6. 사용자 정의 CEPH STORAGE NIC 구성 생성

이 절차에서는 Leaf0, Leaf1 및 Leaf2에 Ceph Storage 노드에 대한 YAML 구조를 생성합니다.

절차

1. 사용자 정의 NIC 디렉터리로 변경합니다.

```
$ cd ~/templates/spine-leaf-nics/
```

2. 매개변수 섹션의 **ControlPlaneSubnetCidr** 및 **ControlPlaneDefaultRoute 매개변수** 로 스크롤합니다. 이러한 매개변수는 다음 스니펫과 유사합니다.

```
  ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlaneDefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
```

Leaf0에 맞게 이러한 매개변수를 수정합니다.

```
  ControlPlane0SubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
    type: string
  ControlPlane0DefaultRoute: # Override this via parameter_defaults
    description: The default route of the control plane network.
    type: string
```

3. 매개 변수 섹션의 **EC2MetadataIp 매개변수** 로 스크롤합니다. 이 매개변수는 다음 스니펫과 유사합니다.

```
  EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

Leaf0에 맞게 이 매개변수를 수정합니다.

```
  Leaf0EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
    type: string
```

4. 네트워크 구성 섹션으로 스크롤합니다. 이 섹션은 다음 스니펫과 유사합니다.

```
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:
```

스크립트의 위치를 절대 경로로 변경합니다.

```
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:
```

5. **network_config** 섹션에서 컨트롤 플레인 / 프로비저닝 인터페이스를 정의합니다. 예를 들면 다음과 같습니다

```
            network_config:
            - type: interface
              name: nic1
              use_dhcp: false
              dns_servers:
                get_param: DnsServers
              addresses:
              - ip_netmask:
                  list_join:
                  - /
                  - - get_param: ControlPlaneIp
                    - get_param: ControlPlane0SubnetCidr
              routes:
              - ip_netmask: 169.254.169.254/32
                next_hop:
                  get_param: Leaf0EC2MetadataIp
              - ip_netmask: 192.168.10.0/24
                next_hop:
                  get_param: ControlPlane0DefaultRoute
```

이 경우 사용되는 매개변수는 Leaf0: **ControlPlane0SubnetCidr**,**Leaf0EC2MetadataIp**, **ControlPlane0DefaultRoute** 에 고유합니다. 또한 프로비저닝 네트워크(192.168.10.0/24)의 Leaf0에 CIDR을 사용하여 경로로 사용됩니다.

6. **members** 섹션의 각 VLAN에는 관련 Leaf0 매개변수가 포함되어 있습니다. 예를 들어 스토리지 네트워크 VLAN 정보는 다음 스니펫과 유사해야 합니다.

```
            - type: vlan
              vlan_id:
                get_param: Storage0NetworkVlanID
```

```
        addresses:
        - ip_netmask:
            get_param: Storage0IpSubnet
```

라우팅 매개 변수를 정의하는 섹션을 추가합니다. 여기에는 슈퍼넷 경로(이 경우**StorageSupernet**)와 리프 기본 경로(이 경우 **Storage0InterfaceDefaultRoute**)가 포함됩니다.

```
      - type: vlan
        vlan_id:
          get_param: Storage0NetworkVlanID
        addresses:
        - ip_netmask:
            get_param: Storage0IpSubnet
        routes:
        - ip_netmask:
            get_param: StorageSupernet
          next_hop:
            get_param: Storage0InterfaceDefaultRoute
```

다음 컨트롤러 네트워크에 대한 VLAN 구조를 추가합니다. 스토리지 , **Storage Mgmt**.

7. 이 파일을 저장합니다.

8. **ceph-storage1.yaml** 을 편집하고 동일한 단계를 수행합니다. 다음은 변경 사항 목록입니다.

   - **ControlPlaneSubnetCidr** 를 **ControlPlane1SubnetCidr** 로 변경합니다.
   - **ControlPlaneDefaultRoute** 를 **ControlPlane1DefaultRoute** 로 변경합니다.
   - **EC2MetadataIp** 를 **Leaf1EC2MetadataIp** 로 변경합니다.
   - 네트워크 구성 스크립트를 **../scripts/run-os-net-config.sh** 에서 **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh**.sh로 변경합니다.
   - Leaf1 매개변수를 사용하도록 컨트롤 플레인/프로비저닝 인터페이스를 수정합니다.
   - Leaf1 경로를 포함하도록 각 VLAN을 수정합니다.

   완료되면 이 파일을 저장합니다.

9. **ceph-storage2.yaml** 을 편집하고 동일한 단계를 수행합니다. 다음은 변경 사항 목록입니다.

   - **ControlPlaneSubnetCidr** 를 **ControlPlane2SubnetCidr** 로 변경합니다.
   - **ControlPlaneDefaultRoute** 를 **ControlPlane2DefaultRoute** 로 변경합니다.
   - **EC2MetadataIp** 를 **Leaf2EC2MetadataIp** 로 변경합니다.
   - 네트워크 구성 스크립트를 **../scripts/run-os-net-config.sh** 에서 **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh**.sh로 변경합니다.
   - Leaf2 매개변수를 사용하도록 컨트롤 플레인/프로비저닝 인터페이스를 수정합니다.
   - Leaf2 경로를 포함하도록 각 VLAN을 수정합니다.

   완료되면 이 파일을 저장합니다.

4.7. 네트워크 환경 파일 만들기

이 절차에서는 나중에 사용할 기본 네트워크 환경 파일을 생성합니다.

절차

1. stack 사용자의 **templates** 디렉터리에 **network-environment.yaml** 파일을 생성합니다.
2. 환경 파일에 다음 섹션을 추가합니다.

```
resource_registry:

parameter_defaults:
```

   다음을 확인합니다.

   - **resource_registry** 는 네트워킹 리소스를 해당 NIC 템플릿에 매핑합니다.
   - **parameter_defaults** 는 구성과 관련된 추가 네트워킹 매개 변수를 정의합니다.

다음 두 섹션은 스파인 리프 아키텍처의 특정 측면을 구성하기 위해 네트워크 환경 파일에 세부 정보를 추가합니다. 완료되면 **openstack overcloud deploy** 명령을 사용하여 이 파일을 포함합니다.

4.8. NIC 템플릿에 네트워크 리소스 매핑

이 절차에서는 네트워크 구성에 대한 관련 리소스를 해당 NIC 템플릿에 매핑합니다.

절차

1. **network-environment.yaml** 파일을 편집합니다.
2. **resource_registry** 에 리소스 매핑을 추가합니다. 리소스 이름은 다음 형식을 사용합니다.

```
OS::TripleO::[ROLE]::Net::SoftwareConfig: [NIC TEMPLATE]
```

   이 가이드의 시나리오에서 **resource_registry** 에는 다음과 같은 리소스 매핑이 포함됩니다.

```
resource_registry:
  OS::TripleO::Controller0::Net::SoftwareConfig: ./spine-leaf-nics/controller0.yaml
  OS::TripleO::Compute0::Net::SoftwareConfig: ./spine-leaf-nics/compute0.yaml
  OS::TripleO::Compute1::Net::SoftwareConfig: ./spine-leaf-nics/compute1.yaml
  OS::TripleO::Compute2::Net::SoftwareConfig: ./spine-leaf-nics/compute2.yaml
  OS::TripleO::CephStorage0::Net::SoftwareConfig: ./spine-leaf-nics/cephstorage0.yaml
  OS::TripleO::CephStorage1::Net::SoftwareConfig: ./spine-leaf-nics/cephstorage1.yaml
  OS::TripleO::CephStorage2::Net::SoftwareConfig: ./spine-leaf-nics/cephstorage2.yaml
```

3. **network-environment.yaml** 파일을 저장합니다.

4.9. 스파인 리프 라우팅

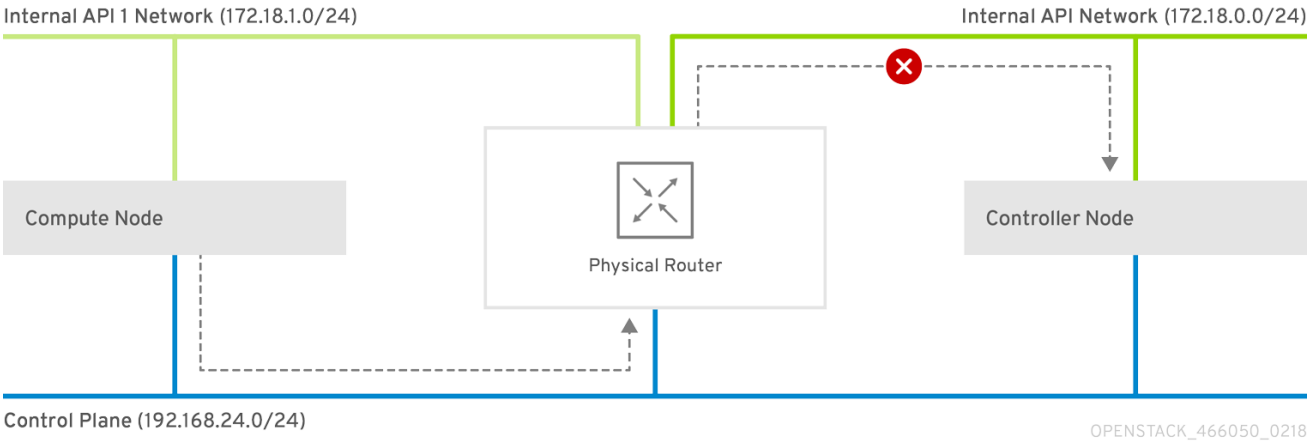
각 역할에는 동일한 기능에 사용되는 다른 서브넷을 가리키는 분리된 각 네트워크의 경로가 필요합니다. 따라서 *Compute1* 노드가 **InternalApi** VIP의 컨트롤러에 연결되면 트래픽은 **InternalApi1** 게이트웨이를 통해 **InternalApi1** 인터페이스를 대상으로 합니다. 결과적으로 컨트롤러에서 **InternalApi1** 네트워크로의 반환 트래픽이 **InternalApi** 네트워크 게이트웨이를 통과해야 합니다.

슈퍼넷 경로는 기본 게이트웨이를 통해 트래픽을 전송하지 않도록 각 역할의 모든 격리된 네트워크에 적용됩니다. 기본적으로 비컨트롤러의 *컨트롤* 플레인 네트워크와 컨트롤러의 *외부* 네트워크입니다.

Red Hat Enterprise Linux는 기본적으로 인바운드 트래픽에 대해 엄격한 역방향 경로 필터링을 구현하므로 격리된 네트워크에서 이러한 경로를 구성해야 합니다. API가 *내부 API* 인터페이스에서 수신하고 요청이 해당 API에 들어오는 경우 반환 경로 경로가 *내부 API* 인터페이스에 있는 경우에만 요청을 허용합니다. 서버가 *내부 API* 네트워크에서 수신 대기 중이지만 클라이언트의 반환 경로가 *컨트롤* 플레인을 통해 있는 경우 서버는 역방향 경로 필터로 인해 요청을 삭제합니다.

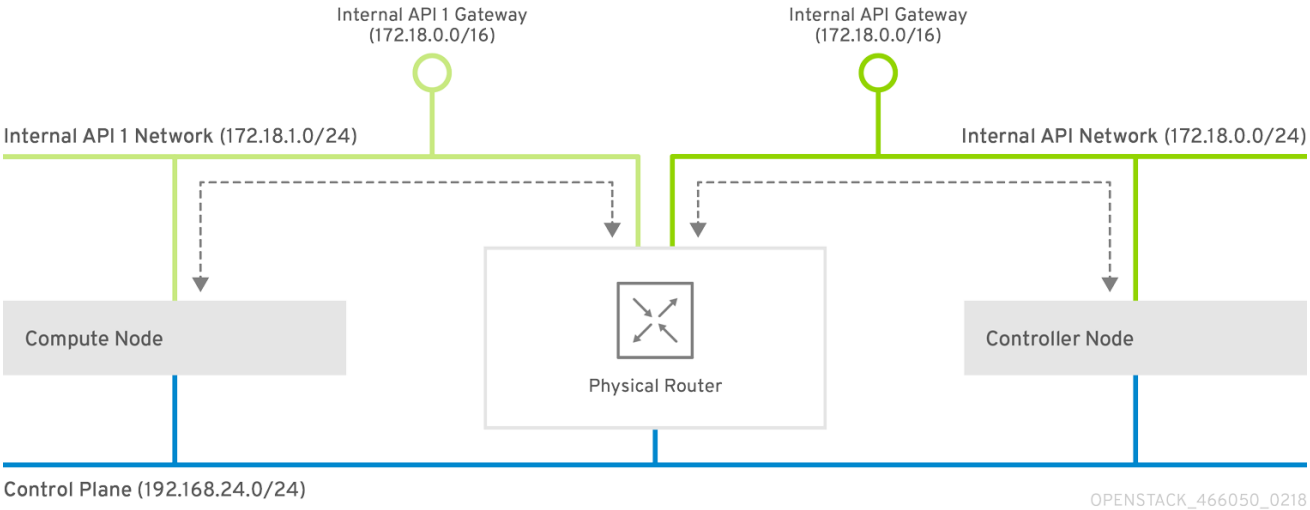
다음 다이어그램은 성공하지 못하는 컨트롤 플레인을 통해 트래픽을 라우팅하려는 시도를 보여줍니다. 라우터에서 컨트롤러 노드로의 반환 경로는 VIP가 수신 대기하는 인터페이스와 일치하지 않으므로 패킷이 삭제됩니다. **192.168.24.0/24** 는 컨트롤러에 직접 연결되므로 *컨트롤 플레인* 네트워크의 로컬으로 간주됩니다.

그림 4.1. 컨트롤 플레인을 통해 라우팅된 트래픽



이 다이어그램은 *내부 API* 네트워크를 통해 실행되는 라우팅을 보여줍니다.

그림 4.2. 내부 API를 통한 트래픽 라우팅



4.10. 구성 가능 네트워크에 경로 할당

이 절차에서는 리프 네트워크의 라우팅을 정의합니다.

절차

1. **network-environment.yaml** 파일을 편집합니다.
2. supernet 경로 매개 변수를 **parameter_defaults** 섹션에 추가합니다. 격리된 각 네트워크에는 슈퍼넷 경로가 적용되어야 합니다. 예를 들어 다음과 같습니다.

```
parameter_defaults:
  StorageSupernet: 172.16.0.0/16
  StorageMgmtSupernet: 172.17.0.0/16
  InternalApiSupernet: 172.18.0.0/16
  TenantSupernet: 172.19.0.0/16
```

참고

네트워크 인터페이스 템플릿에는 각 네트워크에 대한 슈퍼넷 매개 변수가 포함되어야 합니다. 예를 들어 다음과 같습니다.

```
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
      get_param: Storage0IpSubnet
  routes:
  - ip_netmask:
      get_param: StorageSupernet
    next_hop:
      get_param: Storage0InterfaceDefaultRoute
```

3. **ServiceNetMap HostnameResolveNetwork** 매개변수를 **parameter_defaults** 섹션에 추가하여 다른 리프 노드를 확인하는 데 사용할 호스트 이름 목록이 리프에 제공됩니다. 예를 들어 다음과 같습니다.

```
parameter_defaults:
  ...
  ServiceNetMap:
    Compute1HostnameResolveNetwork: internal_api1
    Compute2HostnameResolveNetwork: internal_api2
    Compute3HostnameResolveNetwork: internal_api3
    CephStorage1HostnameResolveNetwork: storage1
    CephStorage2HostnameResolveNetwork: storage2
    CephStorage3HostnameResolveNetwork: storage3
```

컴퓨팅 노드는 리프의 내부 API 네트워크를 사용하고 Ceph Storage 노드는 리프의 스토리지 네트워크를 사용합니다.

4. 다음 **ExtraConfig** 설정을 **parameter_defaults** 섹션에 추가하여 Compute 및 Ceph Storage 노드의 특정 구성 요소에 대한 라우팅을 처리합니다.

표 4.1. Compute ExtraConfig 매개변수

매개변수	이 값으로 설정
nova::compute::libvirt::vncserver_listen	VNC 서버가 청취하는 IP 주소입니다.
nova::compute::vncserver_proxyclient_address	VNC 프록시 클라이언트를 실행하는 서버의 IP 주소입니다.
neutron::agents::ml2::ovs::local_ip	OpenStack Networking(neutron) 터널 엔드포인트의 IP 주소.
cold_migration_ssh_inbound_addr	콜드 마이그레이션 SSH 연결을 위한 로컬 IP 주소.
live_migration_ssh_inbound_addr	실시간 마이그레이션 SSH 연결을 위한 로컬 IP 주소.
nova::migration::libvirt::live_migration_inbound_addr	실시간 마이그레이션 트래픽에 사용되는 IP 주소입니다. **참고** SSL/TLS를 사용하는 경우 네트워크 이름 앞에 "fqdn_"을 추가하여 인증서가 FQDN에 대해 확인되는지 확인합니다.
nova::my_ip	호스트의 Compute(nova) 서비스의 IP 주소입니다.
tripleo::profile::base::database::mysql::client::mysql_client_bind_address	데이터베이스 클라이언트의 IP 주소입니다. 이 경우 Compute 노드의 **mysql** 클라이언트입니다.

표 4.2. **CephAnsibleExtraConfig** parameters

매개변수	이 값으로 설정
public_network	Ceph 노드(프리프당 하나씩)가 포함된 모든 스토리지 네트워크의 쉼표로 구분된 목록(예: *172.16.0.0/24,172.16.1.0/24,172.16.2.0/24*)
cluster_network	Ceph 노드(프리프당 하나씩)가 포함된 스토리지 관리 네트워크의 쉼표로 구분된 목록(예: *172.17.0.0/24,172.17.1.0/24,172.17.2.0/24*)

예를 들어 다음과 같습니다.

```
parameter_defaults:
  ...
  Compute1ExtraConfig:
    nova::compute::libvirt::vncserver_listen: "%{hiera('internal_api1')}"
    nova::compute::vncserver_proxyclient_address: "%{hiera('internal_api1')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant1')}"
    cold_migration_ssh_inbound_addr: "%{hiera('internal_api1')}"
    live_migration_ssh_inbound_addr: "%{hiera('internal_api1')}"
    nova::migration::libvirt::live_migration_inbound_addr: "%{hiera('internal_api1')}"
    nova::my_ip: "%{hiera('internal_api1')}"
    tripleo::profile::base::database::mysql::client::mysql_client_bind_address: "%
{hiera('internal_api1')}"

  Compute2ExtraConfig:
    nova::compute::libvirt::vncserver_listen: "%{hiera('internal_api2')}"
    nova::compute::vncserver_proxyclient_address: "%{hiera('internal_api2')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant2')}"
    cold_migration_ssh_inbound_addr: "%{hiera('internal_api2')}"
    live_migration_ssh_inbound_addr: "%{hiera('internal_api2')}"
    nova::migration::libvirt::live_migration_inbound_addr: "%{hiera('internal_api2')}"
    nova::my_ip: "%{hiera('internal_api2')}"
    tripleo::profile::base::database::mysql::client::mysql_client_bind_address: "%
{hiera('internal_api2')}"

  Compute3ExtraConfig:
    nova::compute::libvirt::vncserver_listen: "%{hiera('internal_api3')}"
    nova::compute::vncserver_proxyclient_address: "%{hiera('internal_api3')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant3')}"
    cold_migration_ssh_inbound_addr: "%{hiera('internal_api3')}"
    live_migration_ssh_inbound_addr: "%{hiera('internal_api3')}"
    nova::migration::libvirt::live_migration_inbound_addr: "%{hiera('internal_api3')}"
    nova::my_ip: "%{hiera('internal_api3')}"
    tripleo::profile::base::database::mysql::client::mysql_client_bind_address: "%
{hiera('internal_api3')}"

  CephAnsibleExtraConfig:
    public_network: '172.16.0.0/24,172.16.1.0/24,172.16.2.0/24'
    cluster_network: '172.17.0.0/24,172.17.1.0/24,172.17.2.0/24'
```

4.11. 컨트롤 플레인 매개변수 설정

일반적으로 **network_data** 파일을 사용하여 격리된 스파인-리프형 네트워크에 대한 네트워킹 세부 정보를 정의합니다. 예외는 언더클라우드가 생성한 컨트롤 플레인 네트워크입니다. 그러나 오버클라우드는 각 리프마다 컨트롤 플레인에 액세스해야 합니다. 이를 위해서는 **network-environment.yaml** 파일에 정의된 몇 가지 추가 매개변수가 필요합니다. 예를 들어 다음 스니펫은 Leaf0에서 Controller 역할에 대한 예제 NIC 템플릿입니다.

```
- type: interface
  name: nic1
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
```

```
- ip_netmask:
    list_join:
    - /
    - - get_param: ControlPlaneIp
      - get_param: ControlPlane0SubnetCidr
routes:
- ip_netmask: 169.254.169.254/32
  next_hop:
    get_param: Leaf0EC2MetadataIp
- ip_netmask: 192.168.10.0/24
  next_hop:
    get_param: ControlPlane0DefaultRoute
```

이 경우 Leaf 0에서 각 컨트롤 플레인 네트워크에 대한 IP, 서브넷, 메타데이터 IP 및 기본 경로를 정의해야 합니다.

절차

1. **network-environment.yaml** 파일을 편집합니다.
2. **parameter_defaults** 섹션에서 다음을 수행합니다.
   a. 기본 컨트롤 플레인 서브넷에 매핑을 추가합니다.

   ```
   parameter_defaults:
     ...
     ControlPlaneSubnet: leaf0
   ```

   b. 각 스파인-리프형 네트워크에 대해 컨트롤 플레인 서브넷 매핑을 추가합니다.

   ```
   parameter_defaults:
     ...
     Controller0ControlPlaneSubnet: leaf0
     Compute0ControlPlaneSubnet: leaf0
     Compute1ControlPlaneSubnet: leaf1
     Compute2ControlPlaneSubnet: leaf2
     CephStorage0ControlPlaneSubnet: leaf0
     CephStorage1ControlPlaneSubnet: leaf1
     CephStorage2ControlPlaneSubnet: leaf2
   ```

   c. 각 리프의 컨트롤 플레인 경로를 추가합니다.

   ```
   parameter_defaults:
     ...
     ControlPlane0DefaultRoute: 192.168.10.1
     ControlPlane0SubnetCidr: '24'
     ControlPlane1DefaultRoute: 192.168.11.1
     ControlPlane1SubnetCidr: '24'
     ControlPlane2DefaultRoute: 192.168.12.1
     ControlPlane2SubnetCidr: '24'
   ```

   기본 경로 매개 변수는 일반적으로 각 프로비저닝 서브넷의 **게이트웨이에** 설정된 IP 주소입니다. 이 정보는 **undercloud.conf** 파일을 참조하십시오.

   d. EC2 메타데이터 IP에 대한 매개변수를 추가합니다.

```
parameter_defaults:
  ...
  Leaf0EC2MetadataIp: 192.168.10.1
  Leaf1EC2MetadataIp: 192.168.11.1
  Leaf2EC2MetadataIp: 192.168.12.1
```

이는 EC2 메타데이터 서비스(169.254.169.254/32)의 컨트롤 플레인을 통해 경로 역할을 하며 일반적으로 프로비저닝 네트워크의 각 리프의 각 **게이트웨이** 로 설정해야 합니다.

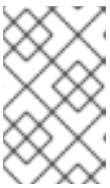
3. **network-environment.yaml** 파일을 저장합니다.

4.12. 스파인-리프가 활성화된 오버클라우드 배포

이제 모든 파일이 배포 준비가 완료되었습니다. 이 섹션에서는 각 파일과 배포 명령을 검토합니다.

절차

1. **/home/stack/template/network_data_spine_leaf.yaml** 파일을 검토하고 각 리프에 대한 각 네트워크가 포함되어 있는지 확인합니다.



참고

현재 네트워크 서브넷 및 **allocation_pools** 값에 대해 유효성 검사가 수행되지 않습니다. 이러한 정의를 일관되게 정의했는지 확인하고 기존 네트워크와 충돌하지 않습니다.

2. **~/templates/spine-leaf-nics/** 에 포함된 NIC 템플릿을 검토하고 각 리프에서 각 역할의 인터페이스가 올바르게 정의되어 있는지 확인합니다.

3. **network-environment.yaml** 환경 파일을 검토하고 네트워크 데이터 파일의 제어 범위를 벗어나는 모든 사용자 지정 매개변수가 포함되어 있는지 확인합니다. 여기에는 경로, 컨트롤 플레인 매개변수, 각 역할에 대한 사용자 정의 NIC 템플릿을 참조하는 **resource_registry** 섹션이 포함됩니다.

4. **/home/stack/templates/roles_data_spine_leaf.yaml** 값을 검토하고 각 리프에 대한 역할을 정의했는지 확인합니다.

5. '/home/stack/templates/nodes_data.yaml 파일을 확인하고 모든 역할에 할당된 플레이버 및 노드 수가 있는지 확인합니다. 또한 각 리프의 모든 노드가 올바르게 태그되어 있는지 확인합니다.

6. **openstack overcloud deploy** 명령을 실행하여 스파인 리프 구성을 적용합니다. 예를 들어 다음과 같습니다.

```
openstack overcloud deploy --templates \
-n /home/stack/template/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 /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/nodes_data.yaml \
-e [OTHER ENVIRONMENT FILES]
```

- **network-isolation.yaml** 은 동일한 위치에 있는 Jinja2 파일의 렌더링된 이름입니다 (**network-isolation.j2.yaml**). director가 각 네트워크를 올바른 리프로 분리하도록 하려면 이 파일을 포함합니다. 이렇게 하면 오버클라우드 생성 프로세스 중에 네트워크가 동적으로 생성됩니다.

- **network-isolation.yaml** 및 기타 네트워크 기반 환경 파일 뒤에 **network-environment.yaml** 파일을 포함합니다. 이렇게 하면 **network-environment.yaml** 에 정의된 매개변수 및 리소스가 다른 환경 파일에 이전에 정의된 동일한 매개변수 및 리소스를 덮어씁니다.

- 추가 환경 파일을 추가합니다. 예를 들어 컨테이너 이미지 위치 또는 Ceph 클러스터 구성이 있는 환경 파일입니다.

7. 스파인-리프가 활성화된 오버클라우드가 배포될 때까지 기다립니다.

부록 A. NETWORK_DATA 파일 예

```
# Storage
- name: Storage
  vip: true
  name_lower: storage
  ip_subnet: '172.16.0.0/24'
  allocation_pools: [{'start': '172.16.0.4', 'end': '172.16.0.250'}]
- name: Storage1
  vip: false
  name_lower: storage1
  ip_subnet: '172.16.1.0/24'
  allocation_pools: [{'start': '172.16.1.4', 'end': '172.16.1.250'}]
- name: Storage2
  vip: false
  name_lower: storage2
  ip_subnet: '172.16.2.0/24'
  allocation_pools: [{'start': '172.16.2.4', 'end': '172.16.2.250'}]

# StorageMgmt
- name: StorageMgmt
  name_lower: storage_mgmt
  vip: true
  ip_subnet: '172.17.0.0/24'
  allocation_pools: [{'start': '172.17.0.0', 'end': '172.17.0.250'}]
- name: StorageMgmt1
  name_lower: storage_mgmt1
  vip: false
  ip_subnet: '172.17.1.0/24'
  allocation_pools: [{'start': '172.17.1.4', 'end': '172.17.1.250'}]
- name: StorageMgmt2
  name_lower: storage_mgmt2
  vip: false
  ip_subnet: '172.17.2.0/24'
  allocation_pools: [{'start': '172.17.2.4', 'end': '172.17.2.250'}]

# Internal API
- name: InternalApi
  name_lower: internal_api
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: false
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: false
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]

# Tenant
- name: Tenant
```

```
  vip: false  # Tenant network does not use VIPs
  name_lower: tenant
  ip_subnet: '172.19.0.0/24'
  allocation_pools: [{'start': '172.19.0.4', 'end': '172.19.0.250'}]
- name: Tenant1
  vip: false  # Tenant network does not use VIPs
  name_lower: tenant1
  ip_subnet: '172.19.1.0/24'
  allocation_pools: [{'start': '172.19.1.4', 'end': '172.19.1.250'}]
- name: Tenant2
  vip: false  # Tenant network does not use VIPs
  name_lower: tenant2
  ip_subnet: '172.19.2.0/24'
  allocation_pools: [{'start': '172.19.2.4', 'end': '172.19.2.250'}]

- name: External
  vip: true
  name_lower: external
  ip_subnet: '10.0.0.0/24'
  allocation_pools: [{'start': '10.0.0.4', 'end': '10.0.0.250'}]
  gateway_ip: '10.0.0.1'
```

부록 B. 사용자 정의 NIC 템플릿

다음은 스파인 리프 네트워킹을 위한 네트워크 인터페이스 템플릿 구성을 시작하는 템플릿입니다. **resources** 섹션은 불완전하며 인터페이스 정의가 필요합니다.

```
heat_template_version: queens

parameters:
  # Supernets
  StorageSupernet:
    type: string
  StorageMgmtSupernet:
    type: string
  InternalApiSupernet:
    type: string
  TenantSupernet:
    type: string
  ExternalSupernet:
    type: string

  # Default Routes
  ControlPlane0DefaultRoute:
    type: string
  ControlPlane1DefaultRoute:
    type: string
  ControlPlane2DefaultRoute:
    type: string
  StorageInterfaceDefaultRoute:
    type: string
  Storage1InterfaceDefaultRoute:
    type: string
  Storage2InterfaceDefaultRoute:
    type: string
  StorageMgmtInterfaceDefaultRoute:
    type: string
  StorageMgmt1InterfaceDefaultRoute:
    type: string
  StorageMgmt2InterfaceDefaultRoute:
    type: string
  InternalApiInterfaceDefaultRoute:
    type: string
  InternalApi1InterfaceDefaultRoute:
    type: string
  InternalApi2InterfaceDefaultRoute:
    type: string
  TenantInterfaceDefaultRoute:
    type: string
  Tenant1InterfaceDefaultRoute:
    type: string
  Tenant2InterfaceDefaultRoute:
    type: string
  ExternalInterfaceDefaultRoute:
    type: string

  # IP subnets
```

```
  StorageIpSubnet:
    default: ''
    type: string
  Storage1IpSubnet:
    default: ''
    type: string
  Storage2IpSubnet:
    default: ''
    type: string
  StorageMgmtIpSubnet:
    default: ''
    type: string
  StorageMgmt1IpSubnet:
    default: ''
    type: string
  StorageMgmt2IpSubnet:
    default: ''
    type: string
  InternalApiIpSubnet:
    default: ''
    type: string
  InternalApi1IpSubnet:
    default: ''
    type: string
  InternalApi2IpSubnet:
    default: ''
    type: string
  TenantIpSubnet:
    default: ''
    type: string
  Tenant1IpSubnet:
    default: ''
    type: string
  Tenant2IpSubnet:
    default: ''
    type: string
  ExternalIpSubnet:
    default: ''
    type: string
  ManagementIpSubnet:
    default: ''
    type: string

  # VLAN IDs
  StorageNetworkVlanID:
    type: number
  Storage1NetworkVlanID:
    type: number
  Storage2NetworkVlanID:
    type: number
  StorageMgmtNetworkVlanID:
    type: number
  StorageMgmt1NetworkVlanID:
    type: number
  StorageMgmt2NetworkVlanID:
    type: number
```

```
  InternalApiNetworkVlanID:
    type: number
  InternalApi1NetworkVlanID:
    type: number
  InternalApi1NetworkVlanID:
    type: number
  TenantNetworkVlanID:
    type: number
  Tenant1NetworkVlanID:
    type: number
  Tenant2NetworkVlanID:
    type: number
  ExternalNetworkVlanID:
    type: number
  ManagementNetworkVlanID:
    type: number

  # Subnet CIDR
  ControlPlane0SubnetCidr:
    type: string
  ControlPlane1SubnetCidr:
    type: string
  ControlPlane1SubnetCidr:
    type: string

  ControlPlaneIp:
    type: string
  DnsServers:
    type: comma_delimited_list

  # EC2 metadata server IPs
  Leaf0EC2MetadataIp:
    type: string
  Leaf1EC2MetadataIp:
    type: string
  Leaf2EC2MetadataIp:
    type: string

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:
                [NETWORK CONFIG HERE]

outputs:
  OS::stack_id:
```

```
  description: The OsNetConfigImpl resource.
  value:
    get_resource: OsNetConfigImpl
```

부록 C. ROLES_DATA 파일의 예

```
###############################################################################
# Role: Controller0                                                           #
###############################################################################
- name: Controller0
  description: |
    Controller role that has all the controler services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
  default_route_networks: ['External']
  HostnameFormatDefault: '%stackname%-controller0-%index%'
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AodhApi
    - OS::TripleO::Services::AodhEvaluator
    - OS::TripleO::Services::AodhListener
    - OS::TripleO::Services::AodhNotifier
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::BarbicanApi
    - OS::TripleO::Services::BarbicanBackendSimpleCrypto
    - OS::TripleO::Services::BarbicanBackendDogtag
    - OS::TripleO::Services::BarbicanBackendKmip
    - OS::TripleO::Services::BarbicanBackendPkcs11Crypto
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CeilometerApi
    - OS::TripleO::Services::CeilometerCollector
    - OS::TripleO::Services::CeilometerExpirer
    - OS::TripleO::Services::CeilometerAgentCentral
    - OS::TripleO::Services::CeilometerAgentNotification
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephMds
    - OS::TripleO::Services::CephMgr
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephRbdMirror
    - OS::TripleO::Services::CephRgw
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::CinderApi
    - OS::TripleO::Services::CinderBackendDellPs
    - OS::TripleO::Services::CinderBackendDellSc
    - OS::TripleO::Services::CinderBackendDellEMCUnity
```

```
- OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
- OS::TripleO::Services::CinderBackendDellEMCVNX
- OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
- OS::TripleO::Services::CinderBackendNetApp
- OS::TripleO::Services::CinderBackendScaleIO
- OS::TripleO::Services::CinderBackendVRTSHyperScale
- OS::TripleO::Services::CinderBackup
- OS::TripleO::Services::CinderHPELeftHandISCSI
- OS::TripleO::Services::CinderScheduler
- OS::TripleO::Services::CinderVolume
- OS::TripleO::Services::Clustercheck
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::Congress
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Ec2Api
- OS::TripleO::Services::Etcd
- OS::TripleO::Services::ExternalSwiftProxy
- OS::TripleO::Services::Fluentd
- OS::TripleO::Services::GlanceApi
- OS::TripleO::Services::GlanceRegistry
- OS::TripleO::Services::GnocchiApi
- OS::TripleO::Services::GnocchiMetricd
- OS::TripleO::Services::GnocchiStatsd
- OS::TripleO::Services::HAproxy
- OS::TripleO::Services::HeatApi
- OS::TripleO::Services::HeatApiCloudwatch
- OS::TripleO::Services::HeatApiCfn
- OS::TripleO::Services::HeatEngine
- OS::TripleO::Services::Horizon
- OS::TripleO::Services::Ipsec
- OS::TripleO::Services::IronicApi
- OS::TripleO::Services::IronicConductor
- OS::TripleO::Services::IronicPxe
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Keepalived
- OS::TripleO::Services::Kernel
- OS::TripleO::Services::Keystone
- OS::TripleO::Services::LoginDefs
- OS::TripleO::Services::ManilaApi
- OS::TripleO::Services::ManilaBackendCephFs
- OS::TripleO::Services::ManilaBackendIsilon
- OS::TripleO::Services::ManilaBackendNetapp
- OS::TripleO::Services::ManilaBackendUnity
- OS::TripleO::Services::ManilaBackendVNX
- OS::TripleO::Services::ManilaBackendVMAX
- OS::TripleO::Services::ManilaScheduler
- OS::TripleO::Services::ManilaShare
- OS::TripleO::Services::Memcached
- OS::TripleO::Services::MistralApi
- OS::TripleO::Services::MistralEngine
- OS::TripleO::Services::MistralExecutor
- OS::TripleO::Services::MistralEventEngine
- OS::TripleO::Services::MongoDb
- OS::TripleO::Services::MySQL
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronApi
```

```
    - OS::TripleO::Services::NeutronBgpVpnApi
    - OS::TripleO::Services::NeutronSfcApi
    - OS::TripleO::Services::NeutronCorePlugin
    - OS::TripleO::Services::NeutronDhcpAgent
    - OS::TripleO::Services::NeutronL2gwAgent
    - OS::TripleO::Services::NeutronL2gwApi
    - OS::TripleO::Services::NeutronL3Agent
    - OS::TripleO::Services::NeutronLbaasv2Agent
    - OS::TripleO::Services::NeutronLbaasv2Api
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronMetadataAgent
    - OS::TripleO::Services::NeutronML2FujitsuCfab
    - OS::TripleO::Services::NeutronML2FujitsuFossw
    - OS::TripleO::Services::NeutronOvsAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaApi
    - OS::TripleO::Services::NovaConductor
    - OS::TripleO::Services::NovaConsoleauth
    - OS::TripleO::Services::NovaIronic
    - OS::TripleO::Services::NovaMetadata
    - OS::TripleO::Services::NovaPlacement
    - OS::TripleO::Services::NovaScheduler
    - OS::TripleO::Services::NovaVncProxy
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OctaviaApi
    - OS::TripleO::Services::OctaviaDeploymentConfig
    - OS::TripleO::Services::OctaviaHealthManager
    - OS::TripleO::Services::OctaviaHousekeeping
    - OS::TripleO::Services::OctaviaWorker
    - OS::TripleO::Services::OpenDaylightApi
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::OVNDBs
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::Pacemaker
    - OS::TripleO::Services::PankoApi
    - OS::TripleO::Services::RabbitMQ
    - OS::TripleO::Services::Redis
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::SaharaApi
    - OS::TripleO::Services::SaharaEngine
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::SkydiveAnalyzer
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::SwiftProxy
    - OS::TripleO::Services::SwiftDispersion
    - OS::TripleO::Services::SwiftRingBuilder
    - OS::TripleO::Services::SwiftStorage
    - OS::TripleO::Services::Tacker
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
```

```
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::Zaqar
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: Compute0                                                             #
##############################################################################
- name: Compute0
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi
    - Tenant
    - Storage
  HostnameFormatDefault: '%stackname%-compute0-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
```

```
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: Compute1                                             #
##############################################################################
- name: Compute1
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi1
    - Tenant1
    - Storage1
  HostnameFormatDefault: '%stackname%-compute1-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
```

```
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: Compute2                                             #
##############################################################################
- name: Compute2
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi2
    - Tenant2
    - Storage2
  HostnameFormatDefault: '%stackname%-compute2-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
```

```
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Iscsid
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::NeutronBgpVpnBagpipe
    - OS::TripleO::Services::NeutronLinuxbridgeAgent
    - OS::TripleO::Services::NeutronVppAgent
    - OS::TripleO::Services::NovaCompute
    - OS::TripleO::Services::NovaLibvirt
    - OS::TripleO::Services::NovaMigrationTarget
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::OpenDaylightOvs
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::SkydiveAgent
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Vpp
    - OS::TripleO::Services::OVNController
    - OS::TripleO::Services::OVNMetadataAgent
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: CephStorage0                                                         #
##############################################################################
- name: CephStorage0
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage0
    - StorageMgmt0
  HostnameFormatDefault: '%stackname%-cephstorage0-%index%'
  uses_deprecated_params: False
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
```

```
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: CephStorage1                                                         #
##############################################################################
- name: CephStorage1
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage1
    - StorageMgmt1
  HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
  uses_deprecated_params: False
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCrond
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp
##############################################################################
# Role: CephStorage2                                                         #
##############################################################################
- name: CephStorage2
  description: |
```

```
  Ceph OSD Storage node role
networks:
  - Storage2
  - StorageMgmt2
HostnameFormatDefault: '%stackname%-cephstorage2-%index%'
uses_deprecated_params: False
deprecated_nic_config_name: 'ceph-storage.yaml'
ServicesDefault:
  - OS::TripleO::Services::Aide
  - OS::TripleO::Services::AuditD
  - OS::TripleO::Services::CACerts
  - OS::TripleO::Services::CephOSD
  - OS::TripleO::Services::CertmongerUser
  - OS::TripleO::Services::Collectd
  - OS::TripleO::Services::Docker
  - OS::TripleO::Services::Fluentd
  - OS::TripleO::Services::Ipsec
  - OS::TripleO::Services::Kernel
  - OS::TripleO::Services::LoginDefs
  - OS::TripleO::Services::MySQLClient
  - OS::TripleO::Services::Ntp
  - OS::TripleO::Services::ContainersLogrotateCrond
  - OS::TripleO::Services::Rhsm
  - OS::TripleO::Services::RsyslogSidecar
  - OS::TripleO::Services::Securetty
  - OS::TripleO::Services::SensuClient
  - OS::TripleO::Services::Snmp
  - OS::TripleO::Services::Sshd
  - OS::TripleO::Services::Timezone
  - OS::TripleO::Services::TripleoFirewall
  - OS::TripleO::Services::TripleoPackages
  - OS::TripleO::Services::Tuned
  - OS::TripleO::Services::Ptp
```

부록 D. NETWORK_ENVIRONMENT 파일 예

```
resource_registry:
  OS::TripleO::ControllerLeaf2::Net::SoftwareConfig: /home/stack/nics/controllerleaf2.yaml
  OS::TripleO::ComputeLeaf3::Net::SoftwareConfig: /home/stack/nics/computeleaf3.yaml
  OS::TripleO::ComputeLeaf4::Net::SoftwareConfig: /home/stack/nics/computeleaf4.yaml
parameter_defaults:
  ControlPlaneSubnet: leaf1
  ControllerLeaf2ControlPlaneSubnet: leaf2
  ComputeLeaf3ControlPlaneSubnet: leaf3
  ComputeLeaf4ControlPlaneSubnet: leaf4
  ControlPlaneDefaultRoute: 10.10.1.1
  ControlPlaneSubnetCidr: '24'
  ControlPlane2DefaultRoute: 10.10.2.1
  ControlPlane2SubnetCidr: '24'
  ControlPlane3DefaultRoute: 10.10.3.1
  ControlPlane3SubnetCidr: '24'
  ControlPlane4DefaultRoute: 10.10.4.1
  ControlPlane4SubnetCidr: '24'
  InternalApiSupernet: 10.20.0.0/16
  TenantSupernet: 10.30.0.0/16
  ProvisioningSupernet: 10.10.0.0/16
  EC2MetadataIp: 10.10.1.10
  Leaf2EC2MetadataIp: 10.10.1.10
  Leaf3EC2MetadataIp: 10.10.1.10
  Leaf4EC2MetadataIp: 10.10.1.10

  ServiceNetMap:
    ComputeLeaf3HostnameResolveNetwork: internal_api3
    ComputeLeaf4HostnameResolveNetwork: internal_api4

  ComputeLeaf3ExtraConfig:
    nova::compute::libvirt::vncserver_listen: "%{hiera('internal_api3')}"
    nova::compute::vncserver_proxyclient_address: "%{hiera('internal_api3')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant3')}"
    cold_migration_ssh_inbound_addr: "%{hiera('internal_api3')}"
    live_migration_ssh_inbound_addr: "%{hiera('internal_api3')}"
    nova::migration::libvirt::live_migration_inbound_addr: "%{hiera('internal_api3')}"
    nova::my_ip: "%{hiera('internal_api3')}"
    tripleo::profile::base::database::mysql::client::mysql_client_bind_address: "%
{hiera('internal_api3')}"

  ComputeLeaf4ExtraConfig:
    nova::compute::libvirt::vncserver_listen: "%{hiera('internal_api4')}"
    nova::compute::vncserver_proxyclient_address: "%{hiera('internal_api4')}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant4')}"
    cold_migration_ssh_inbound_addr: "%{hiera('internal_api4')}"
    live_migration_ssh_inbound_addr: "%{hiera('internal_api4')}"
    nova::migration::libvirt::live_migration_inbound_addr: "%{hiera('internal_api4')}"
    nova::my_ip: "%{hiera('internal_api4')}"
    tripleo::profile::base::database::mysql::client::mysql_client_bind_address: "%
{hiera('internal_api4')}"
```