Red Hat OpenStack Platform 17.1

스파인-리프 네트워킹 구성

Red Hat OpenStack Platform director를 사용하여 라우팅된 스파인-리프트 네트워크 구성

Last Updated: 2024-11-22

Red Hat OpenStack Platform 17.1 스파인-리프 네트워킹 구성

Red Hat OpenStack Platform director를 사용하여 라우팅된 스파인-리프트 네트워크 구성

OpenStack Team
rhos-docs@redhat.com

법적 공지

초록

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

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지 를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. Red Hat이 어떻게 더 나은지 알려주십시오.

Jira에서 문서 피드백 제공

Create Issue 양식을 사용하여 OpenShift (RHOSO) 또는 이전 Red Hat OpenStack Platform (RHOSP)의 Red Hat OpenStack Services 문서에 대한 피드백을 제공합니다. RHOSO 또는 RHOSP 문서에 대한 문제를 생성할 때 RHOSO Jira 프로젝트에 문제가 기록되어 피드백의 진행 상황을 추적할 수 있습니다.

문제 생성 양식을 완료하려면 Jira에 로그인해야 합니다. Red Hat Jira 계정이 없는 경우 https://issues.redhat.com 에서 계정을 생성할 수 있습니다.

1. 다음 링크를 클릭하여 **문제 생성** 페이지를 엽니다.
   https://issues.redhat.com/secure/CreateIssueDetails!init.jspa?pid=12336920&summary=Documentation%20feedback:%20%3CAdd%20summary%20here%3E&i
   <Include+the+documentation+URL,+the%20chapter+or+section+number,+and+a+detailed+descrip

2. **요약** 및 **설명** 필드를 작성합니다. **설명** 필드에 문서 URL, 장 또는 섹션 번호, 문제에 대한 자세한 설명을 포함합니다. 양식의 다른 필드를 수정하지 마십시오.

3. **생성**을 클릭합니다.

1장. 스파인-리프 네트워킹 소개

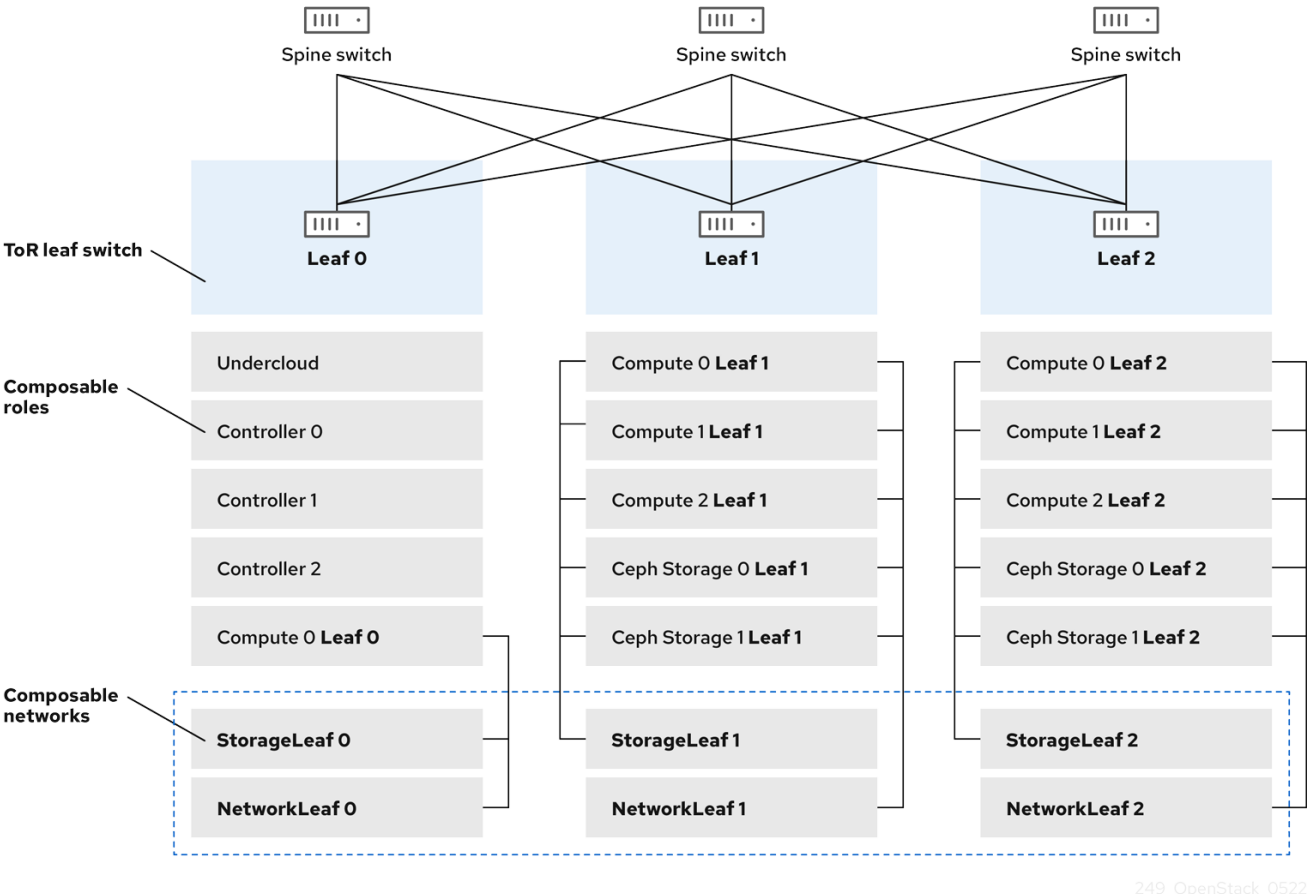
다음 장에서는 Red Hat OpenStack Platform 환경의 스파인-리프트 네트워크 토폴로지 구성에 대한 정보를 제공합니다. 여기에는 전체 엔드 투 엔드 시나리오와 예제 파일이 포함되며, 사용자 환경 내에서 보다 광범위한 네트워크 토폴로지를 복제할 수 있습니다.

1.1. 스파인-리프 네트워킹

Red Hat OpenStack Platform에는 라우팅된 스파인-리프트 데이터 센터 토폴로지에 네트워킹을 조정하는 데 사용할 수 있는 구성 가능 네트워크 아키텍처가 있습니다. 라우팅된 스파인-리프트의 실제 적용에서, 리프는 일반적으로 그림 1.1, "Routed spine-leaf 예제" 와 같이 데이터 센터 랙에서 구성 가능한 Compute 또는 Storage 역할로 표시됩니다. *Leaf 0* rack에는 언더클라우드 노드, 컨트롤러 노드 및 컴퓨팅 노드가 있습니다. 구성 가능 네트워크는 구성 가능 역할에 할당된 노드에 제공됩니다. 다음 다이어그램에는 다음 구성이 포함되어 있습니다.

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

그림 1.1. 라우팅된 스파인-리프트 예



1.2. SPINE-LEAF 네트워크 토폴로지

스파인-리프 시나리오에서는 OpenStack Networking(neutron) 기능을 활용하여 단일 네트워크의 세그먼트 내에서 여러 서브넷을 정의합니다. 각 네트워크는 Leaf 0으로 작동하는 기본 네트워크를 사용합니다. director는 기본 네트워크의 세그먼트로 Leaf 1 및 Leaf 2 서브넷을 생성합니다.

이 시나리오에서는 다음 네트워크를 사용합니다.

표 1.1. 리프 0 네트워크 (기본 네트워크)

네트워크	연결된 역할	subnet
프로비저닝 / Ctlplane / Leaf0	Controller, ComputeLeaf0, CephStorageLeaf0	192.168.10.0/24
스토리지	Controller, ComputeLeaf0, CephStorageLeaf0	172.16.0.0/24
StorageMgmt	Controller, CephStorageLeaf0	172.17.0.0/24
InternalApi	Controller, ComputeLeaf0	172.18.0.0/24
테넌트 [1]	Controller, ComputeLeaf0	172.19.0.0/24
외부	컨트롤러	10.1.1.0/24

[1] 테넌트 네트워크는 프로젝트 네트워크라고도 합니다.

표 1.2. 리프 1 네트워크

네트워크	연결된 역할	subnet
프로비저닝 / Ctlplane / Leaf1	ComputeLeaf1, CephStorageLeaf1	192.168.11.0/24
StorageLeaf1	ComputeLeaf1, CephStorageLeaf1	172.16.1.0/24
StorageMgmtLeaf1	CephStorageLeaf1	172.17.1.0/24
InternalApiLeaf1	ComputeLeaf1	172.18.1.0/24
TenantLeaf1 [1]	ComputeLeaf1	172.19.1.0/24

[1] 테넌트 네트워크는 프로젝트 네트워크라고도 합니다.

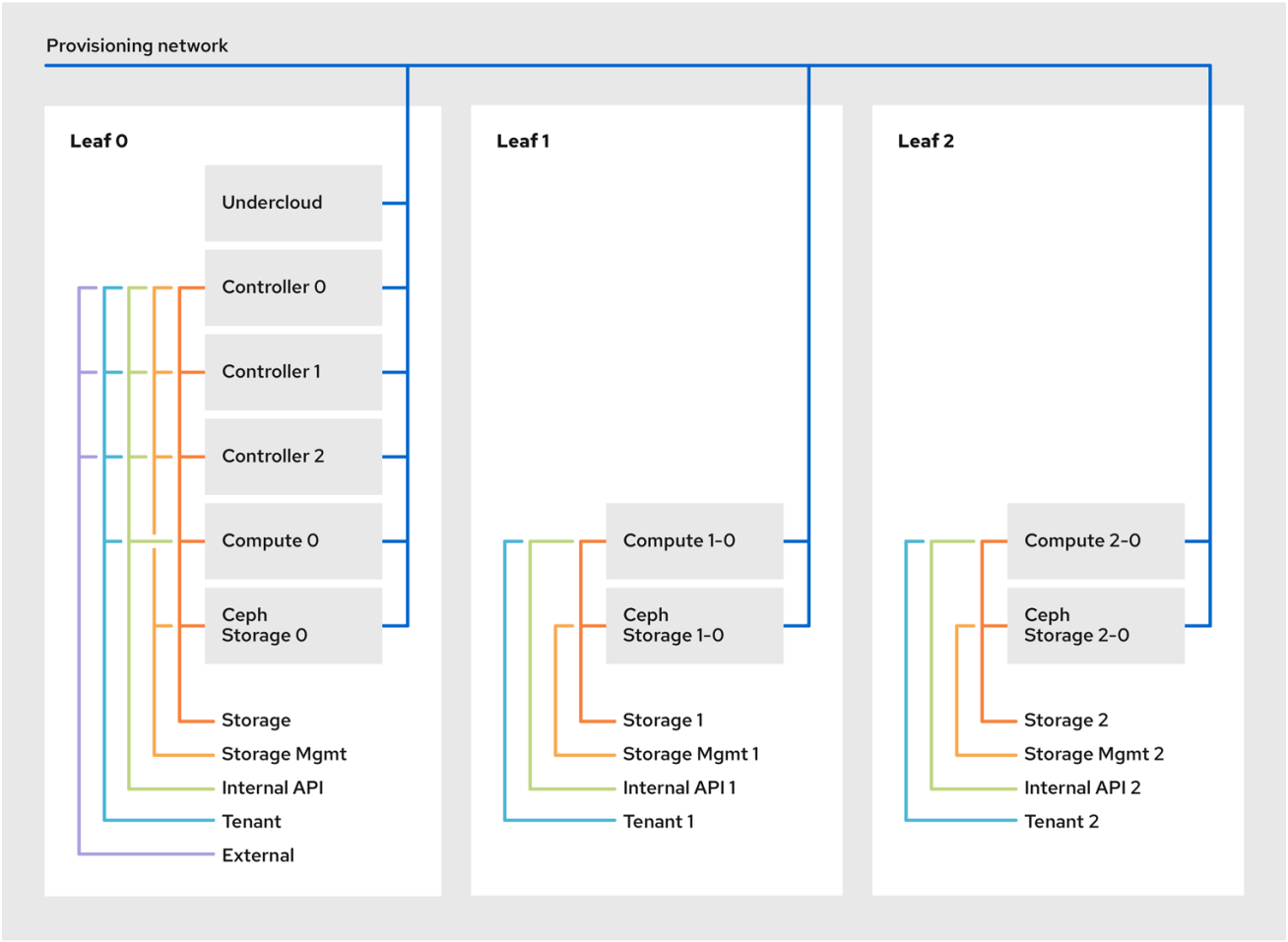
표 1.3. 리프 2 네트워크

네트워크	연결된 역할	subnet
프로비저닝 / Ctlplane / Leaf2	ComputeLeaf2, CephStorageLeaf2	192.168.12.0/24
StorageLeaf2	ComputeLeaf2, CephStorageLeaf2	172.16.2.0/24

네트워크	연결된 역할	subnet
StorageMgmtLeaf2	CephStorageLeaf2	172.17.2.0/24
InternalApiLeaf2	ComputeLeaf2	172.18.2.0/24
TenantLeaf2 [1]	ComputeLeaf2	172.19.2.0/24

[1] 테넌트 네트워크는 프로젝트 네트워크라고도 합니다.

그림 1.2. spine-leaf 네트워크 토폴로지



249_OpenStack_0522

1.3. SPINE-LEAF 요구 사항

L3 라우팅 아키텍처가 있는 네트워크에 오버클라우드를 배포하려면 다음 사전 요구 사항 단계를 완료합니다.

Layer-3 라우팅

다른 L2 세그먼트 간 트래픽을 활성화하도록 네트워크 인프라의 라우팅을 구성합니다. 이 라우팅을 정적 또는 동적으로 구성할 수 있습니다.

DHCP-Relay

언더클라우드의 로컬이 아닌 각 L2 세그먼트는 **dhcp-relay** 를 제공해야 합니다. 언더클라우드가 연결된 프로비저닝 네트워크 세그먼트에서 DHCP 요청을 언더클라우드로 전달해야 합니다.

참고

언더클라우드는 두 개의 DHCP 서버를 사용합니다. 하나는 베어 메탈 노드 인트로스펙션, 다른 하나는 오버클라우드 노드 배포를 위한 것입니다. **dhcp-relay**.

1.4. SPINE-LEAF 제한 사항

- Controller 역할과 같은 일부 역할은 가상 IP 주소 및 클러스터링을 사용합니다. 이 기능의 메커니즘에는 이러한 노드 간 L2 네트워크 연결이 필요합니다. 이러한 노드를 동일한 리프 내에 배치해야 합니다.
- 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 컴퓨팅 노드 간에 특정 VLAN을 공유해야 합니다.

참고

여러 네트워크 노드 세트로 네트워크 서비스를 구성할 수 있습니다. 각 네트워크 노드 세트는 네트워크의 경로를 공유하고 VRRP는 각 네트워크 노드 세트 내에서 고가용성 기본 경로를 제공합니다. 이러한 유형의 구성에서는 네트워크를 공유하는 모든 Networker 노드가 동일한 L2 네트워크 세그먼트에 있어야 합니다.

2장. 언더클라우드에서 라우팅된 스파인-리프트 구성

이 섹션에서는 구성 가능 네트워크를 사용하여 라우팅된 스파인-리프트를 수용하도록 언더클라우드를 구성하는 방법에 대해 설명합니다.

2.1. 스파인 리프 프로비저닝 네트워크 구성

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

절차

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

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

3. **undercloud.conf** 파일을 편집합니다.
4. **[DEFAULT]** 섹션에 다음 값을 설정합니다.
   a. **local_ip** 를 **leaf0** 의 언더클라우드 IP로 설정합니다.

```
local_ip = 192.168.10.1/24
```

   b. **undercloud_public_host** 를 언더클라우드의 외부 연결 IP 주소로 설정합니다.

```
undercloud_public_host = 10.1.1.1
```

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

```
undercloud_admin_host = 192.168.10.2
```

   d. **local_interface** 를 로컬 네트워크의 bridge로 설정합니다.

```
local_interface = eth1
```

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

```
enable_routed_networks = true
```

   f. **subnets** 매개변수를 사용하여 서브넷 목록을 정의합니다. 라우트된 스파인 및 리프의 각 L2 세그먼트에 대해 하나의 서브넷을 정의합니다.

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

   g. **local_subnet** 매개변수를 사용하여 로컬에 있는 물리적 L2 세그먼트와 연결된 서브넷을 지정합니다.

```
local_subnet = leaf0
```

h. **undercloud_nameservers** 값을 설정합니다.

```
undercloud_nameservers = 10.11.5.19,10.11.5.20
```

작은 정보

/etc/resolv.conf에서 확인하여 언더클라우드 네임서버에 사용되는 DNS 서버의 현재 IP 주소를 찾을 수 있습니다.

5. **subnets** 매개변수에 정의된 각 서브넷에 대한 새 섹션을 만듭니다.

```
[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
```

이 구성은 provisioning 네트워크 또는 컨트롤 플레인에 세 개의 서브넷을 생성합니다. 오버클라우드는 각 네트워크를 사용하여 각 리프 내에 시스템을 프로비저닝합니다.

언더클라우드에 대한 DHCP 요청을 올바르게 릴레이하려면 DHCP 릴레이를 구성해야 할 수 있습니다.

2.2. DHCP 릴레이 구성

요청을 전달하려는 원격 네트워크 세그먼트에 연결된 스위치, 라우터 또는 서버에서 DHCP 릴레이 서비스를 실행합니다.

참고

언더클라우드에서 DHCP 릴레이 서비스를 실행하지 마십시오.

언더클라우드는 provisioning 네트워크에서 두 개의 DHCP 서버를 사용합니다.

- 인트로스펙션 DHCP 서버입니다.
- 프로비저닝 DHCP 서버.

DHCP 요청을 언더클라우드의 두 DHCP 서버로 전달하도록 DHCP 릴레이를 구성해야 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 유형의 DHCP 릴레이 구성은 이 문서의 범위를 벗어납니다. 이 문서에서는 ISC DHCP 소프트웨어의 구현을 사용하여 DHCP 릴레이 구성 예제를 제공합니다. 자세한 내용은 수동 페이지 dhcrelay(8)를 참조하십시오.

중요

DHCP 옵션 79는 일부 릴레이, 특히 DHCPv6 주소를 제공하는 릴레이 및 원래 MAC 주소에서 전달하지 않는 릴레이에 필요합니다. 자세한 내용은 RFC6939 를 참조하십시오.

브로드캐스트 DHCP 릴레이

이 방법은 DHCP 서버 또는 서버가 있는 L2 네트워크 세그먼트에 UDP 브로드캐스트 트래픽을 사용하여 DHCP 요청을 릴레이합니다. 네트워크 세그먼트의 모든 장치는 브로드캐스트 트래픽을 수신합니다. UDP 브로드캐스트를 사용하면 언더클라우드의 두 DHCP 서버가 릴레이된 DHCP 요청을 수신합니다. 구현에 따라 인터페이스 또는 IP 네트워크 주소를 지정하여 이를 구성할 수 있습니다.

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 언더클라우드의 인트로스펙션에 사용되는 인터페이스에 할당된 IP 주소와 OpenStack Networking(neutron) 서비스에서 생성한 네트워크 네임스페이스의 IP 주소와 **ctlplane** 네트워크의 DHCP 서비스를 호스팅하기 위해 DHCP 릴레이를 제공하는 장치를 구성해야 합니다.

인트로스펙션에 사용되는 인터페이스는 **undercloud.conf** 파일에서 **inspection_interface** 로 정의된 인터페이스입니다. 이 매개변수를 설정하지 않은 경우 언더클라우드의 기본 인터페이스는 **br-ctlplane** 입니다.

참고

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

Neutron DHCP 네임스페이스에 할당된 IP 주소는 **undercloud.conf** 파일의 **local_subnet** 에 대해 구성하는 IP 범위에서 사용 가능한 첫 번째 주소입니다. IP 범위의 첫 번째 주소는 구성에서 **dhcp_start** 로 정의된 주소입니다. 예를 들어 다음 구성을 사용하는 경우 **192.168.10.10** 은 IP 주소입니다.

```
[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 이 주소는 IP 범위의 첫 번째 주소입니다. 이 경우 언더클라우드에서 다음 명령을 실행합니다.

```
$ 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 구성의 예

다음 예제에서 **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 **192.168.10.1** 에서 수신 대기 중입니다.

- 프로비저닝에 사용되는 DHCP 서버는 IP 주소 **192.168.10.10** 에서 수신 대기 중입니다.

그러면 다음과 같은 **dhcrelay** 명령이 생성됩니다.

- **dhcrelay** 버전 4.2.x:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -i eth0 -i eth1 -i eth2 -i eth3
```

- **dhcrelay** 버전 4.3.x 이상:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -iu eth0 -id eth1 -id eth2 -id eth3
```

Cisco IOS 라우팅 스위치 구성의 예

이 예에서는 다음 Cisco IOS 구성을 사용하여 다음 작업을 수행합니다.

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

provisioning 네트워크를 구성했으므로 나머지 오버클라우드 리프 네트워크를 구성할 수 있습니다.

2.3. 리프 노드에 대한 역할 지정

각 리프 네트워크의 각 역할에는 해당 리프에 노드를 태그할 수 있도록 플레이버 및 역할 할당이 필요합니다. 각 플레이버를 만들고 역할에 할당하려면 다음 단계를 완료합니다.

절차

1. **stackrc** 파일을 소싱합니다.

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

2. 노드 목록을 검색하여 UUID를 확인합니다.

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

3. 리프 네트워크 및 역할을 식별하는 사용자 정의 리소스 클래스를 사용하여 역할에 지정할 각 베어 메탈 노드를 할당합니다.

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

- <ROLE>을 역할을 식별하는 이름으로 바꿉니다.
- <node>를 베어 메탈 노드의 ID로 바꿉니다.
  예를 들어 다음 명령을 입력하여 UUID 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13을 Leaf2 의 Compute 역할에 태그를 지정합니다.

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

4. 아직 정의되지 않은 경우 **overcloud-baremetal-deploy.yaml** 에 각 역할을 추가합니다.
5. 역할의 노드에 할당할 리소스 클래스를 정의합니다.

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

- <role>을 역할 이름으로 바꿉니다.
- <ROLE>을 역할을 식별하는 이름으로 바꿉니다.

6. baremetal-deploy.yaml 파일에서 역할의 노드에 할당할 리소스 클래스를 정의합니다. 배포하려는 역할, 프로필, 수량 및 관련 네트워크를 지정합니다.

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

- <role>을 역할 이름으로 바꿉니다.

- <ROLE>을 역할을 식별하는 이름으로 바꿉니다.

참고

배포하는 모든 스택에 대해 **baremetal-deploy.yaml** 환경 파일을 **/home/stack/<stack> 에서 생성해야 합니다**.

2.4. 베어 메탈 노드 포트를 컨트롤 플레인 네트워크 세그먼트에 매핑

L3 라우팅 네트워크에서 배포를 활성화하려면 베어 메탈 포트에서 **physical_network** 필드를 구성해야 합니다. 각 베어 메탈 포트는 OpenStack Bare Metal(ironic) 서비스의 베어 메탈 노드와 연결됩니다. 물리적 네트워크 이름은 언더클라우드 구성의 **subnets** 옵션에 포함된 이름입니다.

참고

undercloud.conf 파일에서 **local_subnet** 으로 지정된 서브넷의 물리적 네트워크 이름은 항상 **ctlplane** 입니다.

절차

1. **stackrc** 파일을 소싱합니다.

```
$ source ~/stackrc
```

2. 베어 메탈 노드를 확인합니다.

```
$ openstack baremetal node list
```

3. 베어 메탈 노드가 **등록** 또는 **관리 가능** 상태인지 확인합니다. 베어 메탈 노드가 이러한 상태 중 하나에 없는 경우 baremetal 포트에서 **physical_network** 속성을 설정하는 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 포트가 어떤 baremetal 노드와 연결되어 있는지 확인합니다.

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

5. 포트의 **physical-network** 매개변수를 설정합니다. 아래 예제에서 세 개의 서브넷은 **leaf0**,**leaf1**, **leaf2** 에 정의되어 있습니다. local_subnet은 **leaf0** 입니다. **local_subnet** 의 물리적 네트워크는 항상 **ctlplane** 이므로 **leaf0** 에 연결된 baremetal 포트는 ctlplane을 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>
```

6. 오버클라우드를 배포하기 전에 노드를 검사합니다. 배포를 위해 노드를 사용 가능한 대로 설정하려면 **--all-manageable** 및 **--provide** 옵션을 포함합니다.

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

2.5. 스파인-리프 프로비저닝 네트워크에 새 리프 추가

새 물리적 사이트 추가 포함 가능한 네트워크 용량을 늘릴 때 Red Hat OpenStack Platform 스파인-리프 프로비저닝 네트워크에 새 리프와 해당 서브넷을 추가해야 할 수 있습니다. 오버클라우드에서 리프를 프로비저닝하면 해당 언더클라우드 리프가 사용됩니다.

사전 요구 사항

- RHOSP 배포에서는 스파인-리프트 네트워크 토폴로지를 사용합니다.

절차

1. 언더클라우드 호스트에 stack 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **/home/stack/undercloud.conf** 파일에서 다음을 수행합니다.
   a. **subnets** 매개변수를 찾고 추가할 리프의 새 서브넷을 추가합니다.
   서브넷은 라우팅된 스파인 및 리프의 L2 세그먼트를 나타냅니다.

   예제

   이 예에서는 새 리프(**leaf3**)에 대해 새 서브넷(**leaf3**)이 추가됩니다.

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

   b. 추가한 서브넷에 대한 섹션을 생성합니다.

   예제

   이 예제에서는 새 서브넷(**leaf3**)에 대해 **[leaf3]** 섹션이 추가되었습니다.

```
[leaf0]
cidr = 192.168.10.0/24
dhcp_start = 192.168.10.10
dhcp_end = 192.168.10.90
inspection_iprange = 192.168.10.100,192.168.10.190
gateway = 192.168.10.1
masquerade = False

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
```

```
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False

[leaf3]
cidr = 192.168.13.0/24
dhcp_start = 192.168.13.10
dhcp_end = 192.168.13.90
inspection_iprange = 192.168.13.100,192.168.13.190
gateway = 192.168.13.1
masquerade = False
```

4. **undercloud.conf** 파일을 저장합니다.
5. 언더클라우드를 다시 설치합니다.

```
$ openstack undercloud install
```

추가 리소스

- 스파인-리프 배포에 새 리프 추가

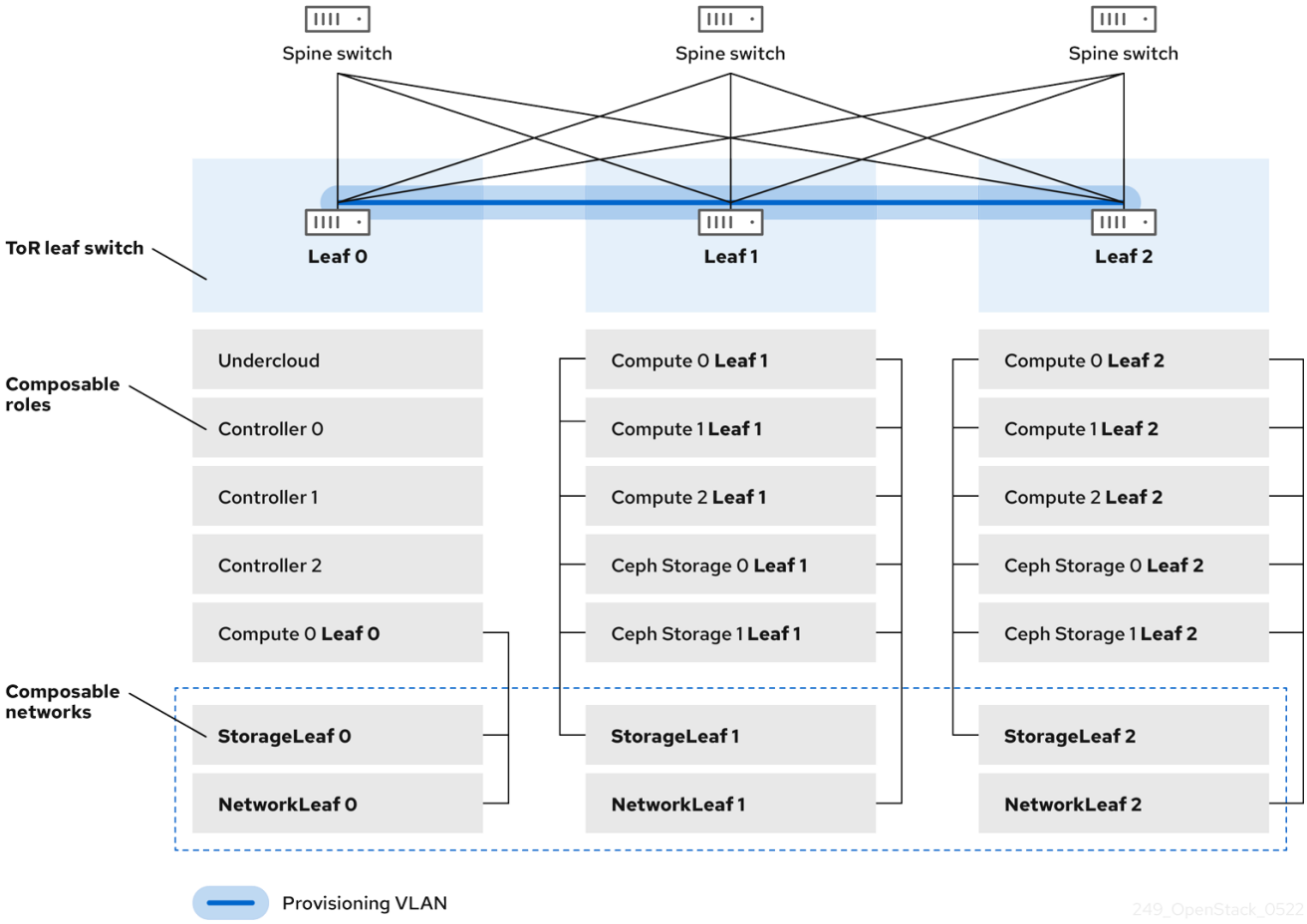
3장. 대체 프로비저닝 네트워크 방법

이 섹션에는 구성 가능 네트워크를 사용하여 라우팅된 스파인-리프트를 수용하도록 프로비저닝 네트워크를 구성하는 데 사용할 수 있는 다른 방법에 대한 정보가 포함되어 있습니다.

3.1. VLAN 프로비저닝 네트워크

이 예제에서 director는 provisioning 네트워크를 통해 새 오버클라우드 노드를 배포하고 L3 토폴로지에서 VLAN 터널을 사용합니다. 자세한 내용은 그림 3.1, "VLAN 프로비저닝 네트워크 토폴로지" 를 참조하십시오. VLAN 프로비저닝 네트워크를 사용하는 경우 director DHCP 서버에서 **DHCPOFFER** 브로드캐스트를 모든 리프로 보낼 수 있습니다. 이 터널을 설정하려면 Top-of-Rack(ToRack) 리프 스위치 간에 VLAN을 트렁크합니다. 다음 다이어그램에서 **StorageLeaf** 네트워크는 Ceph 스토리지 및 컴퓨팅 노드에 표시됩니다. **NetworkLeaf** 는 구성하려는 모든 네트워크의 예를 나타냅니다.

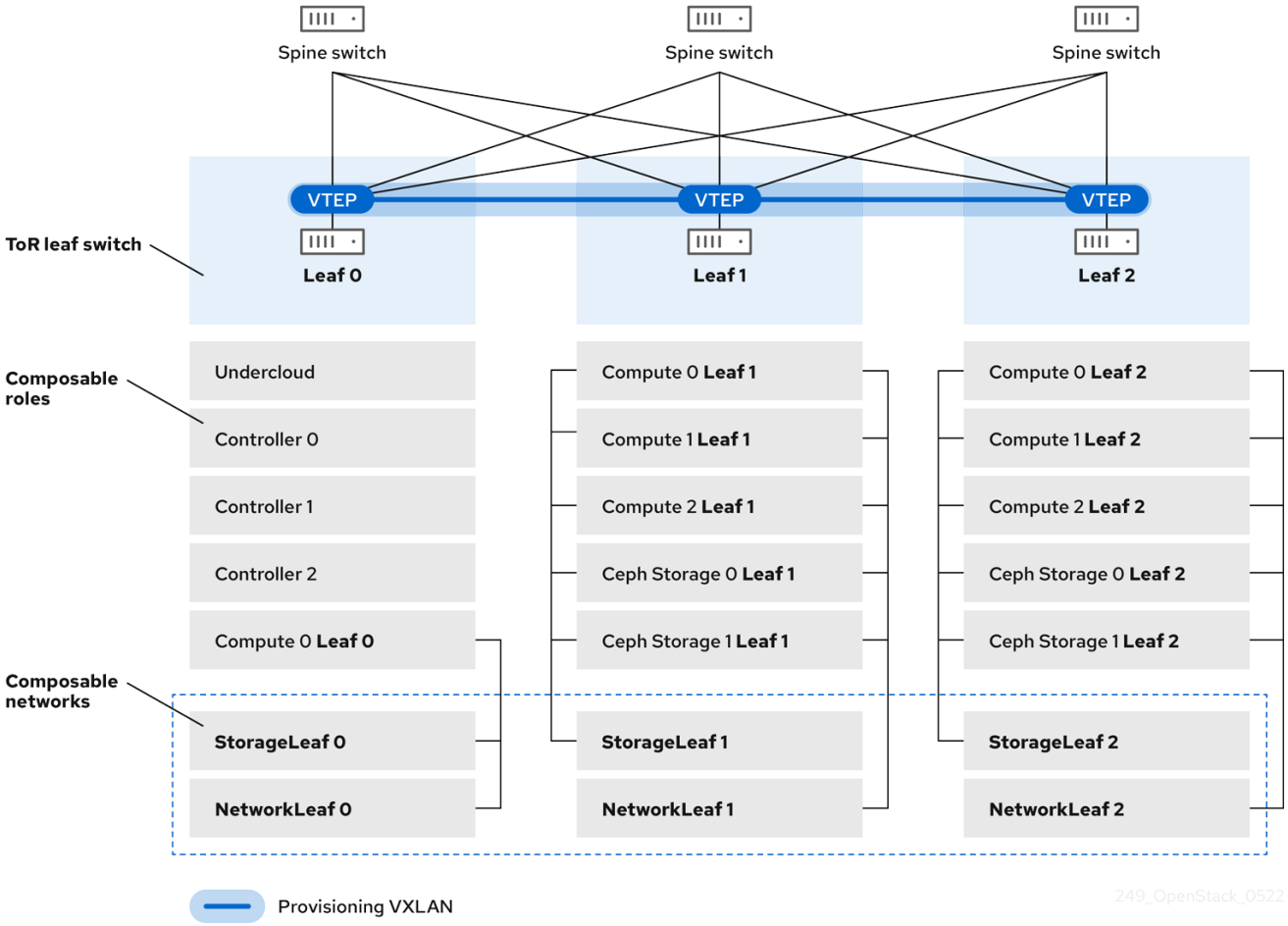
그림 3.1. VLAN 프로비저닝 네트워크 토폴로지



3.2. VXLAN 프로비저닝 네트워크

이 예제에서 director는 provisioning 네트워크를 통해 새 오버클라우드 노드를 배포하고 VXLAN 터널을 사용하여 계층 3 토폴로지에 걸쳐 있습니다. 자세한 내용은 그림 3.2, "VXLAN provisioning network topology" 를 참조하십시오. VXLAN 프로비저닝 네트워크를 사용하는 경우 director DHCP 서버에서 **DHCPOFFER** 브로드캐스트를 모든 리프로 보낼 수 있습니다. 이 터널을 설정하려면 Top-of-Rack(ToRack) 리프 스위치에서 VXLAN 끝점을 구성합니다.

그림 3.2. VXLAN 프로비저닝 네트워크 토폴로지



4장. 오버클라우드 설정

RHOSP(Red Hat OpenStack Platform) director를 사용하여 RHOSP 오버클라우드에 스파인 리프 네트워킹을 설치하고 구성합니다. 높은 수준의 단계는 다음과 같습니다.

1. 각 리프의 오버클라우드 네트워크를 정의합니다.
2. 각 리프에 대한 구성 가능 역할을 생성하고 구성 가능 네트워크를 각 역할에 연결합니다 .
3. 각 역할에 대한 고유한 NIC 구성을 생성합니다.
4. 각 리프가 해당 리프의 특정 브리지 또는 VLAN을 통해 트래픽을 라우팅하도록 브리지 매핑을 변경합니다.
5. 오버클라우드 엔드포인트에 대한 가상 IP(VIP)를 정의하고 각 VIP의 서브넷을 식별합니다 .
6. 오버클라우드 네트워크 및 오버클라우드 VIP를 프로비저닝합니다.
7. 오버클라우드에 베어 메탈 노드를 등록합니다.

참고

사전 프로비저닝된 베어 메탈 노드를 사용하는 경우 단계 7, 8 및 9를 건너뜁니다.

8. 오버클라우드의 베어 메탈 노드를 검사합니다.
9. 베어 메탈 노드를 프로비저닝 합니다.
10. 이전 단계에서 설정한 구성을 사용하여 오버클라우드를 배포합니다 .

4.1. 리프 네트워크 정의

RHOSP(Red Hat OpenStack Platform) director는 사용자가 구성하는 YAML 형식의 사용자 지정 네트워크 정의 파일에서 오버클라우드 리프 네트워크를 생성합니다. 이 사용자 지정 네트워크 정의 파일은 각 구성 가능 네트워크 및 해당 속성을 나열하고 각 리프에 필요한 서브넷도 정의합니다.

오버클라우드의 스파인-리프트 네트워크의 사양이 포함된 YAML 형식의 사용자 지정 네트워크 정의 파일을 생성하려면 다음 단계를 완료합니다. 나중에 프로비저닝 프로세스에서 RHOSP 오버클라우드를 배포할 때 포함하는 네트워크 정의 파일에서 heat 환경 파일을 생성합니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **templates** 디렉토리를 **/home/stack**:에 생성

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

4. 기본 네트워크 정의 템플릿 **routed-networks.yaml** 을 사용자 지정 **템플릿** 디렉터리에 복사합니다.

예제

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

5. 네트워크 정의 템플릿의 사본을 편집하여 각 기본 네트워크와 관련 리프 서브넷을 구성 가능 네트워크 항목으로 정의합니다.

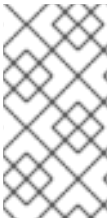
작은 정보

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리의* 네트워크 정의 파일 구성 옵션을 참조하십시오.

예제

다음 예제에서는 내부 API 네트워크 및 리프 네트워크를 정의하는 방법을 보여줍니다.

```
- 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
```



참고

언더클라우드에서 이미 이러한 네트워크를 생성했기 때문에 사용자 정의 네트워크 정의 템플릿에 컨트롤 플레인 네트워크를 정의하지 않습니다. 그러나 오버클라우드에서 NIC를 적절하게 구성할 수 있도록 매개변수를 수동으로 설정해야 합니다. 자세한 내용은 언더클라우드의 라우팅된 스파인-리프트 구성을 참조하십시오.

참고

RHOSP는 네트워크 서브넷 및 **allocation_pools** 값에 대한 자동 검증을 수행하지 않습니다. 이러한 값을 일관되게 정의하고 기존 네트워크와 충돌하지 않도록 합니다.

참고

vip 매개변수를 추가하고 컨트롤러 기반 서비스를 호스팅하는 네트워크의 값을 **true** 로 설정합니다. 이 예에서 **InternalApi** 네트워크에는 이러한 서비스가 포함되어 있습니다.

다음 단계

1. 생성한 사용자 정의 네트워크 정의 파일의 경로와 파일 이름을 기록해 둡니다. 나중에 RHOSP 오버클라우드용 네트워크를 프로비저닝할 때 이 정보가 필요합니다.
2. 다음 단계로 이동하여 리프 역할 정의 및 네트워크 연결 .

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리의* 네트워크 정의 파일 구성 옵션

4.2. 리프 역할 정의 및 네트워크 연결

RHOSP(Red Hat OpenStack Platform) director는 각 리프에 대해 구성 가능한 역할을 생성하고 사용자가 구성하는 역할 템플릿에서 구성 가능 네트워크를 각 역할에 연결합니다. 먼저 director 코어 템플릿에서 기본 Controller, Compute, Ceph Storage 역할을 복사하고 환경 요구 사항을 충족하도록 수정합니다. 개별 역할을 모두 생성한 후 **openstack overcloud roles generate** 명령을 실행하여 하나의 대규모 사용자 지정 역할 데이터 파일에 연결합니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. RHOSP와 함께 제공되는 Controller, Compute 및 Ceph Storage 역할의 기본 역할을 **stack** 사용자의 홈 디렉터리에 복사합니다. 파일의 이름을 변경하여 리프 0임을 나타냅니다.

```
$ 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. 리프 0 파일을 복사하여 리프 1 및 리프 2 파일을 만듭니다.

```
$ 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. 각 파일에서 매개 변수를 편집하여 해당 리프 매개 변수와 일치합니다.

작은 정보

역할 데이터 템플릿의 다양한 매개변수에 대한 자세한 내용은 *Red Hat OpenStack Platform 배포 사용자 정의 가이드*의 역할 매개 변수 검사를 참조하십시오.

예 - ComputeLeaf0

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

예 - CephStorageLeaf0

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

6. 각 리프 **네트워크** 매개변수와 일치하도록 리프 1 및 리프 2 파일에서 network 매개변수를 편집합니다.

예 - ComputeLeaf1

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

예 - CephStorageLeaf1

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

참고

이는 리프 1과 리프 2에만 적용됩니다. leaf 0의 **network** 매개변수는 **_subnet** 접미사와 결합된 각 서브넷의 소문자 이름인 기본 서브넷 값을 유지합니다. 예를 들어 리프 0의 내부 API는 **internal_api_subnet** 입니다.

7. 각 Controller, Compute 및 (있는 경우) Networker 역할 파일에서 **ServicesDefault** 매개변수 아래의 서비스 목록에 OVN BGP 에이전트를 추가합니다.

예제

```
- name: ControllerRack1
  ...
  ServicesDefault:
    ...
    - OS::TripleO::Services::Frr
    - OS::TripleO::Services::OVNBgpAgent
    ...
```

8. 역할 구성이 완료되면 **오버클라우드 roles generate** 명령을 실행하여 전체 역할 데이터 파일을 생성합니다.

예제

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

이렇게 하면 각 리프 네트워크에 대한 모든 사용자 지정 역할을 포함하는 하나의 사용자 지정 역할 데이터 파일이 생성됩니다.

다음 단계

1. **오버클라우드 roles generate 명령으로 생성된 사용자 지정 역할 데이터 파일의 경로와 파일 이름을 기록해** 둡니다. 오버클라우드를 배포할 때 나중에 이 경로를 사용합니다.
2. 리프 역할에 대한 사용자 정의 NIC 구성 생성 다음 단계로 이동합니다.

추가 리소스

- *Red Hat OpenStack Platform 배포 가이드 사용자* 정의에서 역할 매개변수 검사

4.3. 리프 역할에 대한 사용자 정의 NIC 구성 생성

RHOSP(Red Hat OpenStack Platform) director에서 생성하는 각 역할에는 고유한 NIC 설정이 필요합니다. 사용자 지정 NIC 템플릿 세트와 사용자 지정 템플릿을 해당 역할에 매핑하는 사용자 지정 환경 파일을 생성하려면 다음 단계를 완료합니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.
- 사용자 지정 네트워크 정의 파일이 있습니다.

- 사용자 지정 역할 데이터 파일이 있습니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 기본 NIC 템플릿 중 하나에서 콘텐츠를 복사하여 NIC 구성에 대한 사용자 지정 템플릿을 생성합니다.

예제

이 예에서는 NIC 구성에 대한 사용자 지정 템플릿에 사용할 **single-nic-vlans** NIC 템플릿이 복사됩니다.

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

4. 이전 단계에서 생성한 각 NIC 템플릿에서 spine-leaf 토폴로지의 세부 사항과 일치하도록 NIC 구성을 변경합니다.

예제

```
{% 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 %}
```

작은 정보

*자세한 내용은 director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 사용자 지정 네트워크 인터페이스 템플릿 정의*를 참조하십시오.

5. 사용자 지정 NIC 템플릿을 각 사용자 지정 역할에 매핑하는 **parameter_defaults** 섹션이 포함된 **spine-leaf-nic-roles-map.yaml** 과 같은 사용자 지정 환경 파일을 생성합니다.

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

예제

```
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'
```

다음 단계

1. 사용자 정의 NIC 템플릿의 경로 및 파일 이름과 사용자 정의 NIC 템플릿을 각 사용자 지정 역할에 매핑하는 사용자 정의 환경 파일을 확인합니다. 오버클라우드를 배포할 때 나중에 이 경로를 사용합니다.

2. 리프 네트워크 구성을 위한 다음 단계로 이동합니다 .

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 사용자 지정 네트워크 인터페이스 템플릿 정의

4.4. 리프 네트워크 구성

스파인 리프 아키텍처에서 각 리프는 해당 리프의 특정 브릿지 또는 VLAN을 통해 트래픽을 라우팅합니다. 이러한 트래픽은 종종 엣지 컴퓨팅 시나리오의 경우입니다. 따라서 RHOSP(Red Hat OpenStack Platform) 컨트롤러 및 컴퓨팅 네트워크 구성에서 **br-ex** 브리지를 사용하는 기본 매핑을 변경해야 합니다.

RHOSP director는 언더클라우드를 생성하는 동안 컨트롤 플레인 네트워크를 생성합니다. 그러나 오버클라우드는 각 리프의 컨트롤 플레인에 액세스해야 합니다. 이 액세스를 활성화하려면 배포에 추가 매개변수를 정의해야 합니다.

별도의 네트워크 매핑이 포함된 사용자 정의 네트워크 환경 파일을 생성하고 오버클라우드의 컨트롤 플레인 네트워크에 대한 액세스를 설정하려면 다음 단계를 완료합니다.

사전 요구 사항

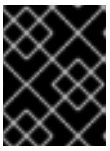
- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **spine-leaf-ctlplane.yaml** 과 같은 새 사용자 지정 환경 파일에서 **parameter_defaults** 섹션을 생성하고 기본 **br-ex** 브리지를 사용하는 각 리프에 대해 **NeutronBridgeMappings** 매개변수를 설정합니다.



중요

네트워크 정의를 포함하도록 생성하는 사용자 지정 환경 파일의 이름은 **.yaml** 또는 **.template** 으로 끝나야 합니다.

- 플랫 네트워크 매핑의 경우 **NeutronFlatNetworks** 매개변수의 각 리프를 나열하고 각 리프에 대해 **NeutronBridgeMappings** 매개변수를 설정합니다.

예제

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

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

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

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

작은 정보

자세한 내용은 Chapter 17. *Overcloud 매개변수 가이드의 네트워킹(neutron) 매개변수*

- VLAN 네트워크 매핑의 경우 **NeutronNetworkType** 에 **vlan** 을 추가하고 **NeutronNetworkVLANRanges** 를 사용하여 리프 네트워크에 VLAN을 매핑합니다.

  예제

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

  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"

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

참고

스파인-리프 토폴로지에서 플랫 네트워크와 VLAN을 모두 사용할 수 있습니다.

4. < **role>ControlPlaneSubnet** 매개변수를 사용하여 각 spine-leaf 네트워크에 대한 컨트롤 플레인 서브넷 매핑을 추가합니다.

   예제

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    ControllerControlPlaneSubnet: leaf0
  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Controller1ControlPlaneSubnet: leaf0
```

```
Controller2Parameters:
  NeutronBridgeMappings: "leaf0:br-ex"
  Controller2ControlPlaneSubnet: leaf0
Compute0Parameters:
  NeutronBridgeMappings: "leaf0:br-ex"
  Compute0ControlPlaneSubnet: leaf0
CephStorage0Parameters:
  CephStorage0ControlPlaneSubnet: leaf0
Compute1Parameters:
  NeutronBridgeMappings: "leaf1:br-ex"
  Compute1ControlPlaneSubnet: leaf1
CephStorage1Parameters:
  CephStorage1ControlPlaneSubnet: leaf1
Compute2Parameters:
  NeutronBridgeMappings: "leaf2:br-ex"
  Compute2ControlPlaneSubnet: leaf2
CephStorage2Parameters:
  CephStorage2ControlPlaneSubnet: leaf2
```

다음 단계

1. 생성한 사용자 지정 네트워크 환경 파일의 경로와 파일 이름을 기록해 둡니다. 오버클라우드를 배포할 때 나중에 이 정보가 필요합니다.
2. 가상 IP 주소의 서브넷 설정 다음 단계로 이동합니다.

추가 리소스

- *17장. Overcloud 매개변수 가이드의 네트워킹(neutron) 매개변수*

4.5. 가상 IP 주소의 서브넷 설정

기본적으로 RHOSP(Red Hat Openstack Platform) 컨트롤러 역할은 각 네트워크의 가상 IP(VIP) 주소를 호스팅합니다. RHOSP 오버클라우드는 컨트롤 플레인을 제외하고 각 네트워크의 기본 서브넷에서 VIP를 가져옵니다. 컨트롤 플레인은 표준 언더클라우드 설치 중에 생성된 기본 서브넷 이름인 **ctlplane-subnet** 을 사용합니다.

이 문서에서 사용되는 스파인-리프 예에서 기본 기본 프로비저닝 네트워크는 **ctlplane-subnet** 대신 **leaf0** 입니다. 즉, 서브넷을 **leaf0** 에 매핑하려면 **network:ctlplane** 매개변수에 값 쌍 **subnet: leaf0** 을 추가해야 합니다.

오버클라우드의 VIP에 대한 덮어쓰기가 포함된 YAML 형식의 사용자 지정 네트워크 VIP 정의 파일을 생성하려면 다음 단계를 완료합니다. 나중에 프로비저닝 프로세스에서 RHOSP 오버클라우드를 배포할 때 포함하는 네트워크 VIP 정의 파일에서 heat 환경 파일을 생성합니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **spine-leaf-vip-data.yaml** 과 같은 새 사용자 지정 네트워크 VIP 정의 템플릿에서 컨트롤러 노드에서 사용하는 특정 서브넷에 생성해야 하는 가상 IP 주소를 나열합니다.

예제

```
- 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
```

spine-leaf-vip-data.yaml 파일에서 다음 매개변수를 사용할 수 있습니다.

network

neutron 네트워크 이름을 설정합니다. 이는 유일한 필수 매개변수입니다.

ip_address

VIP의 IP 주소를 설정합니다.

서브넷

neutron 서브넷 이름을 설정합니다. 가상 IP neutron 포트를 생성할 때 서브넷을 지정하려면 를 사용합니다. 이 매개변수는 배포에서 라우팅된 네트워크를 사용하는 경우 필요합니다.

dns_name

FQDN(정규화된 도메인 이름)을 설정합니다.

name

가상 IP 이름을 설정합니다.

작은 정보

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 구성 가능 네트워크 추가* 를 참조하십시오.

다음 단계

1. 생성한 사용자 정의 네트워크 VIP 정의 템플릿의 경로와 파일 이름을 기록해 둡니다. RHOSP 오버클라우드의 네트워크 VIP를 프로비저닝할 때 나중에 이 경로를 사용합니다.
2. 오버클라우드의 프로비저닝 네트워크 및 VIP 단계로 이동합니다.

4.6. 오버클라우드의 네트워크 및 VIP 프로비저닝

RHOSP(Red Hat OpenStack Platform) 프로비저닝 프로세스는 네트워크 정의 파일을 사용하여 네트워크 사양이 포함된 새 heat 환경 파일을 생성합니다. 배포에서 VIP를 사용하는 경우 RHOSP는 VIP 정의 파일에서 새 heat 환경 파일을 생성합니다. 네트워크와 VIP를 프로비저닝한 후 나중에 오버클라우드를 배포하는 데 사용하는 두 개의 heat 환경 파일이 있습니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.
- 네트워크 구성 템플릿이 있습니다.
- VIP를 사용하는 경우 VIP 정의 템플릿이 있습니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 오버클라우드 네트워크를 프로비저닝합니다.
오버클라우드 network provision 명령을 사용하여 이전에 생성한 네트워크 정의 파일의 경로를 제공합니다.

작은 정보

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드 네트워크 정의 구성 및 프로비저닝 을 참조하십시오.

예제

이 예에서 경로는 **/home/stack/templates/spine-leaf-networks-data.yaml** 입니다. **--output** 인수를 사용하여 명령으로 생성된 파일의 이름을 지정합니다.

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

중요

지정하는 출력 파일의 이름은 **.yaml** 또는 **.template** 으로 끝나야 합니다.

4. 오버클라우드 VIP를 프로비저닝합니다.
이전에 생성한 VIP 정의 파일의 이름을 지정하려면 **--stack** 인수와 함께 **오버클라우드 네트워크 vip provision** 명령을 사용합니다. **--output** 인수를 사용하여 명령으로 생성된 파일의 이름을 지정합니다.

작은 정보

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 오버클라우드의 네트워크 VIP 구성 및 프로비저닝*을 참조하십시오.

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

중요

지정하는 출력 파일의 이름은 **.yaml** 또는 **.template** 으로 끝나야 합니다.

5. 생성된 출력 파일의 경로와 파일 이름을 기록해 둡니다. 오버클라우드를 배포할 때 나중에 이 정보를 사용합니다.

검증

- 다음 명령을 사용하여 명령이 오버클라우드 네트워크 및 서브넷을 생성했는지 확인할 수 있습니다.

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

<network>, <subnet> 및 <port>를 확인할 네트워크, 서브넷 및 포트의 이름 또는 UUID로 바꿉니다.

다음 단계

1. 사전 프로비저닝된 노드를 사용하는 경우 오버클라우드 배포 명령 실행으로 건너 뜁니다.
2. 그렇지 않으면 오버클라우드의 베어 메탈 노드 등록 단계를 진행합니다.

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드 네트워크 정의 구성 및 프로비저닝
- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 네트워크 VIP 구성 및 프로비저닝
- *명령줄 인터페이스 참조*의 오버클라우드 네트워크 프로비저닝
- *명령줄 인터페이스 참조*에서 오버클라우드 네트워크 vip 프로비저닝

4.7. 오버클라우드에 베어 메탈 노드 등록

RHOSP(Red Hat OpenStack Platform) director에는 물리적 시스템의 하드웨어 및 전원 관리 세부 정보를 지정하는 사용자 정의 노드 정의 템플릿이 필요합니다. JSON 또는 YAML 형식으로 이 템플릿을 생성할

수 있습니다. 물리적 머신을 베어 메탈 노드로 등록한 후 인트로스펙션을 수행한 다음, 마지막으로 프로비저닝합니다.

참고

사전 프로비저닝된 베어 메탈 노드를 사용하는 경우 베어 메탈 노드 등록, 인트로스펙션 및 프로비저닝을 건너뛰고 스파인 -리프가 활성화된 오버클라우드 배포로 이동할 수 있습니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **baremetal-nodes.yaml** 과 같은 새 노드 정의 템플릿을 생성합니다. 하드웨어 및 전원 관리 세부 정보가 포함된 물리적 시스템 목록을 추가합니다.

예제

```
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"
```

작은 정보

템플릿 매개변수 값 및 JSON 예에 대한 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 노드 등록을 참조하십시오.

4. 템플릿 형식 및 구문을 확인합니다.

예제

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

5. 오류를 수정하고 노드 정의 템플릿을 저장합니다.

6. 노드 정의 템플릿을 RHOSP director로 가져와 템플릿의 각 노드를 director에 등록합니다.

예제

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

검증

- 노드 등록 및 설정이 완료되면 director가 노드를 성공적으로 등록했는지 확인합니다.

```
$ openstack baremetal node list
```

baremetal node list 명령에는 가져온 노드가 포함되어야 하며 상태를 **관리할** 수 있어야 합니다.

다음 단계

- 다음 단계로 이동하여 오버클라우드의 베어 메탈 노드 인트로스펙션 .

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 노드 등록.
- *명령줄 인터페이스 참조*에서 오버클라우드 노드 가져오기

4.8. 오버클라우드에서 베어 메탈 노드 인트로스펙션

물리적 머신을 베어 메탈 노드로 등록한 후 OpenStack Platform(RHOSP) director 인트로스펙션을 사용하여 노드의 하드웨어 세부 정보를 자동으로 추가하고 각 이더넷 MAC 주소에 대한 포트를 생성할 수 있습니다. 베어 메탈 노드에서 인트로스펙션을 수행한 후 최종 단계는 이를 프로비저닝하는 것입니다.

참고

사전 프로비저닝된 베어 메탈 노드를 사용하는 경우 베어 메탈 노드를 인트로스펙션하고 인트로스펙션을 건너뛰고 스파인-리프가 활성화된 오버클라우드 배포로 이동할 수 있습니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.
- RHOSP를 사용하여 오버클라우드의 베어 메탈 노드를 등록했습니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. pre-introspection 검증 그룹을 실행하여 인트로스펙션 요구 사항을 확인합니다.

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

4. 검증 보고서 결과를 확인하십시오.
5. (선택 사항) 특정 검증의 자세한 출력을 검토합니다.

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

<UUID>를 검토할 보고서의 특정 검증 UUID로 바꿉니다.

중요

FAILED 검증으로 인해 RHOSP를 배포하거나 실행하지 못합니다. 그러나 **FAILED** 검증 결과는 프로덕션 환경에서 잠재적으로 문제가 발행할 수 있다는 것을 의미합니다.

6. 모든 노드의 하드웨어 속성을 검사합니다.

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

작은 정보

자세한 내용은 director 가이드를 사용하여 *Red Hat OpenStack Platform 설치 및 관리에서 director 인트로스펙션을 사용하여* 베어 메탈 노드 하드웨어 정보를 수집합니다 .

별도의 터미널 창에서 인트로스펙션 진행 상태 로그를 모니터링합니다.

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

검증

- 인트로스펙션이 완료되면 모든 노드가 available 상태로 변경됩니다.

다음 단계

- 다음 단계로 이동하여 오버클라우드의 베어 메탈 노드를 프로비저닝합니다.

추가 리소스

- director 인트로스펙션을 사용하여 *director 가이드로 Red Hat OpenStack Platform 설치 및 관리*에서 베어 메탈 노드 하드웨어 정보 수집
- *명령줄 인터페이스 참조*에서 오버클라우드 노드 인트로스펙션

4.9. 오버클라우드의 베어 메탈 노드 프로비저닝

RHOSP(Red Hat OpenStack Platform)의 베어 메탈 노드를 프로비저닝하려면 이러한 노드에 오버클라우드 역할을 배포하고 할당하려는 베어 메탈 노드의 수량 및 속성을 정의합니다. 노드의 네트워크 레이아웃도 정의합니다. 노드 정의 파일에 이 모든 정보를 YAML 형식으로 추가합니다.

프로비저닝 프로세스에서 노드 정의 파일에서 heat 환경 파일을 생성합니다. 이 heat 환경 파일에는 노드 수, 예측 노드 배치, 사용자 정의 이미지, 사용자 정의 NIC를 포함하여 노드 정의 파일에 구성한 노드 사양이 포함되어 있습니다. 오버클라우드를 배포할 때 이 heat 환경 파일을 배포 명령에 포함합니다. 프로비저닝 프로세스는 노드 정의 파일에서 각 노드 또는 역할에 대해 정의된 모든 네트워크의 포트 리소스도 프로비저닝합니다.

참고

사전 프로비저닝된 베어 메탈 노드를 사용하는 경우 베어 메탈 노드 프로비저닝을 건너뛰고 스파인-리프가 활성화된 오버클라우드 배포로 이동할 수 있습니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.
- 베어 메탈 노드는 등록되어 있으며 프로비저닝 및 배포에 사용할 수 있습니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. **spine-leaf-baremetal-nodes.yaml** 과 같은 베어 메탈 노드 정의 파일을 생성하고 프로비저닝하려는 각 역할의 노드 수를 정의합니다.

예제

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

```
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
      default_route_network:
      - external
- name: Compute0
  count: 1
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
      subnet: tenant_subnet02
    - network: storage
      subnet: storage_subnet02
    network_config:
      template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
- name: Compute1
...
```

작은 정보

베어 메탈 노드 정의 파일을 설정할 수 있는 속성에 대한 자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드의 베어 메탈 노드 프로비저닝 을 참조하십시오.

4. 오버클라우드 노드 프로비저닝 명령을 사용하여 **오버클라우드 베어 메탈 노드를 프로비저닝합니다**.

예제

```
$ 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
```

중요

지정하는 출력 파일의 이름은 **.yaml** 또는 **.template** 으로 끝나야 합니다.

5. 별도의 터미널에서 프로비저닝 진행 상황을 모니터링합니다. 프로비저닝이 성공하면 노드 상태가 **available** 에서 **active** 로 변경됩니다.

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

6. **metalsmith** 툴을 사용하여 할당 및 포트를 포함하여 노드의 통합 보기를 가져옵니다.

```
$ metalsmith list
```

7. 생성된 출력 파일의 경로와 파일 이름을 기록해 둡니다. 오버클라우드를 배포할 때 나중에 이 경로가 필요합니다.

검증

- 호스트 이름과 노드 연결을 확인합니다.

```
$ openstack baremetal allocation list
```

다음 단계

- 다음 단계로 이동하여 스파인 -리프가 활성화된 오버클라우드 배포.

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서* 오버클라우드용 베어 메탈 노드 프로비저닝

4.10. 스파인-리프트가 활성화된 오버클라우드 배포

RHOSP(Red Hat OpenStack Platform) 오버클라우드 배포의 마지막 단계는 **overcloud deploy** 명령을 실행하는 것입니다. 명령에 대한 입력에는 사용자가 구성한 모든 다양한 오버클라우드 템플릿 및 환경 파일이 포함됩니다. RHOSP director는 이러한 템플릿과 파일을 오버클라우드 설치 및 설정 방법에 대한 계획으로 사용합니다.

사전 요구 사항

- 언더클라우드 호스트 및 **stack** 사용자의 인증 정보에 액세스합니다.
- 이 섹션의 이전 절차에 나열된 모든 단계를 수행하고 **overcloud deploy** 명령에 입력으로 사용할 다양한 heat 템플릿 및 환경 파일을 모두 어셈블했습니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.
2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 오버클라우드 환경에 필요한 사용자 지정 환경 파일 및 사용자 지정 템플릿을 수집합니다. 이 목록에는 director 설치와 함께 제공된 편집되지 않은 heat 템플릿 파일과 사용자가 생성한 사용자 지정 파일이 포함되어 있습니다. 다음 파일에 대한 경로가 있는지 확인합니다.
    - 오버클라우드의 스파인-리프트 네트워크에 대한 사양이 포함된 사용자 정의 네트워크 정의

파일(예: **spine-leaf-networks-data.yaml**).
자세한 내용은 리프 네트워크 정의를 참조하십시오.

- 각 리프에 대한 역할을 정의하는 사용자 지정 역할 데이터 파일입니다.
예: **spine-leaf-roles.yaml**.

  자세한 내용은 리프 역할 정의 및 네트워크 연결을 참조하십시오.

- 각 역할의 역할 및 사용자 지정 NIC 템플릿 매핑이 포함된 사용자 지정 환경 파일입니다.
예: **spine-leaf-nic-roles-map.yaml**.

  자세한 내용은 리프 역할에 대한 사용자 정의 NIC 구성 생성을 참조하십시오.

- 별도의 네트워크 매핑이 포함된 사용자 정의 네트워크 환경 파일은 오버클라우드의 컨트롤 플레인 네트워크에 대한 액세스를 설정합니다.
예: **spine-leaf-ctlplane.yaml**

  자세한 내용은 리프 네트워크 구성을 참조하십시오.

- 오버클라우드 네트워크를 프로비저닝한 출력 파일입니다.
예: **spine-leaf-networks-provisioned.yaml**

  자세한 내용은 오버클라우드의 프로비저닝 네트워크 및 VIP를 참조하십시오.

- 오버클라우드 VIP 프로비저닝의 출력 파일입니다.
예: **spine-leaf-vips-provisioned.yaml**

  자세한 내용은 오버클라우드의 프로비저닝 네트워크 및 VIP를 참조하십시오.

- 사전 프로비저닝된 노드를 사용하지 않는 경우 베어 메탈 노드의 프로비저닝의 출력 파일입니다.
예: **spine-leaf-baremetal-nodes-provisioned.yaml**.

  자세한 내용은 오버클라우드의 베어 메탈 노드 프로비저닝을 참조하십시오 .

- 기타 사용자 지정 환경 파일

4. 사용자 지정 환경 파일 및 명령에 입력된 사용자 지정 템플릿을 신중하게 정렬하여 **overcloud deploy** 명령을 입력합니다.
일반 규칙은 편집되지 않은 heat 템플릿 파일을 먼저 지정하고 사용자 지정 환경 파일 및 기본 속성 덮어쓰기와 같은 사용자 지정 구성이 포함된 사용자 지정 템플릿을 지정하는 것입니다.

   오버클라우드 배포 명령에 대한 입력을 나열하려면 다음 순서를 따르십시오.

   a. 각 역할에 매핑된 사용자 정의 NIC 템플릿이 포함된 사용자 지정 환경 파일을 포함합니다.
   예: **network-environment.yaml** 이후 **spine-leaf-nic-roles-map.yaml**.

      network-environment.yaml 파일은 매핑 파일에서 재정의하는 구성 가능 네트워크 매개변수에 대한 기본 네트워크 구성을 제공합니다. director는 **network-environment.j2.yaml** Jinja2 템플릿에서 이 파일을 렌더링합니다.

   b. 다른 스파인 리프 네트워크 환경 파일을 생성한 경우 roles-NIC 템플릿 매핑 파일 뒤에 이러한 환경 파일을 포함합니다.

   c. 추가 환경 파일을 추가합니다. 예를 들어 컨테이너 이미지 위치 또는 Ceph 클러스터 구성이 포함된 환경 파일입니다.

예제

샘플 **overcloud deploy** 명령에서 발췌한 내용은 명령 입력의 적절한 순서를 보여줍니다.

```
$ openstack overcloud deploy --templates \
  -n /home/stack/templates/spine-leaf-networks-data.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml
\
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/frr.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ovn-bgp-
agent.yaml \
  -e /home/stack/templates/spine-leaf-nic-roles-map.yaml \
  -e /home/stack/templates/spine-leaf-ctlplane.yaml \
  -e /home/stack/templates/spine-leaf-baremetal-provisioned.yaml \
  -e /home/stack/templates/spine-leaf-networks-provisioned.yaml \
  -e /home/stack/templates/spine-leaf-vips-provisioned.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /home/stack/inject-trust-anchor-hiera.yaml \
  -r /home/stack/templates/spine-leaf-roles-data.yaml
  ...
```

작은 정보

자세한 내용은 *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 오버클라우드 생성* 을 참조하십시오.

5. **overcloud deploy** 명령을 실행합니다.
   오버클라우드 생성이 완료되면 RHOSP director에서 오버클라우드 액세스에 도움이 되는 세부 정보를 제공합니다.

검증

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 오버클라우드 배포 검증* 단계를 수행합니다.

추가 리소스

- *director 가이드를 사용하여 Red Hat OpenStack Platform 설치 및 관리에서 오버클라우드 생성*
- *명령줄 인터페이스 참조에* overcloud deploy

4.11. 스파인-리프 배포에 새 리프 추가

네트워크 용량을 늘리거나 새 물리적 사이트를 추가할 때 RHOSP(Red Hat OpenStack Platform) 스파인-리프 네트워크에 새 리프를 추가해야 할 수 있습니다.

사전 요구 사항

- RHOSP 배포에서는 스파인-리프트 네트워크 토폴로지를 사용합니다.

프로세스

1. 언더클라우드 호스트에 **stack** 사용자로 로그인합니다.

2. **stackrc** 언더클라우드 인증 정보 파일을 소싱합니다.

```
$ source ~/stackrc
```

3. 네트워크 정의 템플릿을 엽니다(예: **/home/stack/templates/spine-leaf-networks-data.yaml**). 적절한 기본 네트워크에서 리프 서브넷을 추가할 새 리프에 대한 구성 가능 네트워크 항목으로 추가합니다.

예제

이 예에서는 새 리프(**leaf**3)의 subnet 항목이 추가되었습니다.

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

4. 추가할 새 리프에 대한 역할 데이터 파일을 만듭니다.

   a. 추가하려는 새 리프에 대한 리프 Compute 및 리프 Ceph Storage 파일을 복사합니다.

   예제

   이 예에서 **Compute1.yaml** 및 **CephStorage1.yaml** 은 각각 새 리프, **Compute3.yaml** 및 **CephStorage3.yaml** 에 대해 복사됩니다.

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

   b. 새 리프 파일에서 **name** 및 **HostnameFormatDefault** 매개변수를 편집하여 해당 리프 매개변수와 일치하도록 합니다.

   예제

   예를 들어 Leaf 1 Compute 파일의 매개변수에는 다음과 같은 값이 있습니다.

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

예제

Leaf 1 Ceph Storage 매개변수에는 다음과 같은 값이 있습니다.

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

c. 해당 Leaf 네트워크 매개변수와 일치하도록 새 리프 파일에서 network 매개변수를 편집합니다.

예제

예를 들어 Leaf 1 Compute 파일의 매개변수에는 다음과 같은 값이 있습니다.

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

예제

Leaf 1 Ceph Storage 매개변수에는 다음과 같은 값이 있습니다.

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

d. 역할 구성이 완료되면 다음 명령을 실행하여 전체 역할 데이터 파일을 생성합니다. 네트워크에 있는 모든 리프와 추가 중인 새 리프를 포함합니다.

예제

이 예에서는 leaf3이 leaf0, leaf1 및 leaf2에 추가됩니다.

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

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

5. 추가할 리프에 대한 사용자 정의 NIC 구성을 생성합니다.

a. 추가하려는 새 리프에 대한 리프 Compute 및 리프 Ceph Storage NIC 구성 파일을 복사합니다.

예제

이 예에서 **computeleaf1.yaml** 및 **ceph-storageleaf1.yaml** 은 각각 새 leaf, **computeleaf3.yaml** 및 **ceph-storageleaf3.yaml** 에 대해 복사됩니다.

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

6. 각 역할의 역할 및 사용자 지정 NIC 템플릿 매핑이 포함된 사용자 정의 환경 파일을 엽니다(예: spine-leaf-nic-roles-map.yaml). 추가할 새 리프의 각 역할에 대한 항목을 삽입합니다.

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

예제

이 예에서는 **ComputeLeaf3NetworkConfigTemplate** 및 **CephStorage3NetworkConfigTemplate** 항목이 추가되었습니다.

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

7. 별도의 네트워크 매핑이 포함된 사용자 정의 네트워크 환경 파일을 열고 오버클라우드의 컨트롤 플레인 네트워크에 대한 액세스를 설정합니다(예: **spine-leaf-ctlplane.yaml**). **parameter_defaults** 섹션에서 새 리프 네트워크의 컨트롤 플레인 서브넷 매핑을 추가합니다. 또한 새 리프 네트워크의 외부 네트워크 매핑을 포함합니다.

- 플랫 네트워크 매핑의 경우 **NeutronFlatNetworks** 매개변수에 새 **리프(leaf**3)를 나열하고 새 리프에 대한 **NeutronBridgeMappings** 매개변수를 설정합니다.

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2,leaf3
```

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

- VLAN 네트워크 매핑의 경우 새 리프(**leaf**3) 네트워크의 VLAN을 매핑하도록 **NeutronNetworkVLANRanges** 를 추가로 설정합니다.

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

예제

이 예에서는 플랫 네트워크 매핑이 사용되며 새 리프(**leaf3**) 항목이 추가됩니다.

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

8. 수정된 네트워크를 프로비저닝합니다.
   자세한 내용은 오버클라우드 네트워크 및 오버클라우드 VIP 프로비저닝을 참조하십시오.

9. 이전에 생성한 베어 메탈 노드 정의 파일(예: **spine-leaf-baremetal-nodes.yaml**)에서 **network_config_update** 변수가 **true** 로 설정되어 있는지 확인합니다.

예제

```
- name: Controller
  count: 3
```

```
defaults:
  networks:
  - network: ctlplane
    vif: true
  - network: external
    subnet: external_subnet
  - network: internal_api
    subnet: internal_api_subnet01
  - network: storage
    subnet: storage_subnet01
  - network: storage_mgmt
    subnet: storage_mgmt_subnet01
  - network: tenant
    subnet: tenant_subnet01
  network_config:
    template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
    default_route_network:
    - external
  network_config_update: true
```

10. 수정된 노드를 프로비저닝합니다.
    자세한 내용은 베어 메탈 노드 프로비저닝 을 참조하십시오.

11. 스파인-리프가 활성화된 오버클라우드 배포 단계에 따라 스파인-리프가 활성화된 오버클라우드를 재배포합니다.