Red Hat OpenStack Platform 15

スパイン/リーフ型ネットワーク

Red Hat OpenStack Platform director を使用したルーティング対応のスパイン/リーフ型ネットワークの設定

Last Updated: 2022-09-15

Red Hat OpenStack Platform 15 スパイン/リーフ型ネットワーク

Red Hat OpenStack Platform director を使用したルーティング対応のスパイン/リーフ型ネットワークの設定

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本ガイドでは、オーバークラウドにおけるルーティング対応のスパイン/リーフ型ネットワークの設定方法について説明します。これには、アンダークラウドの設定、主要な設定ファイルの記述、ノード用のロールの作成が含まれます。

目次

第1章 はじめに

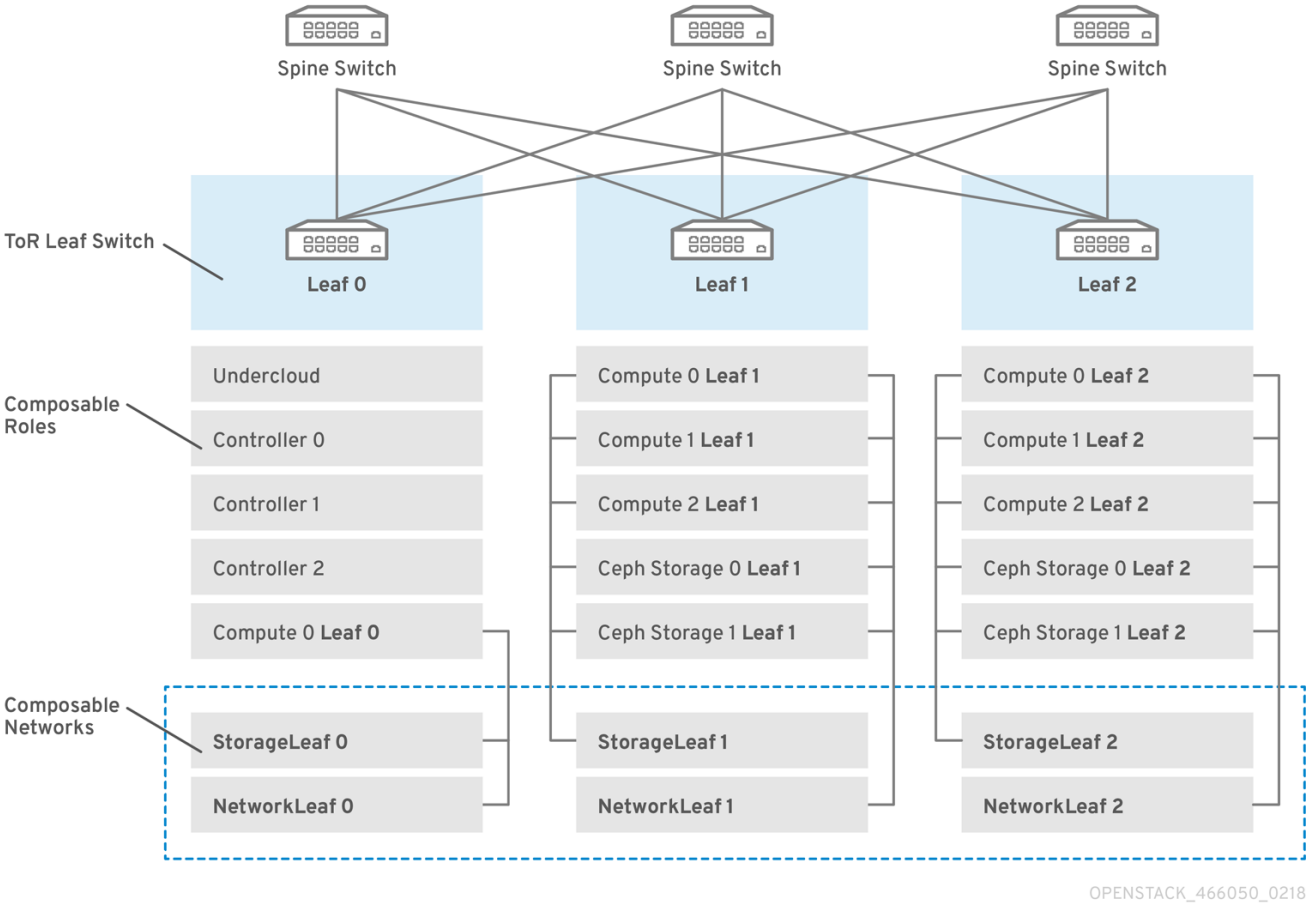
本ガイドでは、Red Hat OpenStack Platform 環境に向けたスパイン/リーフ型ネットワークトポロジーの構築について説明します。これには、完全なエンドツーエンドのシナリオと、お使いの環境でより広範囲なネットワークトポロジーを複製するのに役立つサンプルファイルが含まれます。

1.1. スパイン/リーフ型ネットワーク

Red Hat OpenStack Platform のコンポーザブルネットワークアーキテクチャーを使用して、ネットワークをルーティング対応のスパイン/リーフ型データセントートポロジーに適合させることができます。ルーティング対応のスパイン/リーフの実際の適用では、リーフは Compute または Storage のコンポーザブルロールに相当し、図1.1「ルーティング対応のスパイン/リーフトポロジーの例」に示したように、通常はデータセンターのラック内にあります。**Leaf 0** ラックには、アンダークラウドノード、コントローラーノード、およびコンピュートノードがあります。コンポーザブルネットワークはコンポーザブルロールに割り当てられたノードに提示されます。次の図は、以下の構成を示しています。

- **StorageLeaf** ネットワークは、Ceph Storage とコンピュートノードに提示されます。
- **NetworkLeaf** は、構成する任意のネットワークの例を示します。

図1.1 ルーティング対応のスパイン/リーフトポロジーの例



1.2. スパイン/リーフ型ネットワークトポロジー

スパイン/リーフ型ネットワークのシナリオでは、OpenStack Networking (neutron) の機能を利用して、1 つのネットワークのセグメント内に複数のサブネットが定義されます。それぞれのネットワークは、Leaf 0 として動作するベースネットワークを使用します。director は、メインのネットワークのセ

グメントとして Leaf 1 および Leaf 2 サブネットを作成します。

このシナリオでは、以下のネットワークを使用します。

表1.1 Leaf 0 ネットワーク (ベースネットワーク)

ネットワーク	アタッチされているロール	サブネット
プロビジョニング / コントロールプレーン / Leaf0	Controller、ComputeLeaf0、CephStorageLeaf0	192.168.10.0/24
Storage	Controller、ComputeLeaf0、CephStorageLeaf0	172.16.0.0/24
StorageMgmt	Controller、CephStorageLeaf0	172.17.0.0/24
InternalApi	Controller、ComputeLeaf0	172.18.0.0/24
Tenant	Controller、ComputeLeaf0	172.19.0.0/24
External	Controller、ComputeLeaf0	10.1.1.0/24

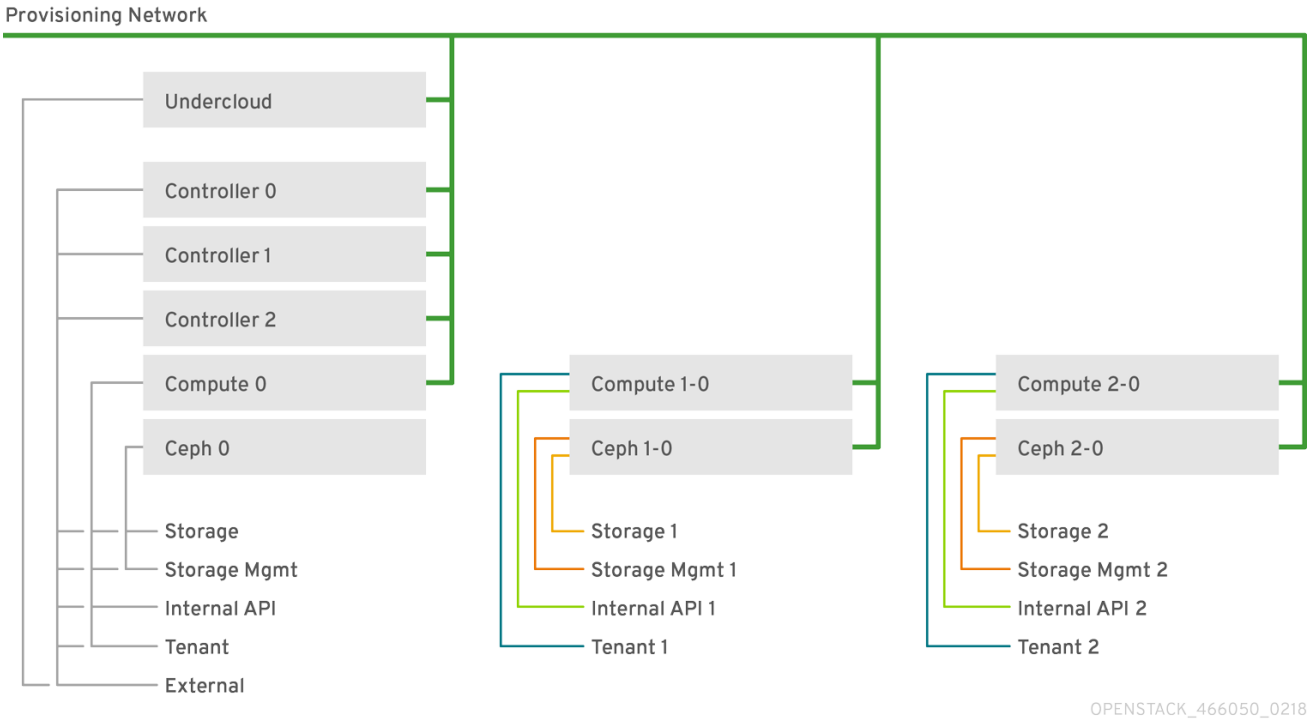
表1.2 Leaf 1 ネットワーク

ネットワーク	アタッチされているロール	サブネット
プロビジョニング / コントロールプレーン / 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	ComputeLeaf1	172.19.1.0/24

表1.3 Leaf 2 ネットワーク

ネットワーク	アタッチされているロール	サブネット
プロビジョニング / コントロールプレーン / Leaf2	ComputeLeaf2、CephStorageLeaf2	192.168.12.0/24
StorageLeaf2	ComputeLeaf2、CephStorageLeaf2	172.16.2.0/24

ネットワーク	アタッチされているロール	サブネット
StorageMgmtLeaf2	CephStorageLeaf2	172.17.2.0/24
InternalApiLeaf2	ComputeLeaf2	172.18.2.0/24
TenantLeaf2	ComputeLeaf2	172.19.2.0/24



1.3. スパイン/リーフ型ネットワークの要件

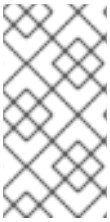
L3 ルーティング対応アーキテクチャーを使用したネットワーク上でオーバークラウドをデプロイするには、前提条件として以下の手順を実施します。

レイヤー 3 ルーティング

ネットワークインフラストラクチャーのルーティングを設定し、異なる L2 セグメント間のトラフィックを有効にします。このルーティングは、静的または動的に設定することができます。

DHCP リレー

アンダークラウドにローカルではない各 L2 セグメントには、**dhcp-relay** を指定する必要があります。DHCP 要求は、アンダークラウドが接続されているプロビジョニングネットワークのセグメントでアンダークラウドに対して送信する必要があります。



注記

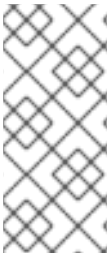
アンダークラウドは 2 つの DHCP サーバーを使用します。1 つは、ベアメタルノードのイントロスペクション用で、もう 1 つはオーバークラウドノードのデプロイ用です。**dhcp-relay** を設定する場合は、DHCP リレーの設定を読んで要件を理解するようにしてください。

1.4. スパイン/リーフ型ネットワークの制限事項

- Controller ロールなどの一部のロールは、仮想 IP アドレスとクラスタリングを使用します。この機能の背後にあるメカニズムには、ノード間の L2 ネットワーク接続が必要です。これらのノードを同じリーフ内に配置する必要があります。

- Networker ノードにも同様の制限が適用されます。Virtual Router Redundancy Protocol (VRRP) により、ネットワークサービスはネットワーク内に高可用性のデフォルトパスを実装します。VRRP は仮想ルーターの IP アドレスを使用するので、マスターとバックアップノードを同じ L2 ネットワークセグメントに接続する必要があります。

- テナントまたはプロバイダーネットワークを VLAN セグメンテーションと共に使用する場合には、すべての Networker ノードおよびコンピュートノード間で特定の VLAN を共有する必要があります。



注記

ネットワークサービスは、複数の Networker ノードセットで設定することが可能です。各 Networker ノードセットはそれらのネットワークのルートを共有し、VRRP が各 Networker ノードセット内の高可用性のデフォルトパスを提供します。この種別の構成では、ネットワークを共有する全 Networker ノードが同じ L2 ネットワークセグメント上になければなりません。

第2章 アンダークラウドでのルーティング対応スパイン/リーフの設定

本項では、コンポーザブルネットワークを使用するルーティング対応のスパイン/リーフを取り入れるための、アンダークラウド設定方法のユースケースについて説明します。

2.1. スパイン/リーフ用のプロビジョニングネットワークの設定

スパイン/リーフインフラストラクチャー用のプロビジョニングネットワークを設定するには、**undercloud.conf** ファイルを編集して、以下の手順で説明する該当パラメーターを設定します。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. **undercloud.conf** ファイルがまだない場合には、サンプルのテンプレートファイルをコピーします。

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

3. **undercloud.conf** ファイルを編集します。
4. **[DEFAULT]** セクションに以下の値を設定します。
   a. **local_ip** を **leaf0** 上のアンダークラウド IP に設定します。

```
local_ip = 192.168.10.1/24
```

   b. **undercloud_public_vip** をアンダークラウドの外部向け IP アドレスに設定します。

```
undercloud_public_vip = 10.1.1.1
```

   c. **undercloud_admin_vip** をアンダークラウドの管理用 IP アドレスに設定します。この IP アドレスは、通常 leaf0 上にあります。

```
undercloud_admin_vip = 192.168.10.2
```

   d. **local_interface** を、ローカルネットワーク用にブリッジを構成するインターフェースに設定します。

```
local_interface = eth1
```

   e. **enable_routed_networks** を **true** に設定します。

```
enable_routed_networks = true
```

   f. **subnets** パラメーターを使用してサブネットの一覧を定義します。ルーティング対応のスパイン/リーフ内の各 L2 セグメントにサブネットを 1 つ定義します。

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

g. **local_subnet** パラメーターを使用して、アンダークラウドにローカルな物理 L2 セグメントに関連付けられるサブネットを指定します。

```
local_subnet = leaf0
```

h. **undercloud_nameservers** の値を設定します。

```
undercloud_nameservers = 10.11.5.19,10.11.5.20
```

ヒント

アンダークラウドのネームサーバーに使用する DNS サーバーの現在の IP アドレスは、/etc/resolv.conf を参照して確認することができます。

5. **subnets** パラメーターで定義するサブネットごとに、新規セクションを作成します。

```
[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** ファイルを保存します。

7. アンダークラウドをインストールするコマンドを実行します。

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

この設定により、プロビジョニングネットワークまたはコントロールプレーン上に 3 つのサブネットが作成されます。オーバークラウドは、各ネットワークを使用して対応する各リーフ内にシステムをプロビジョニングします。

アンダークラウドへの DHCP 要求が適切にリレーされるようにするには、DHCP リレーを設定する必要がある場合があります。

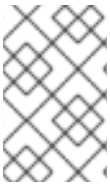
2.2. DHCP リレーの設定

アンダークラウドは、プロビジョニングネットワーク上の 2 つの DHCP サーバーを使用します。

- イントロスペクション DHCP サーバー。
- プロビジョニング DHCP サーバー。

DHCP リレーを設定する際には、アンダークラウド上の両方の DHCP サーバーに DHCP 要求を転送するようにしてください。

UDP ブロードキャストに対応するデバイスでUDP ブロードキャストを使用して、アンダークラウドのプロビジョニングネットワークが接続されている L2 ネットワークセグメントに DHCP 要求をリレーすることができます。または、DHCP 要求を特定の IP アドレスにリレーする UDP ユニキャストを使用することができます。



注記

特定のデバイス種別での DHCP リレーの設定は、本書の対象外となっています。本ガイドでは参考として、ISC DHCP ソフトウェアの実装を使用した DHCP リレー設定の例を説明します。詳細は、dhcrelay(8) の man ページを参照してください。

ブロードキャスト DHCP リレー

この方法では、UDP ブロードキャストトラフィックを使用して DHCP 要求を、DHCP サーバーが存在する L2 ネットワークセグメントにリレーします。ネットワークセグメント上のすべてのデバイスがブロードキャストトラフィックを受信します。UDP ブロードキャストを使用する場合は、アンダークラウド上の両方の DHCP サーバーがリレーされた DHCP 要求を受信します。実装に応じて、インターフェースまたは IP ネットワークアドレスを指定して設定できます。

インターフェース

DHCP 要求がリレーされる L2 ネットワークセグメントに接続されるインターフェースを指定します。

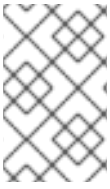
IP ネットワークアドレス

DHCP 要求がリレーされる IP ネットワークのネットワークアドレスを指定します。

ユニキャスト DHCP リレー

この方法では、UDP ユニキャストトラフィックを使用して DHCP 要求を特定の DHCP サーバーにリレーします。UDP ユニキャストを使用する場合には、DHCP リレーを提供するデバイスが、アンダークラウド上のイントロスペクション用に使用されるインターフェースに割り当てられた IP アドレスと、**ctlplane** ネットワーク用の DHCP サービスをホストするために OpenStack Networking (neutron) サービスが作成するネットワーク名前空間の IP アドレスの両方に対して、DHCP 要求をリレーするように設定する必要があります。

イントロスペクションに使用されるインターフェースは、**undercloud.conf** ファイルで **inspection_interface** として定義されるインターフェースです。このパラメーターを設定していない場合には、アンダークラウドのデフォルトインターフェースは **br-ctlplane** になります。



注記

br-ctlplane インターフェースをイントロスペクションに使用するのは一般的です。**undercloud.conf** ファイルで **local_ip** として定義する IP アドレスは、 **br-ctlplane** インターフェース上にあります。

Neutron DHCP 名前空間に確保される IP アドレスは、**undercloud.conf** ファイルの **local_subnet** で設定する IP 範囲内で利用可能な最初のアドレスです。IP 範囲内の最初のアドレスは、設定の **dhcp_start** で定義するアドレスです。たとえば、以下の設定を使用する場合、**192.168.10.10** がその IP アドレスになります。

```
[DEFAULT]
local_subnet = leaf0
subnets = leaf0,leaf1,leaf2

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

警告

DHCP 名前空間の IP アドレスは自動的に割り当てられます。多くの場合、これは IP 範囲の最初のアドレスになります。これを確認するには、アンダークラウドで以下のコマンドを実行します。

```
$ 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 の設定例

以下の例では、**dhcp** パッケージの **dhcrelay** コマンドは以下の設定を使用します。

- DHCP の受信要求をリレーするインターフェースは **eth1**、**eth2**、**eth3** です。
- ネットワークセグメント上のアンダークラウドの DHCP サーバーが接続されているインターフェースは **eth0** です。
- イントロスペクションに使用される DHCP サーバーがリッスンしている IP アドレスは **192.168.10.1** です。

- プロビジョニングに使用される DHCP サーバーがリッスンしている IP アドレスは **192.168.10.10** です。

これで、**dhcrelay** コマンドは以下のようになります。

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

Cisco IOS ルーティングスイッチの設定例

この例では、次のタスクを実行するために、以下に示す Cisco IOS 設定を使用しています。

- プロビジョニングネットワークに使用する VLAN を設定する。
- リーフの IP アドレスを追加する。
- IP アドレス **192.168.10.1** をリッスンするイントロスペクション用 DHCP サーバーに、UDP および BOOTP 要求を転送する。
- IP アドレス **192.168.10.10** をリッスンするプロビジョニング用 DHCP サーバーに、UDP および BOOTP 要求を転送する。

```
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
!
```

これでプロビジョニングネットワークの設定が完了したので、残りのオーバークラウドリーフネットワークを設定することができます。

2.3. リーフネットワーク向けのフレーバーの作成とノードのタグ付け

各リーフネットワークのそれぞれのロールには、フレーバーとロールの割り当てが必要です。これにより、ノードを対応するリーフにタグ付けすることができます。各フレーバーを作成してロールに割り当てるには、以下の手順を実施します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 各カスタムロール用のフレーバーを作成します。

```
$ ROLES="control compute_leaf0 compute_leaf1 compute_leaf2 ceph-storage_leaf0 ceph-storage_leaf1 ceph-storage_leaf2"
$ for ROLE in $ROLES; do openstack flavor create --id auto --ram 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" --property resources:CUSTOM_BAREMETAL='1' --property resources:DISK_GB='0' --property resources:MEMORY_MB='0' --property resources:VCPU='0' $ROLE ; done
```

3. 対応するリーフネットワークにノードをタグ付けします。たとえば、以下のコマンドを実行して、UUID **58c3d07e-24f2-48a7-bbb6-6843f0e8ee13** のノードを Leaf2 上の Compute ロールにタグ付けします。

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

4. フレーバーからロールへのマッピングが含まれた環境ファイル (**~/templates/node-data.yaml**) を作成します。

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

各 ***Count** パラメーターを使用して、オーバークラウド内にデプロイするノード数を設定することもできます。

2.4. ベアメタルノードのポートからコントロールプレーンのネットワークセグメントへのマッピング

L3 ルーティング対応のネットワーク上でのデプロイメントを有効にするには、ベアメタルポートの **physical_network** フィールドを設定する必要があります。各ベアメタルポートは、OpenStack Bare Metal (ironic) サービス内のベアメタルノードに関連付けられます。物理ネットワーク名は、アンダークラウドの設定の **subnets** オプションで指定する名前です。

注記

undercloud.conf ファイルの **local_subnet** で指定されるサブネットの物理ネットワーク名には、必ず **ctlplane** という名前が付けられます。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. ベアメタルノードをチェックします。

```
$ openstack baremetal node list
```

3. ベアメタルノードは **enroll** または **manageable** の状態であることを確認してください。ベアメタルノードがこれらのいずれかの状態にない場合、ベアメタルポートで **physical_network** プロパティーを設定するコマンドは失敗します。全ノードを **manageable** の状態に設定するには、以下のコマンドを実行します。

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

4. ベアメタルポートとベアメタルノードの関連付けを確認します。

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

5. ポートの **physical-network** パラメーターを設定します。以下の例では、**leaf0**、**leaf1**、および **leaf2** の 3 つのサブネットが設定で定義されています。local_subnet は **leaf0** です。**local_subnet** の物理ネットワークは常に **ctlplane** であるため、**leaf0** に接続されたベアメタルポートは必ず ctlplane を使用します。残りのポートは他のリーフ名を使用します。

```
$ 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. オーバークラウドをデプロイする前にノードが使用可能な状態であることを確認します。

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

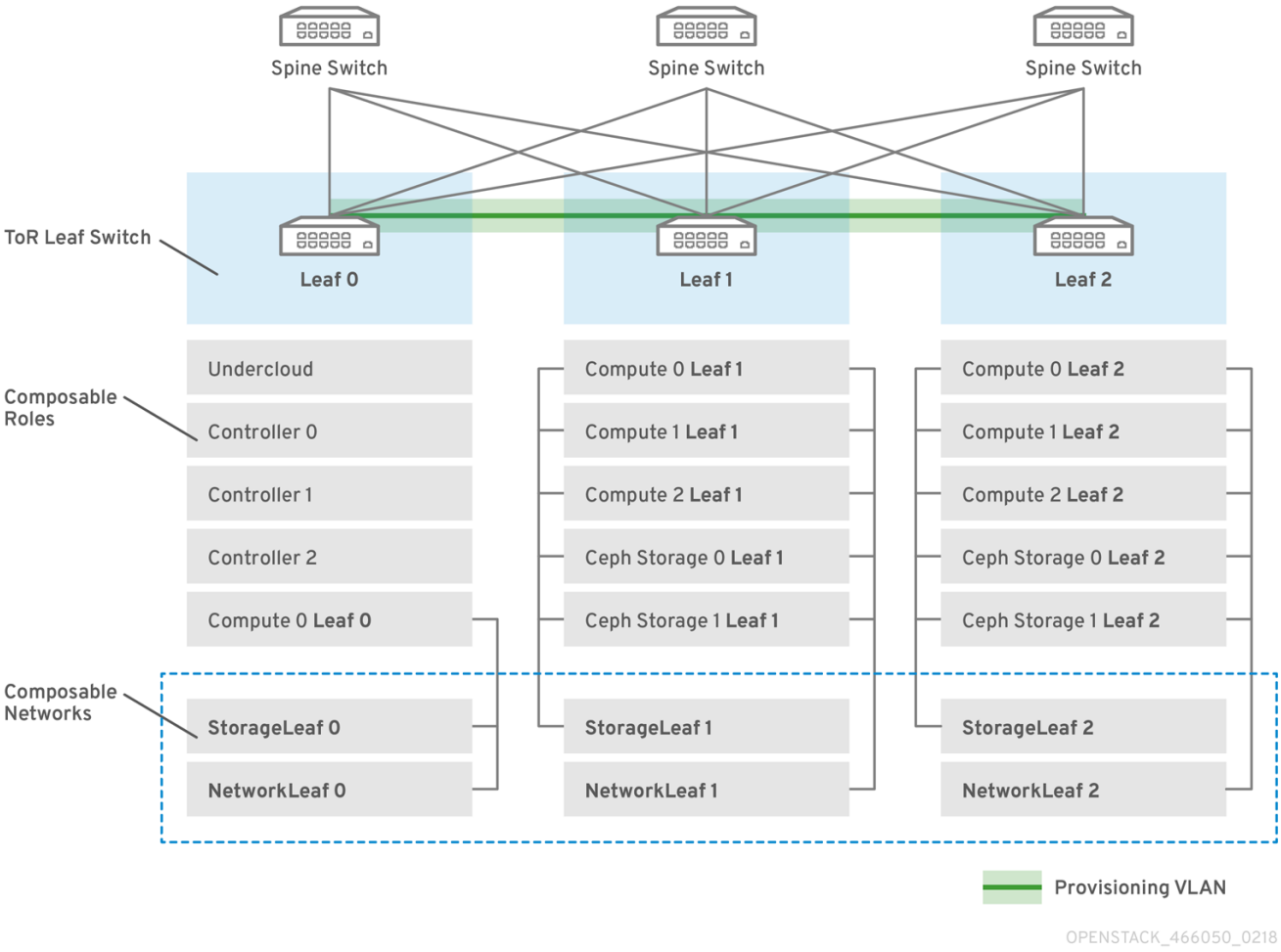
第3章 その他のプロビジョニングネットワーク設定手法

本項では、コンポーザブルネットワークを使用するルーティング対応のスパイン/リーフを取り入れるための、プロビジョニングネットワークを設定するのに使用できるその他の方法について説明します。

3.1. VLAN プロビジョニングネットワーク

以下の例では、director はプロビジョニングネットワークを通じて新たなオーバークラウドノードをデプロイし、L3 トポロジー全体にまたがる VLAN トンネルを使用します。詳細は、「図3.1「VLAN プロビジョニングネットワークトポロジー」」を参照してください。VLAN プロビジョニングネットワークを使用すると、director の DHCP サーバーは、任意のリーフに **DHCPOFFER** ブロードキャストを送信することができます。このトンネルを確立するには、トップオブラック (ToR) リーフスイッチ間で VLAN をトランク接続します。以下の図では、**StorageLeaf** ネットワークは Ceph Storage とコンピュートノードに提示されます。**NetworkLeaf** は、構成する任意のネットワークの例を示します。

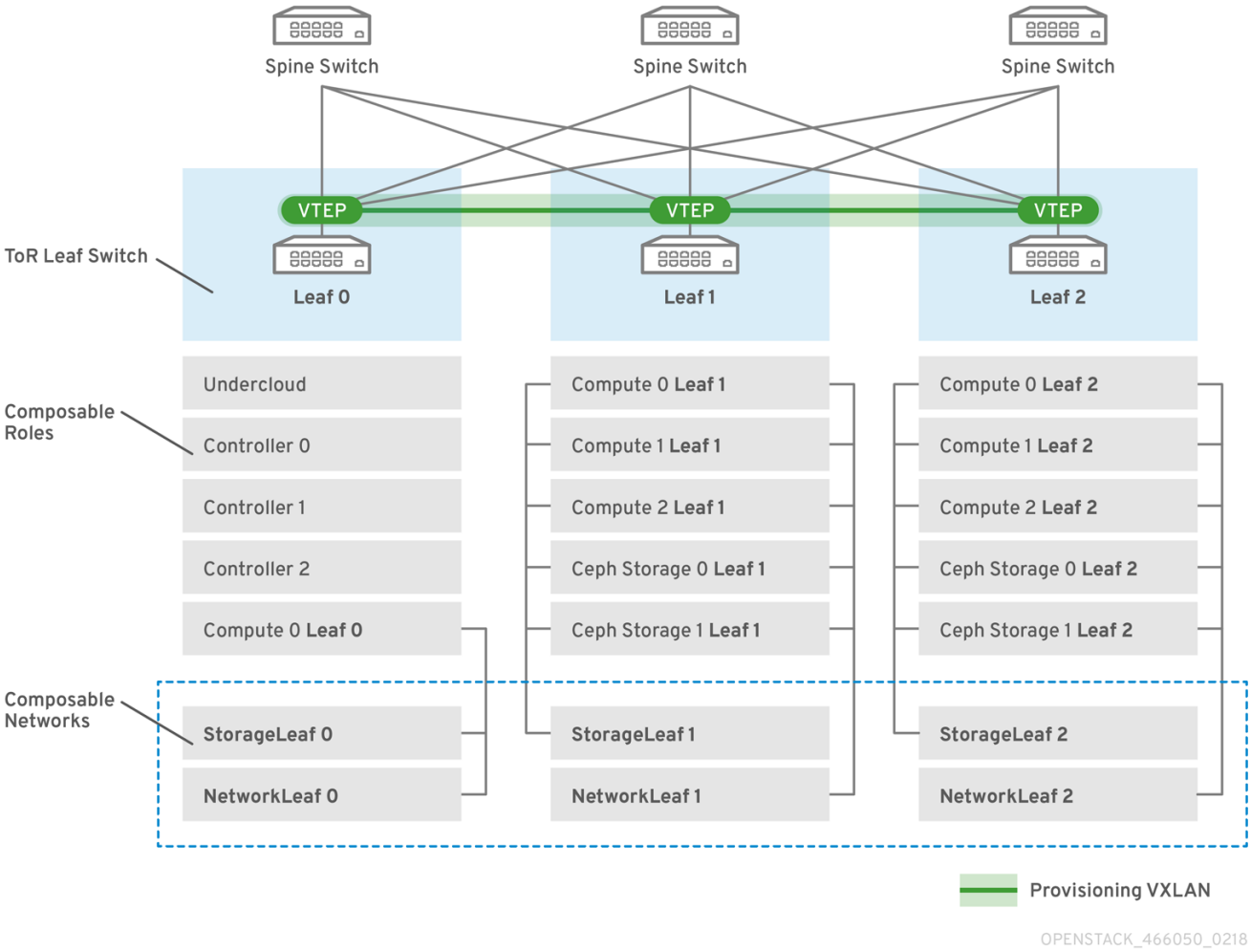
図3.1 VLAN プロビジョニングネットワークトポロジー



3.2. VXLAN プロビジョニングネットワーク

以下の例では、director はプロビジョニングネットワークを通じて新たなオーバークラウドノードをデプロイし、レイヤー 3 トポロジー全体をカバーするために VXLAN トンネルを使用します。詳細は、「図3.2「VXLAN プロビジョニングネットワークトポロジー」」を参照してください。VXLAN プロビジョニングネットワークを使用すると、director の DHCP サーバーは、任意のリーフに **DHCPOFFER** ブロードキャストを送信することができます。このトンネルを確立するには、トップオブラック (ToR) リーフスイッチで VXLAN エンドポイントを設定します。

図3.2 VXLAN プロビジョニングネットワークトポロジー



第4章 オーバークラウドの設定

アンダークラウドを設定したら、設定ファイルのセットを使用して残りのオーバークラウドリーフネットワークを設定することができます。残りのオーバークラウドリーフネットワークを設定してオーバークラウドをデプロイすると、デプロイされる環境には、ルーティングを利用できるネットワークが複数セット実装されます。

4.1. ネットワークデータファイルの作成

リーフネットワークを定義するには、コンポーザブルネットワークとその属性の一覧が YAML 形式で記載された、ネットワークデータファイルを作成します。**subnets** パラメーターを使用して、ベースネットワークと共に追加のリーフサブネットを定義します。

手順

1. **stack** ユーザーのホームディレクトリーに、新たな **network_data_spine_leaf.yaml** ファイルを作成します。デフォルトの **network_data_subnets_routed.yaml** ファイルをベースとして使用します。

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

2. **network_data_spine_leaf.yaml** ファイルの YAML 一覧を編集して、各ベースネットワークおよび対応するリーフサブネットをコンポーザブルネットワーク項目として定義します。以下に示す構文の例を使用して、ベースのリーフおよび 2 つのリーフサブネットを定義します。

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

以下の例は、内部 API ネットワークおよびそのリーフネットワークを定義する方法を示しています。

```
- name: InternalApi
  name_lower: internal_api
  vip: true
  vlan: 10
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
```

```
  gateway_ip: '172.18.0.1'
  subnets:
    internal_api_leaf1:
      vlan: 11
      ip_subnet: '172.18.1.0/24'
      allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
      gateway_ip: '172.18.1.1'
    internal_api_leaf2:
      vlan: 12
      ip_subnet: '172.18.2.0/24'
      allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]
      gateway_ip: '172.18.2.1'
```

注記

Control Plane ネットワークは、アンダークラウドですでに作成済みなので、ネットワークデータファイルでは定義しません。ただし、パラメーターを手動で設定して、オーバークラウドが NIC を適切に設定できるようにする必要があります。

注記

Controller ベースのサービスが含まれるネットワークについて、**vip: true** と定義します。上記の例では、**InternalApiLeaf0** にそれらのサービスが含まれます。

4.2. ロールデータファイルの作成

各リーフ用のそれぞれのコンポーザブルロールを定義して、コンポーザブルネットワークをそれぞれのロールに接続するには、以下の手順を実施します。

手順

1. **stack** ユーザーのホームディレクトリーに、カスタム **roles** ディレクトリーを作成します。

```
$ mkdir ~/roles
```

2. デフォルトの Controller、Compute、Ceph Storage ロールを、director のコアテンプレートコレクションから roles ディレクトリーにコピーします。Leaf 0 に合わせて、Compute および Ceph Storage 用ファイルの名前を変更します。

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

3. Leaf 0 の Compute および Ceph Storage ファイルをベースとして、Leaf 1 および Leaf 2 ファイル用にコピーを作成します。

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

4. それぞれのリーフパラメーターと整合するように、Leaf 0、Leaf 1、および Leaf 2 ファイルの

name、**HostnameFormatDefault**、および **deprecated_nic_config_name** パラメーターを編集します。たとえば、Leaf 0 Compute ファイルのパラメーター値は、以下のように設定します。

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

Leaf 0 Ceph Storage ファイルのパラメーター値は、以下のように設定します。

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

5. それぞれのリーフネットワークのパラメーターと整合するように、Leaf 1 および Leaf 2 ファイルの **networks** パラメーターを編集します。たとえば、Leaf 1 Compute ファイルのパラメーター値は、以下のように設定します。

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

Leaf 1 Ceph Storage ファイルのパラメーター値は、以下のように設定します。

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

注記

この設定を行うのは、Leaf 1 および Leaf 2 だけです。Leaf 0 の **networks** パラメーターは、ベースサブネットの値のままにします (各サブネットの小文字を使用した名前に接尾辞 **_subnet** を追加したもの)。たとえば、Leaf 0 の内部 API は **internal_api_subnet** となります。

6. ロールの設定が完了したら、以下のコマンドを実行して完全なロールデータファイルを生成します。

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

これにより、各リーフネットワーク用の全カスタムロールが含まれた完全な **roles_data_spine_leaf.yaml** ファイルが作成されます。

ロールごとに独自の NIC 設定があります。スパイン/リーフ構成を設定する前に、現在の NIC 設定に適した NIC テンプレートの基本セットを作成する必要があります。

4.3. カスタム NIC 設定の作成

各ロールには、固有の NIC 設定が必要です。NIC テンプレートの基本セットのコピーを作成し、新しいテンプレートを対応する NIC 設定リソースにマッピングするには、以下の手順を実施します。

手順

1. コア heat テンプレートディレクトリーに移動します。

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

2. **tools/process-templates.py** スクリプト、カスタム **network_data** ファイル、およびカスタム **roles_data** ファイルを使用して、Jinja2 テンプレートをレンダリングします。

```
$ 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. ホームディレクトリーに移動します。

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

4. デフォルト NIC テンプレートのいずれかからコンテンツをコピーし、スパイン/リーフテンプレートのベースとして使用します。たとえば、**single-nic-vlans** NIC テンプレートをコピーするには、以下のコマンドを実行します。

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

5. **/home/stack/templates/spine-leaf-nics/** の各 NIC 設定を編集し、設定スクリプトの場所を絶対パスに変更します。以下のスニペットに示すような、ネットワーク設定のセクションにスクロールします。

```
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:
```

スクリプトの場所を絶対パスに変更します。

```
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:
```

それぞれのリーフの各ファイルについてこの変更を行い、変更を保存します。

注記

NIC をさらに変更する場合には、**『オーバークラウドの高度なカスタマイズ』**の「カスタムネットワークインターフェーステンプレート」を参照してください。

6. **spine-leaf-nics.yaml** という名前のファイルを作成し、そのファイルを編集します。

7. ファイルに **resource_registry** セクションを作成し、***::Net::SoftwareConfig** リソースのセットを追加して、対応する NIC テンプレートにマッピングします。

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

これらのリソースマッピングは、デプロイメント時にデフォルトのリソースマッピングをオーバーライドします。

8. **spine-leaf-nics.yaml** ファイルを保存します。

9. レンダリング済みテンプレートのディレクトリーを削除します。

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

上記の手順を実施すると、NIC テンプレートのセットと共に、必要な ***::Net::SoftwareConfig** リソースをそれらのテンプレートにマッピングする環境ファイルが作成されます。最終的に **openstack overcloud deploy** コマンドを実行する際には、環境ファイルを以下の順番で追加するようにしてください。

1. **/usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml**: ネットワーク分離を有効にします。director はこのファイルを **network-isolation.j2.yaml** Jinja2 テンプレートからレンダリングする点に注意してください。

2. **/usr/share/openstack-tripleo-heat-templates/environments/network-environment.yaml**: デフォルトの NIC リソースマッピングが含まれるデフォルトのネットワーク環境ファイル。director はこのファイルを **network-environment.j2.yaml** Jinja2 テンプレートからレンダリングする点に注意してください。

3. **/home/stack/templates/spine-leaf-nics.yaml**: カスタム NIC リソースマッピングが含まれるこのファイルは、デフォルトの NIC リソースマッピングをオーバーライドします。

以下のコマンドスニペットに、環境ファイルの順番を示します。

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

リソース

- NIC テンプレートのカスタマイズに関する詳しい情報は、『**オーバークラウドの高度なカスタマイズ**』の「カスタムネットワークインターフェーステンプレート」を参照してください。

ネットワーク環境ファイルに情報を追加してスパイン/リーフアーキテクチャーの特定機能を定義するには、以降のセクションに記載する手順を実施します。この設定を完了したら、このファイルを **openstack overcloud deploy** コマンドに追加します。

4.4. コントロールプレーンのパラメーターの設定

分離されたスパイン/リーフネットワークのネットワーク詳細の定義には、通常 **network_data** ファイルを使用します。コントロールプレーンネットワークは例外で、これはアンダークラウドによって作成されます。ただし、オーバークラウドには、各リーフのコントロールプレーンへのアクセスが必要です。このアクセスを有効にするには、追加のパラメーターを **network-environment.yaml** ファイルに含める必要があります。

以下の例では、Leaf 0 上の対応するコントロールプレーンネットワークの IP、サブネット、デフォルトルートを定義します。

手順

1. **spine-leaf-ctlplane.yaml** という名前のファイルを作成し、そのファイルを編集します。

2. ファイルに **parameter_defaults** セクションを作成し、それぞれのスパイン/リーフネットワークについてコントロールプレーンのサブネットのマッピングを追加します。

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

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

3. **spine-leaf-ctlplane.yaml** ファイルを保存します。

4.5. 仮想 IP アドレス用サブネットの設定

通常、Controller ロールは各ネットワークの仮想 IP (VIP) アドレスをホストします。デフォルトでは、コントロールプレーンを除き、オーバークラウドは各ネットワークのベースサブネットから仮想 IP を取得します。コントロールプレーンは、標準のアンダークラウドのインストール時に作成されたデフォルトのサブネット名である **ctlplane-subnet** を使用します。

以下のスパイン/リーフのシナリオでは、デフォルトのベースプロビジョニングネットワークは **ctlplane-subnet** ではなく **leaf0** です。つまり、**VipSubnetMap** パラメーターにオーバーライド用の値を追加して、コントロールプレーンの仮想 IP が使用するサブネットを変更する必要があります。

また、各ネットワークの仮想 IP がそのネットワークのベースサブネットを使用しない場合、**VipSubnetMap** パラメーターにさらにオーバーライド値を追加して、director がコントローラーノードに接続する L2 ネットワークセグメントに関連付けられたサブネット上の仮想 IP を作成するようにしなければなりません。

手順

1. **spine-leaf-vips.yaml** という名前のファイルを作成し、そのファイルを編集します。
2. ファイルに **parameter_defaults** セクションを作成し、それぞれの要件に応じて **VipSubnetMap** パラメーターを追加します。
   - プロビジョニング / コントロールプレーンネットワークに **leaf0** を使用する場合には、**ctlplane** の仮想 IP 再マッピングを **leaf0** に設定します。

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

   - 複数の仮想 IP に異なるリーフを使用する場合には、要求に応じて仮想 IP の再マッピングを設定します。たとえば、以下のスニペットを使用して、すべての仮想 IP に **leaf1** を使用するように **VipSubnetMap** パラメーターを設定します。

     ```
     parameter_defaults:
       VipSubnetMap:
         ctlplane: leaf1
         redis: internal_api_leaf1
         InternalApi: internal_api_leaf1
         Storage: storage_leaf1
         StorageMgmt: storage_mgmt_leaf1
     ```

3. **spine-leaf-vips.yaml** ファイルを保存します。

4.6. 異なるネットワークのマッピング

デフォルトでは、OpenStack Platform は Open Virtual Network (OVN) を使用します。この場合、外部ネットワークへのアクセス用に、すべてのコントローラーおよびコンピュートノードを 1 つの L2 ネッ

トワークに接続する必要があります。つまり、コントローラーおよびコンピュートネットワークの設定は、共に **br-ex** ブリッジを使用し、director はデフォルトではこのブリッジをオーバークラウド内の **datacentre** ネットワークにマッピングします。このマッピングは、通常フラットネットワークマッピングまたは VLAN ネットワークマッピングのいずれかに使用されます。スパイン/リーフのアーキテクチャーでは、各リーフがそのリーフ上の特定のブリッジまたは VLAN 経由でトラフィックをルーティングするように、これらのマッピングを変更することができます。この設定は、エッジコンピューティングのシナリオで頻繁に使用されます。

手順

1. **spine-leaf-separate.yaml** という名前のファイルを作成し、そのファイルを編集します。
2. **spine-leaf-separate.yaml** ファイルに **parameter_defaults** セクションを作成し、それぞれのスパイン/リーフネットワークについて外部ネットワークのマッピングを追加します。
   - フラットネットワークのマッピングの場合には、**NeutronFlatNetworks** パラメーターに各リーフの一覧を定義し、各リーフの **NeutronBridgeMappings** パラメーターを設定します。

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

   - VLAN ネットワークのマッピングの場合には、さらに **NeutronNetworkVLANRanges** を設定して、3 つすべてのリーフネットワーク用に VLAN をマッピングします。

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

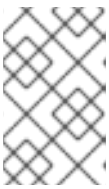
3. **spine-leaf-separate.yaml** ファイルを保存します。

4.7. スパイン/リーフ対応のオーバークラウドのデプロイ

スパイン/リーフ型オーバークラウドの設定が完了したら、以下の手順を実施して各ファイルを確認してからデプロイメントコマンドを実行します。

手順

1. **/home/stack/template/network_data_spine_leaf.yaml** ファイルをチェックして、各リーフ用のネットワークおよびサブネットがすべて含まれていることを確認します。



注記

現在、ネットワークサブネットおよび **allocation_pools** の値に対する自動検証はありません。これらの値を一貫性を持って定義し、既存のネットワークと競合しないようにしてください。

2. **/home/stack/templates/roles_data_spine_leaf.yaml** の値をチェックして、各リーフのロールが定義されていることを確認します。

3. **~/templates/spine-leaf-nics/** ディレクトリーの NIC テンプレートをチェックして、各リーフ上のそれぞれのロールのインターフェースが正しく定義されていることを確認します。

4. **spine-leaf-nics.yaml** カスタム環境ファイルをチェックして、各ロールのカスタム NIC テンプレートを参照する **resource_registry** セクションが含まれていることを確認します。

5. **/home/stack/templates/nodes_data.yaml** ファイルをチェックして、全ロールにフレーバーとノード数が割り当てられていることを確認します。また、各リーフのノードがすべて適切にタグ付けされていることを確認します。

6. **openstack overcloud deploy** コマンドを実行して、スパイン/リーフの設定を適用します。以下に例を示します。

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

   - **network-isolation.yaml** は、同じ場所にある Jinja2 ファイル (**network-isolation.j2.yaml**) のレンダリング後の名前です。デプロイメントコマンドにこのファイルを追加して、director が各ネットワークを正しいリーフに分離できるようにします。これにより、ネットワークはオーバークラウドの作成プロセス中に動的に作成されるようになります。

   - **network-isolation.yaml** ファイルの後に **network-environment.yaml** ファイルを指定します。**network-environment.yaml** ファイルにより、コンポーザブルネットワークパラメーターのデフォルトネットワーク設定が提供されます。

   - **network-environment.yaml** ファイルの後に **spine-leaf-nics.yaml** ファイルを指定します。**spine-leaf-nics.yaml** ファイルは、**network-environment.yaml** ファイルで定義されるデフォルト NIC テンプレートマッピングをオーバーライドします。

   - これ以外にスパイン/リーフネットワークの環境ファイルを作成した場合には、**spine-leaf-nics.yaml** ファイルの後にそれらの環境ファイルを追加します。

   - 環境ファイルを更に追加します。(例: コンテナーイメージの場所や Ceph クラスターの設定を定義した環境ファイルなど)。

7. スパイン/リーフ対応のオーバークラウドがデプロイされるまで待ちます。