Red Hat OpenStack Platform 13

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: 2023-04-10

Red Hat OpenStack Platform 13 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

OpenStack Team
rhos-docs@redhat.com

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

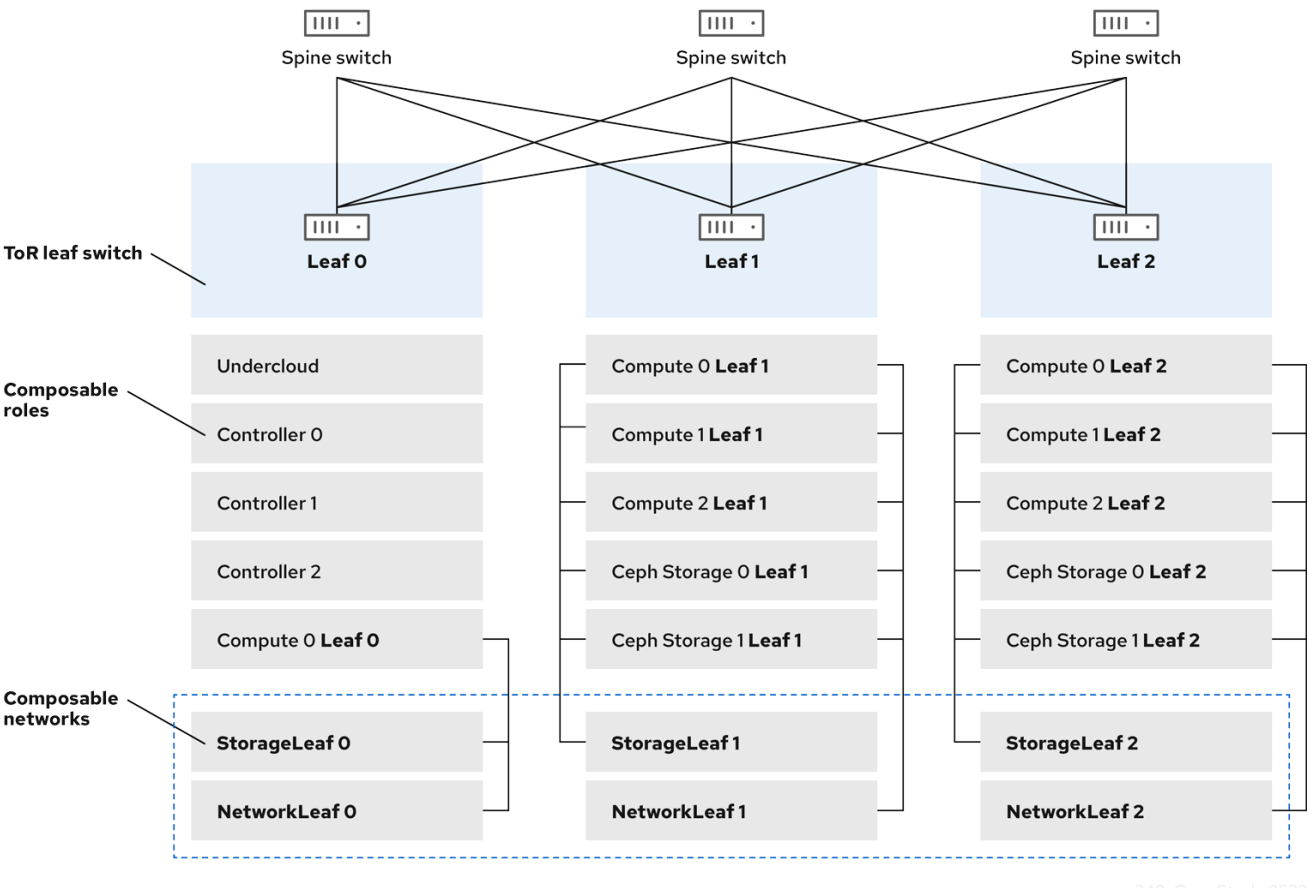
本ガイドは、Red Hat OpenStack Platform 環境に向けたスパイン/リーフ型ネットワークのトポロジーの構築方法についての情報を提供します。これには、完全なエンドツーエンドのシナリオと、お使いの環境でより広範囲なネットワークトポロジーを複製するのに役立つサンプルファイルが含まれます。

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により、ネットワークを、一般的な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、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。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、設定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. ネットワークトポロジー

ルーティング対応のリーフ/スパイン型ベアメタル環境にはレイヤー 3 対応のスイッチが 1 つまたは複数あります。このスイッチは、複数のレイヤー 2 ブロードキャストドメイン内の分離された VLAN 間でトラフィックをルーティングします。

この設計意図は、機能に応じてトラフィックを分離することです。たとえば、コントローラーノードが **Internal API** ネットワーク上で API をホストする場合、コンピュートノードが API にアクセスする際に独自のバージョンの **Internal API** ネットワークを使用する必要があります。このルーティングが機能するには、**Internal API** ネットワークを宛先とするトラフィックが必要なインターフェイスを使用するように強制するルートが必要です。これは、**supernet** ルートを使用して設定することができます。たとえば、**172.18.0.0/24** をコントローラーノード用の **Internal API** ネットワークに使用する場合には、2 番目の **Internal API** ネットワークに **172.18.1.0/24**、3 番目には **172.18.2.0/24**、というように使用できます。その結果、各レイヤー 2 ドメイン内の各ロール向けに、ローカルの **Internal API** ネットワーク上のゲートウェイ IP を使用する、より大きな **172.18.0.0/16** supernet をポイントするルートができます。

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

ネットワーク	アタッチされているロール	インターフェイス	ブリッジ	サブネット
Provisioning / Control Plane	すべて	nic1	br-ctlplane (アンダークラウド)	192.168.10.0/24
Storage	Controller	nic2		172.16.0.0/24
Storage Mgmt	Controller	nic3		172.17.0.0/24
Internal API	Controller	nic4		172.18.0.0/24
Tenant	Controller	nic5		172.19.0.0/24
External	Controller	nic6	br-ex	10.1.1.0/24

表1.2 Leaf 1 ネットワーク

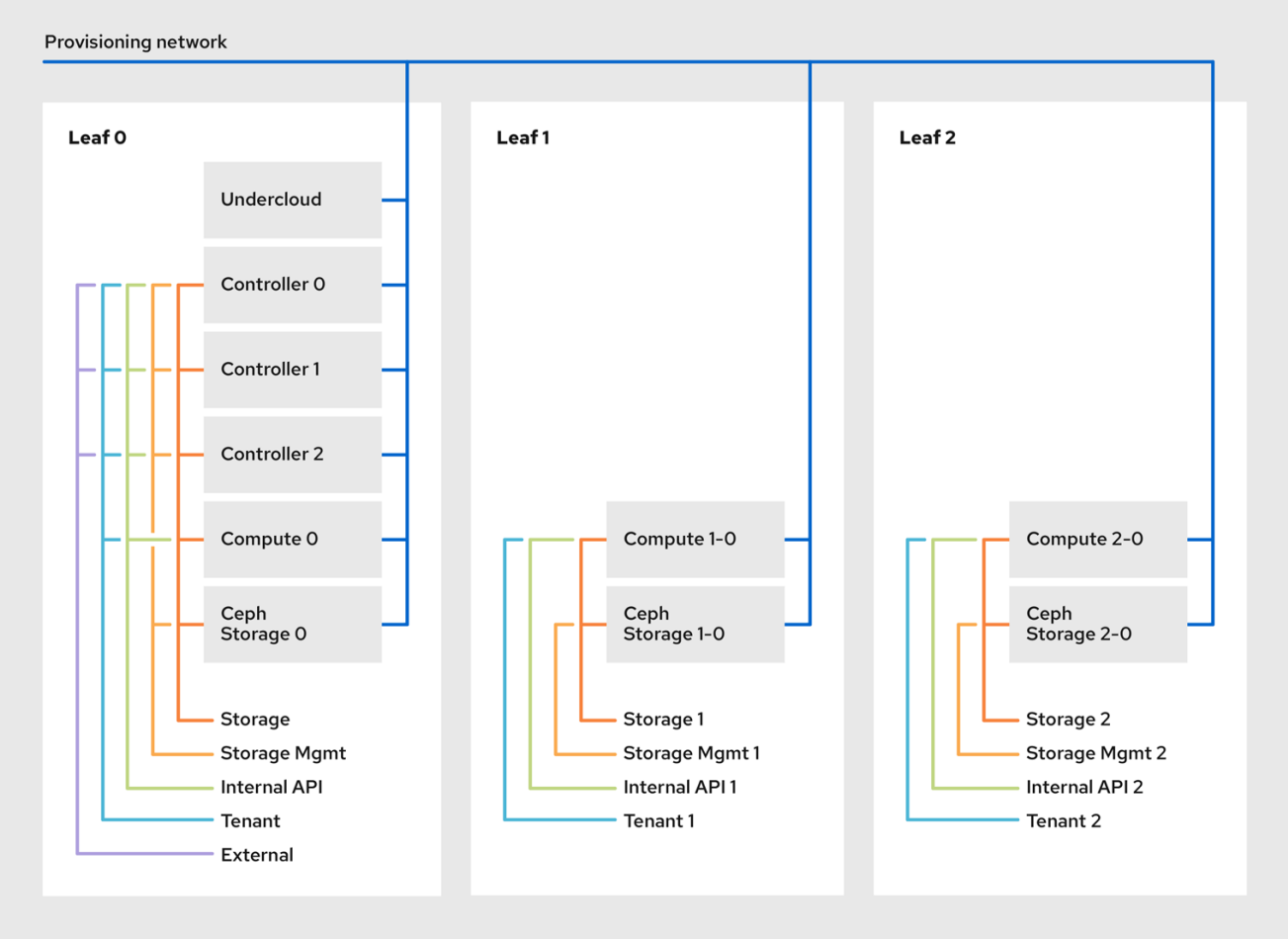
ネットワーク	アタッチされているロール	インターフェイス	ブリッジ	サブネット
Provisioning / Control Plane	すべて	nic1	br-ctlplane (アンダークラウド)	192.168.11.0/24
Storage1	Compute1、Ceph1	nic2		172.16.1.0/24
Storage Mgmt1	Ceph1	nic3		172.17.1.0/24
Internal API1	Compute1	nic4		172.18.1.0/24
Tenant1	Compute1	nic5		172.19.1.0/24

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	インターフェイス	ブリッジ	サブネット
Provisioning / Control Plane	すべて	nic1	br-ctlplane (アンダークラウド)	192.168.12.0/24
Storage2	Compute2、Ceph2	nic2		172.16.2.0/24
Storage Mgmt2	Ceph2	nic3		172.17.2.0/24
Internal API2	Compute2	nic4		172.18.2.0/24
Tenant2	Compute2	nic5		172.19.2.0/24

表1.4 Supernet ルート

ネットワーク	サブネット
Storage	172.16.0.0/16
Storage Mgmt	172.17.0.0/16
Internal API	172.18.0.0/16
Tenant	172.19.0.0/16



249_OpenStack_0522

1.3. スパイン/リーフ型ネットワークの要件

レイヤー 3 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、以下の要件を満たす必要があります。

レイヤー 3 ルーティング

異なるレイヤー 2 セグメント間のトラフィックを有効にするには、ネットワークインフラストラクチャーでルーティングを設定する必要があります。これは、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ない各レイヤー 2 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

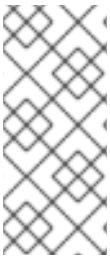
アンダークラウドは 2 つの DHCP サーバーを使用します。1 つは、ベアメタルノードのイントロスペクション用で、もう 1 つはオーバークラウドノードのデプロイ用です。**dhcp-relay** の設定時に要件を理解するには、DHCP リレーの設定を必ず読んでください。

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ー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、ノード間のレイヤー 2 ネットワーク接続が必要です。それらのノードはすべて同じリーフ内に配置されます。

- Networker ノードにも同様の制限が適用されます。ネットワークサービスは、Virtual Router Redundancy Protocol (VRRP) を使用するネットワーク内にデフォルトの高可用性パスを実装します。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。各セットはそれらのネットワークのルートを共有し、VRRP が Networker ノードの各セット内の高可用性のデフォルトパスを提供します。このような設定では、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ある必要があります。

第2章 アンダークラウドの設定

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、以下の手順で定義されているように関連する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** 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/instack-undercloud/undercloud.conf.sample
~/undercloud.conf
```

3. **undercloud.conf** を編集します。
4. **[DEFAULT]** セクションを以下のように編集します。
   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設定するインターフェイスに設定します。

```
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/リーフ内の各レイヤー 2 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** パラメーターでアンダークラウドにローカルな物理レイヤー 2 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
```

5. **subnets** パラメーターで定義されている各サブネットごとに新しい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
```

これにより、プロビジョニングネットワーク / 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。

アンダークラウドに対する 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。次の項では、DHCP リレーの設定方法について説明します。

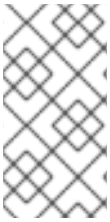
2.2. DHCP リレーの設定

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

- イントロスペクション用 x1
- プロビジョニング用 x1

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 ブロードキャストを対応するデバイスと共に使用して、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、本ガイドの対象範囲外となっています。本ガイドでは参考として、ISC DHCP ソフトウエアの実装を使用した DHCP リレー設定の例を以下に記載しています。この実装の使用方法に関する詳しい情報は、dhcrelay(8) の man ページを参照してください。

ブロードキャストを使用する 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 サーバーが存在する L2 ネットワークセグメントに DHCP 要求をリレーします。このネットワークセグメント上の全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。これは通常、実装に応じて、インターフェイスまたは IP ネットワークアドレスを指定することによって設定されます。

インターフェイス

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するインターフェイスを指定します。

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。

ユニキャストを使用する 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でイントロスペクション用に使用されるインターフェイ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 OpenStack Networking (neutron) サービスによって作成された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イスは、**undercloud.conf** の inspection_interface で定義されているインターフェイスです。

注記

br-ctlplane インターフェイ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** の **local_ip** で定義されている IP アドレスは、**br-ctlplane** インターフェイス上です。

Neutron DHCP 名前空間に割り当てられる IP アドレスは、undercloud.conf の local_subnet で設定されている 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** で定義されているアドレスです。たとえば、以下の設定が使用されている場合には、**192.168.10.10** がその IP アドレスとなります

```
[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。大半の場合は、IP 範囲の最初のアドレスです。アンダークラウドで以下のコマンドを実行して、確認するようにしてください。

```
$ 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イスは **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イスは **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 172.20.0.10 172.20.0.1 \
  -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** をリッスンしてい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** をリッスンしてい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。これは、一連の設定ファイルを使用して行います。

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各ロールには、対応するリーフにノードをタグ付けするためのフレーバーとロールの割り当てが必要です。この手順では、各フレーバーの作成とロールの割り当ての方法を説明します。

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="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. 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。

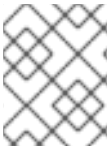
```
parameter_defaults:
  OvercloudController0Flavor: control0
  Controller0Count: 3
```

```
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
```

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でのデプロイメントを有効にするには、ベアメタルのポートの **physical_network** フィールドが設定されている必要があります。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** オプションで使用されている名前です。



注記

undercloud.conf の **local_subnet** に指定されているサブネットの物理ネットワーク名は特別です。これは常に **ctlplane** という名前になります。

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でない場合には、baremetal ポートで physical_network プロパティーの設定に使用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. baremetal ポートが関連付けられている baremetal ノードを確認します。以下に例を示します。

```
$ 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
```

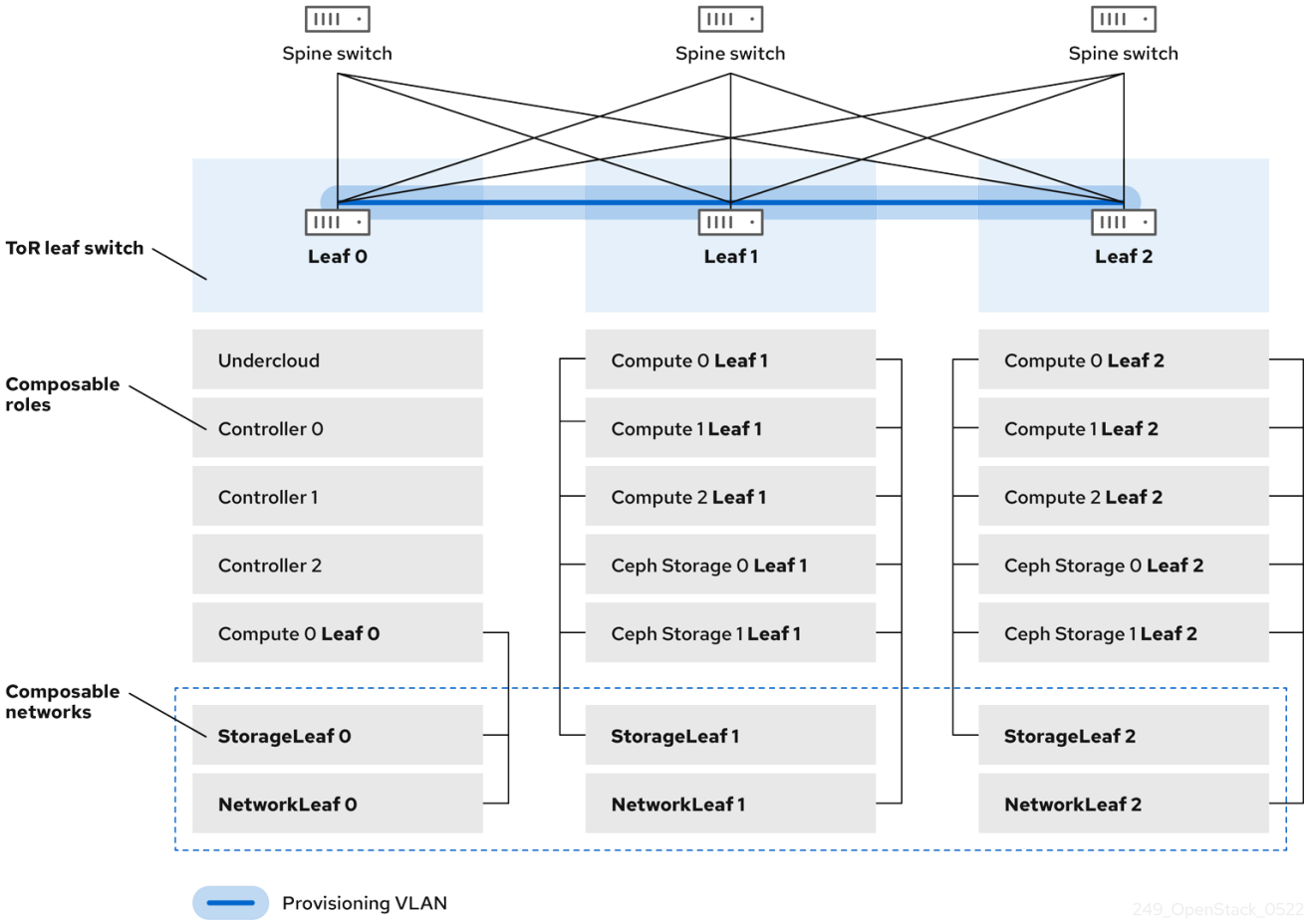
第3章 その他のプロビジョニングネットワーク設定手法

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、その他のプロビジョニングネットワーク設定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、レイヤー 3 トポロジー全体にまたがる VLAN トンネルを使用します (図3.1「VLAN プロビジョニングネットワークトポロジー」を参照)。これにより、director の DHCP サーバーは、どのリーフにも **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、Top-of-Rack (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、設定する任意のネットワークの例を示します。

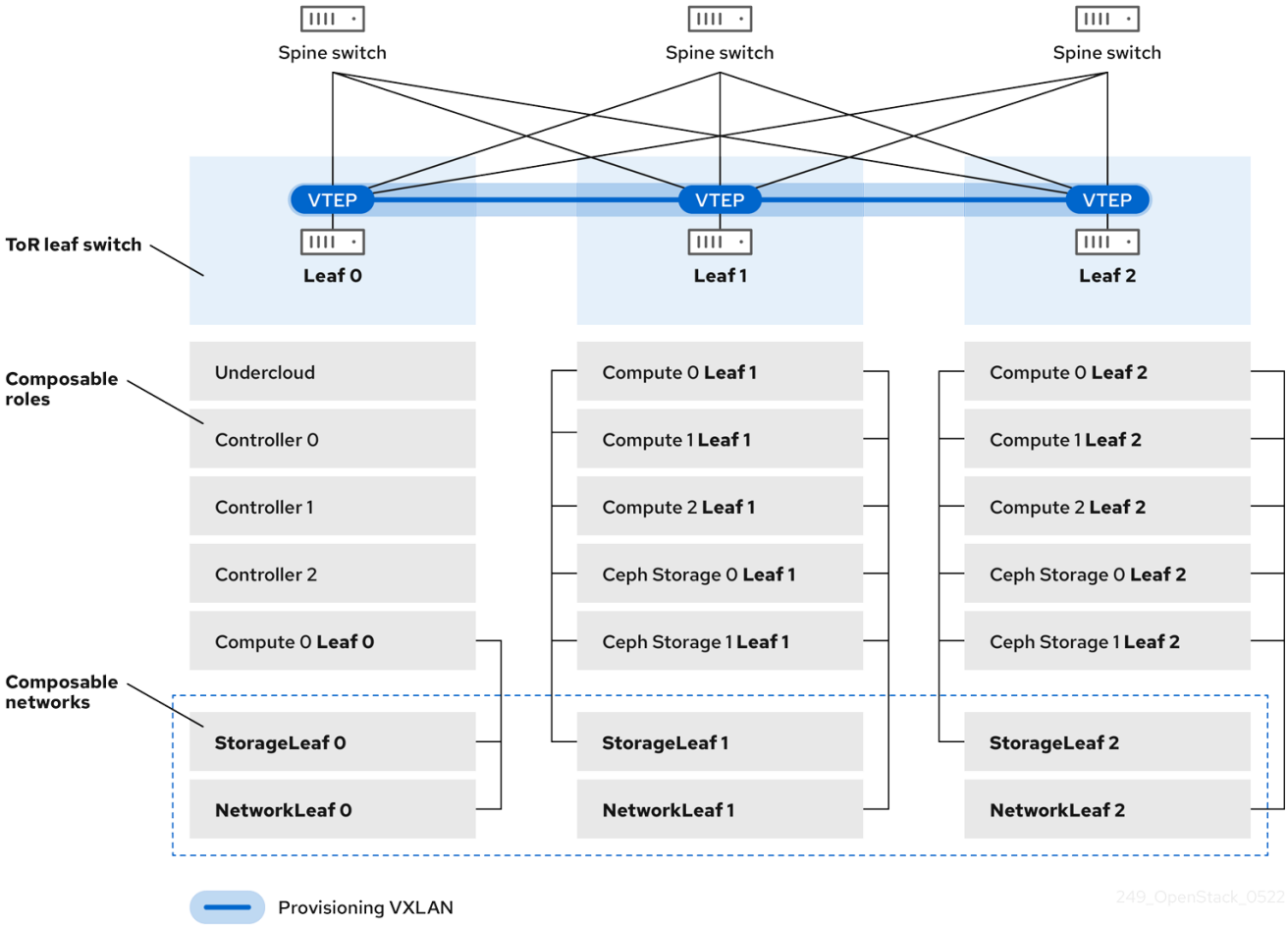
図3.1 VLAN プロビジョニングネットワークトポロジー



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 (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)。これにより、director の DHCP サーバーは、どのリーフにも **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、Top-of-Rack (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。これは、一連の設定ファイルを使用して行います。この作業が終わった後には、オーバークラウドをデプロイします。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。デフォルトのネットワークのデータは、アンダークラウドの **/usr/share/openstack-tripleo-heat-templates/network_data.yaml** にあります。

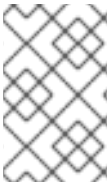
手順

1. **stack** ユーザーのローカルディレクトリーに、新しい **network_data_spine_leaf.yaml** ファイルを作成します。デフォルトの **network_data** 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.yaml
/home/stack/network_data_spine_leaf.yaml
```

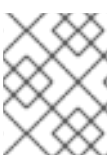
2. **network_data_spine_leaf.yaml** ファイルで、各ネットワークとリーフネットワークをコンポーザブルネットワーク項目として定義する YAML リストを作成します。たとえば、内部 API ネットワークとそのリーフネットワークは、以下の構文を使用して定義します。

```
# 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'}]
```



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。この例では、**InternalApi** にこれらのサービスが含まれています。

コンポーザブルネットワークの完全な例は、付録A *network_data ファイルの例*を参照してください。

4.2. ロールデータファイルの作成

本項では、各リーフ用のコンポーザブルロールを定義して、それぞれのロールにコンポーザブルネットワークを接続する方法について実例をあげて説明します。

手順

1. **stack** ユーザーのローカルディレクトリー内に、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 1 用に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/Compute1.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/CephStorage.yaml
~/roles/CephStorage1.yaml
```

3. **Compute1.yaml** ファイルを編集します。

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

4. このファイルで **name**、**networks**、および **HostnameFormatDefault** のパラメーターを編集して、Leaf 1 固有のパラメーターと一致するようにします。以下に例を示します。

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

このファイルを保存します。

5. **CephStorage1.yaml** ファイルを編集します。

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

6. このファイルで **name** および **networks** のパラメーターを編集して、Leaf 1 固有のパラメーターと一致するようにします。また、**HostnameFormatDefault** パラメーターを追加して、Ceph Storage ノード用の Leaf 1 のホスト名を定義します。以下に例を示します。

```
- name: CephStorage1
  ...
  networks:
```

```
    - Storage1
    - StorageMgmt1
  HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
```

このファイルを保存します。

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

```
$ 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 および Leaf 3 のファイルで **name**、**networks**、および **HostnameFormatDefault** のパラメーターを編集して、対応するリーフネットワークのパラメーターと一致するようにします。たとえば、Leaf 2 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: Compute2
  ...
  networks:
    - InternalApi2
    - Tenant2
    - Storage2
  HostnameFormatDefault: '%stackname%-compute2-%index%'
```

Leaf 2 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: CephStorage2
  ...
  networks:
    - Storage2
    - StorageMgmt2
  HostnameFormatDefault: '%stackname%-cephstorage2-%index%'
```

9. ロールの準備が整ったら、以下のコマンドで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 Compute1 Compute2 Compute3 CephStorage1 CephStorage2 CephStorage3
```

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。

このファイルの完全なサンプルは、付録C *roles_data ファイルの例*に記載しています。

ロールごとに独自の NIC 設定があります。スパイン/リーフ設定を設定する前には、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、独自の 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て、現在の NIC 設定に合わせて変更します。

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/リーフ型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** 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. 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
```

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イステンプレート を参照してください。

4.4. コントローラー用のカスタム NIC 設定の編集

レンダリングされたテンプレートには、スパイン/リーフ設定に対応するのに必要なほとんどのコンテンツが含まれます。ただし、追加の設定変更が必要になります。Leaf0 上のコントローラーノードの YAML 構造を変更するには、以下の手順に従います。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. **controller0.yaml** テンプレートを編集します。

3. **parameters** セクションの **ControlPlaneSubnetCidr** パラメーターおよび **ControlPlaneDefaultRoute** パラメーターまでスクロールします。これらのパラメーターは、以下のスニペットのようになります。

```
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
```

Leaf0 に応じてこれらのパラメーターを変更します。

```
  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. **parameters** セクションの **EC2MetadataIp** パラメーターまでスクロールします。このパラメーターは、以下のスニペットのようになります。

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

Leaf0 に応じてこれらのパラメーターを変更します。

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

5. ネットワーク設定のセクションまでスクロールします。このセクションの例を以下に示します。

```
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:
```

6. **network_config** セクションで、コントロールプレーンとプロビジョニングのインターフェイスを定義します。以下に例を示します。

```
            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
```

この例で使用されているパラメーターは Leaf0 固有の **ControlPlane0SubnetCidr**、**Leaf0EC2MetadataIp**、および **ControlPlane0DefaultRoute** です。ルートとして使用されているプロビジョニングネットワーク (192.168.10.0/24) 上の Leaf0 の CIDR の使用方法にも注目してください。

7. **members** セクションの各 VLAN には、関連する Leaf0 パラメーターが含まれます。たとえば、Storage ネットワークの VLAN 情報は、以下のスニペットのようになります。

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

ルーティングのパラメーターを定義するセクションを追加します。これには、supernet ルート (ここでは **StorageSupernet**) およびリーフのデフォルトルート (ここでは **Storage0InterfaceDefaultRoute**) が含まれます。

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

```
        - ip_netmask:
            get_param: StorageSupernet
          next_hop:
            get_param: Storage0InterfaceDefaultRoute
```

Storage、**StorageMgmt**、**InternalApi**、および **Tenant** コントローラーネットワーク用 VLAN 設定のルートを追加します。

8. このファイルを保存します。

4.5. コンピュート用のカスタム NIC 設定の作成

以下の手順では、Leaf0、Leaf1、および Leaf2 上のコンピュートノードの YAML 設定を作成します。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. **compute0.yaml** テンプレートを編集します。

3. **parameters** セクションの **ControlPlaneSubnetCidr** パラメーターおよび **ControlPlaneDefaultRoute** パラメーターまでスクロールします。これらのパラメーターは、以下のスニペットのようになります。

```
  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
```

Leaf0 に応じてこれらのパラメーターを変更します。

```
  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. **parameters** セクションの **EC2MetadataIp** パラメーターまでスクロールします。このパラメーターは、以下のスニペットのようになります。

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

Leaf0 に応じてこれらのパラメーターを変更します。

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

5. ネットワーク設定のセクションまでスクロールします。このセクションは、以下のスニペットのようになります。

```
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:
```

6. **network_config** セクションで、コントロールプレーンとプロビジョニングのインターフェイスを定義します。以下に例を示します。

```
              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
```

この例で使用されているパラメーターは Leaf0 固有の **ControlPlane0SubnetCidr**、**Leaf0EC2MetadataIp**、および **ControlPlane0DefaultRoute** です。ルートとして使用されているプロビジョニングネットワーク (192.168.10.0/24) 上の Leaf0 の CIDR の使用方法にも注目してください。

7. **members** セクションの各 VLAN には、関連する Leaf0 パラメーターが含まれている必要があります。たとえば、Storage ネットワークの VLAN 情報は、以下のスニペットのようになります。

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

ルーティングのパラメーターを定義するセクションを追加します。これには、supernet ルート (ここでは **StorageSupernet**) およびリーフのデフォルトルート (ここでは **Storage0InterfaceDefaultRoute**) が含まれます。

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

Storage、**InternalApi**、および **Tenant** コントローラーネットワークの VLAN 設定を追加します。

8. このファイルを保存します。

9. **compute1.yaml** を編集して、同じ手順を実施します。変更の一覧は以下のとおりです。

   - **ControlPlaneSubnetCidr** を **ControlPlane1SubnetCidr** に変更します。
   - **ControlPlaneDefaultRoute** を **ControlPlane1DefaultRoute** に変更します。
   - **EC2MetadataIp** を **Leaf1EC2MetadataIp** に変更します。
   - ネットワーク設定スクリプトを **../../scripts/run-os-net-config.sh** から **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh** に変更します。
   - Leaf1 パラメーターを使用するように、コントロールプレーンとプロビジョニングのインターフェイスを変更します。

- Leaf1 ルートが含まれるように各 VLAN を変更します。

編集が終わったらファイルを保存してください。

10. **compute2.yaml** を編集して、同じ手順を実施します。変更の一覧は以下のとおりです。
    - **ControlPlaneSubnetCidr** を **ControlPlane2SubnetCidr** に変更します。
    - **ControlPlaneDefaultRoute** を **ControlPlane2DefaultRoute** に変更します。
    - **EC2MetadataIp** を **Leaf2EC2MetadataIp** に変更します。
    - ネットワーク設定スクリプトを **../../scripts/run-os-net-config.sh** から **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh** に変更します。
    - Leaf2 パラメーターを使用するように、コントロールプレーンとプロビジョニングのインターフェイスを変更します。
    - Leaf2 ルートが含まれるように各 VLAN を変更します。

    編集が終わったらファイルを保存してください。

4.6. CEPH STORAGE 用のカスタム NIC 設定の作成

この手順では、Leaf0、Leaf1、および Leaf2 上の Ceph Storage ノードの YAML 設定を作成します。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. **parameters** セクションの **ControlPlaneSubnetCidr** パラメーターおよび **ControlPlaneDefaultRoute** パラメーターまでスクロールします。これらのパラメーターは、以下のスニペットのようになります。

```
  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
```

Leaf0 に応じてこれらのパラメーターを変更します。

```
  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. **parameters** セクションの **EC2MetadataIp** パラメーターまでスクロールします。このパラメーターは、以下のスニペットのようになります。

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

Leaf0 に応じてこれらのパラメーターを変更します。

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

4. ネットワーク設定のセクションまでスクロールします。このセクションは、以下のスニペットのようになります。

```
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:
```

5. **network_config** セクションで、コントロールプレーンとプロビジョニングのインターフェイスを定義します。以下に例を示します。

```
              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
```

この例で使用されているパラメーターは Leaf0 固有の **ControlPlane0SubnetCidr**、**Leaf0EC2MetadataIp**、および **ControlPlane0DefaultRoute** です。ルートとして使用されているプロビジョニングネットワーク (192.168.10.0/24) 上の Leaf0 の CIDR の使用方法にも注目してください。

6. **members** セクションの各 VLAN には、関連する Leaf0 パラメーターが含まれます。たとえば、Storage ネットワークの VLAN 情報は、以下のスニペットのようになります。

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

ルーティングのパラメーターを定義するセクションを追加します。これには、supernet ルート (ここでは **StorageSupernet**) およびリーフのデフォルトルート (ここでは **Storage0InterfaceDefaultRoute**) が含まれます。

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

Storage、**StorageMgmt** コントローラーネットワークの VLAN 設定を追加します。

7. このファイルを保存します。

8. **ceph-storage1.yaml** を編集して、同じ手順を実施します。変更の一覧は以下のとおりです。

   - **ControlPlaneSubnetCidr** を **ControlPlane1SubnetCidr** に変更します。
   - **ControlPlaneDefaultRoute** を **ControlPlane1DefaultRoute** に変更します。
   - **EC2MetadataIp** を **Leaf1EC2MetadataIp** に変更します。

- ネットワーク設定スクリプトを **../../scripts/run-os-net-config.sh** から **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh** に変更します。
- Leaf1 パラメーターを使用するように、コントロールプレーンとプロビジョニングのインターフェイスを変更します。
- Leaf1 ルートが含まれるように各 VLAN を変更します。

編集が終わったらファイルを保存してください。

9. **ceph-storage2.yaml** を編集して、同じ手順を実施します。変更の一覧は以下のとおりです。
   - **ControlPlaneSubnetCidr** を **ControlPlane2SubnetCidr** に変更します。
   - **ControlPlaneDefaultRoute** を **ControlPlane2DefaultRoute** に変更します。
   - **EC2MetadataIp** を **Leaf2EC2MetadataIp** に変更します。
   - ネットワーク設定スクリプトを **../../scripts/run-os-net-config.sh** から **/usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh** に変更します。
   - Leaf2 パラメーターを使用するように、コントロールプレーンとプロビジョニングのインターフェイスを変更します。
   - Leaf2 ルートが含まれるように各 VLAN を変更します。

   編集が終わったらファイルを保存してください。

4.7. ネットワーク環境ファイルの作成

この手順では、後で使用する基本的なネットワーク環境ファイルを作成します。

手順

1. stack ユーザーの **templates** ディレクトリーに **network-environment.yaml** ファイルを作成します。
2. 環境ファイルに以下のセクションを追加します。

```
resource_registry:

parameter_defaults:
```

   以下の点に注意してください。

   - **resource_registry** は、ネットワークリソースをそれぞれの NIC テンプレートにマッピングします。
   - **parameter_defaults** は、お使いの設定に関連した追加のネットワークパラメーターを定義します。

これ以降の数セクションでは、スパイン/リーフアーキテクチャーの特定機能を設定するために、ネットワーク環境ファイルに情報を追加します。この作業が完了したら、そのファイルを **openstack overcloud deploy** コマンドで指定します。

4.8. NIC テンプレートへのネットワークリソースのマッピング

この手順では、ネットワーク設定の関連するリソースをそれぞれの NIC テンプレートにマッピングします。

手順

1. **network-environment.yaml** ファイルを編集します。
2. **resource_registry** にリソースマッピングを追加します。リソース名には以下の形式を使用します。

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

本ガイドのシナリオでは、**resource_registry** に以下のリソースマッピングが含まれます。

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

4.9. スパイン/リーフのルーティング

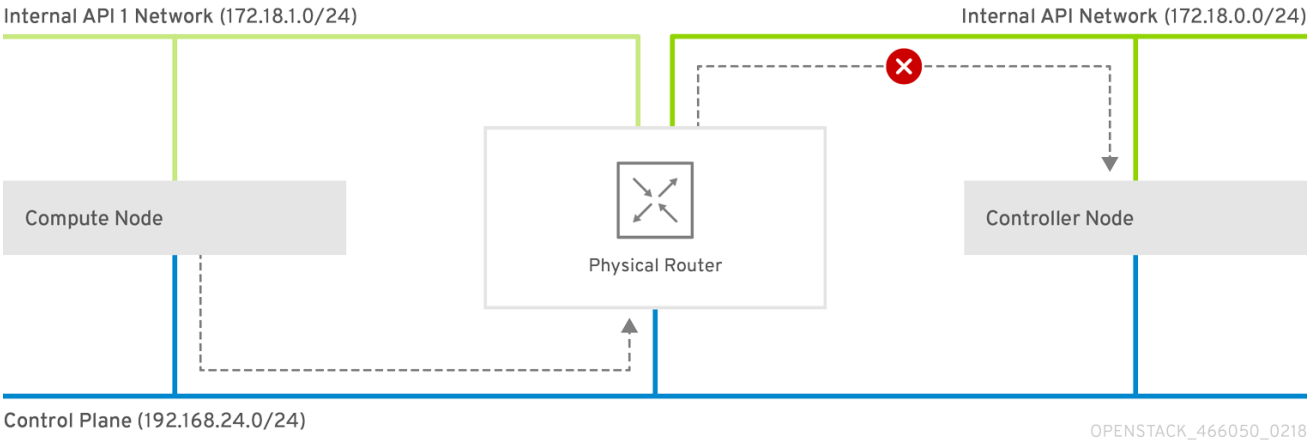
各ロールには、同じ機能によって使用される別のサブネットをポイントする各分離ネットワーク上のルートが必要です。**Compute1** ノードが **InternalApi** VIP 上のコントローラーにコンタクトする場合には、トラフィックは **InternalApi1** ゲートウェイを介した **InternalApi1** インターフェイスをターゲットにする必要があります。その結果、コントローラーから **InternalApi1** ネットワークに戻るトラフィックは、**InternalApi** ネットワークゲートウェイを経由するはずです。

supernet ルートは、各ロール上の全分離ネットワークに適用して、デフォルトのゲートウェイ経由でトラフィックが送信されるのを防ぎます。これは、デフォルトでは、コントローラー以外のロールの場合には **Control Plane** ネットワークで、コントローラー上の場合には **External** ネットワークです。

Red Hat Enterprise Linux は受信トラフィックに対して厳格な逆方向パスフィルターをデフォルトで実装するので、これらのルートを分離ネットワーク上で設定する必要があります。API が **Internal API** インターフェイス上でリッスンしている場合には、要求はその API で受信し、戻るパスのルートが **Internal API** インターフェイス上にある場合にのみ要求を受理します。サーバーが **Internal API** ネットワークをリッスンしているが、クライアントに戻るパスが **Control Plane** 経由の場合には、逆方向パスフィルターによりそのサーバーは要求を破棄します。

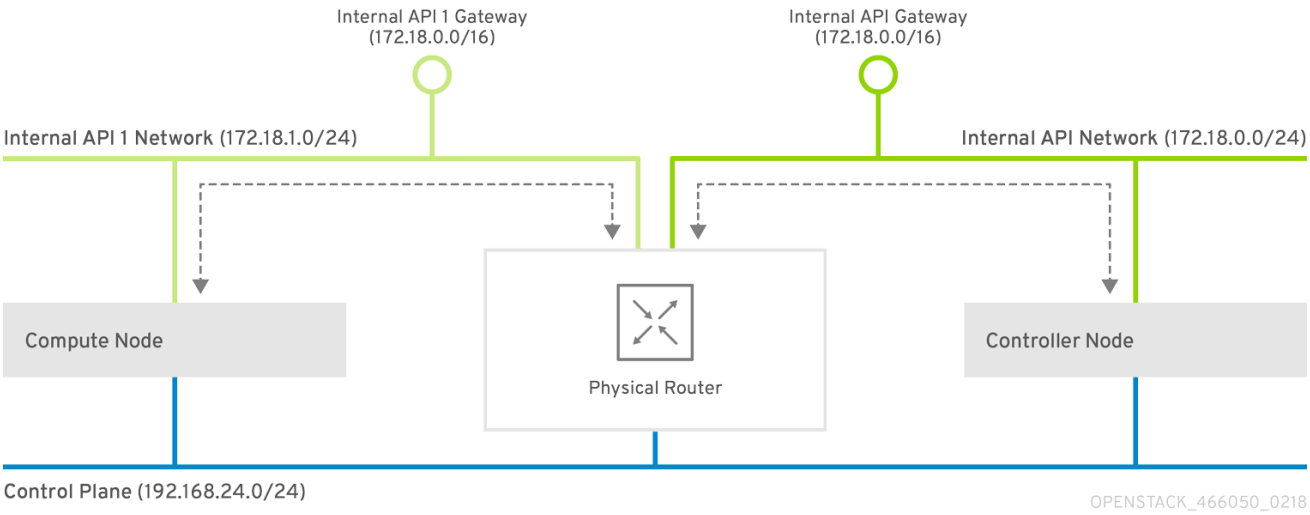
下図には、コントロールプレーンを経由してトラフィックのルーティングを試みて、成功しない例を示しています。ルーターからコントローラーノードに戻るルートは、VIP がリッスンしているインターフェイスとは一致しないので、パケットは破棄されます。**192.168.24.0/24** は直接コントローラーに接続され、**Control Plane** ネットワークに対してローカルであると見なされます。

図4.1 コントロールプレーン経由のトラフィックルーティング



比較のために、**Internal API** ネットワーク経由のルーティングを以下に示します。

図4.2 Internal API 経由のトラフィックルーティング



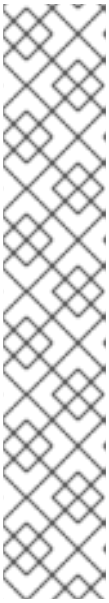
4.10. コンポーザブルネットワークのルート割り当て

この手順では、リーフネットワークのルーティングを定義します。

手順

1. **network-environment.yaml** ファイルを編集します。
2. **parameter_defaults** セクションに supernet ルートパラメーターを追加します。各分離ネットワークごとに supernet ルートを適用する必要があります。以下に例を示します。

```
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
```



注記

ネットワークインターフェイステンプレートには、各ネットワークの supernet パラメーターを指定する必要があります。以下に例を示します。

```
- 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. **parameter_defaults** セクションに **ServiceNetMap HostnameResolveNetwork** パラメーターを追加して、リーフの各ノードに他のリーフノードを解決するのに使用するホスト名の一覧を提供します。以下に例を示します。

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

コンピュートノードはリーフの Internal API ネットワークを使用し、Ceph Storage ノードはリーフの Storage ネットワークを使用します。

4. 以下の **ExtraConfig** 設定を **parameter_defaults** セクションに追加して、コンピュートノードおよび Ceph Storage ノード上の特定のコンポーネントのルーティングに対応します。

表4.1 コンピュートの ExtraConfig パラメーター

パラメーター	この値に設定します
nova::compute::libvirt::vncserver_listen	VNC サーバーがリッスンする IP アドレス
nova::compute::vncserver_proxyclient_address	VNC プロキシークライアントを実行しているサーバーの IP アドレス
neutron::agents::ml2::ovs::local_ip	OpenStack Networking (neutron) トンネルエンドポイントの IP アドレス
cold_migration_ssh_inbound_addr	コールドマイグレーション SSH 接続用のローカル IP アドレス

パラメーター	この値に設定します
live_migration_ssh_inbound_addr	ライブマイグレーション SSH 接続用のローカル IP アドレス
nova::migration::libvirt::live_migration_inbound_addr	ライブマイグレーションのトラフィックに使用される IP アドレス  **注記** SSL/TLS を使用している場合は、ネットワーク名の前に fqdn_を追加して、証明書が FQDN に対してチェックされるようにします。
nova::my_ip	ホスト上の Compute (nova) サービスの IP アドレス
tripleo::profile::base::database::mysql::client::mysql_client_bind_address	データベースクライアントの IP アドレス。この場合、コンピュートノード上の **mysql** クライアントになります。

表4.2 CephAnsibleExtraConfig パラメーター

パラメーター	この値に設定します
public_network	Ceph ノードが含まれるすべてのストレージネットワークのコンマ区切りの一覧 (リーフごとに 1 つ)。たとえば、**172.16.0.0/24,172.16.1.0/24,172.16.2.0/24**。
cluster_network	Ceph ノードが含まれるストレージ管理ネットワークのコンマ区切りの一覧 (リーフごとに 1 つ)。たとえば、**172.17.0.0/24,172.17.1.0/24,172.17.2.0/24**。

以下に例を示します。

```
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. 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。そのためには、いくつかの追加パラメーターを **spine-leaf-ctlplane.yaml** ファイルで定義する必要があります。たとえば、以下のスニペットは、Leaf0 上のコントローラーノード用の NIC テンプレートのサンプルの抜粋です。

```
- 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
```

この場合は、Leaf 0 上の Control Plane ネットワークに対応する IP、サブネット、メタデータ IP、デフォルトルートを定義する必要があります。

手順

1. **network-environment.yaml** ファイルを編集します。
2. **parameter_defaults** セクションを以下のように編集します。
   a. メインのコントロールプレーンのサブネットへのマッピングを追加します。

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

   b. 各スパイン/リーフ型ネットワーク用のコントロールプレーンのサブネットへのマッピングを追加します。

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

   c. 各リーフにコントロールプレーンのルートを追加します。

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

   デフォルトルートのパラメーターは通常、プロビジョニングサブネットの **gateway** に設定される IP アドレスです。この情報については、**undercloud.conf** ファイルを参照してください。

   d. EC2 メタデータ IP 用のパラメーターを追加します。

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

これらは、EC2 メタデータサービス (169.254.169.254/32) 用のコントロールプレーンを介したルートとして機能します。これは通常、プロビジョニングネットワーク上の各リーフの **gateway** に設定します。

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

4.12. スパイン/リーフ対応のオーバークラウドのデプロイ

デプロイメントに向けた全ファイルの準備が整いました。本項では、各ファイルのレビューとデプロイメントのコマンドについて説明します。

手順

1. **/home/stack/template/network_data_spine_leaf.yaml** をチェックして、各リーフ用のネットワークがすべて含まれていることを確認します。

注記

ネットワークサブネットと **allocation_pools** の値に対して実行される検証は現在ありません。これらの値が整合性を維持して定義され、既存ネットワークとの競合が発生しないことを確認してください。

2. **~/templates/spine-leaf-nics/** に含まれている NIC テンプレートをチェックして、各リーフ上の各ロールのインターフェイスが正しく定義されていることを確認します。

3. **network-environment.yaml** 環境ファイルをチェックして、ネットワークデータファイルでは制御できない全カスタムパラメーターが定義されていることを確認します。これには、ルート、コントロールプレーンのパラメーター、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ます。

4. **/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。

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、各リーフの全ノードが正しくタグ付けされていることも確認してください。

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/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** 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。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** およびその他のネットワークベースの環境ファイルの後に、**network-environment.yaml** ファイルを追加します。これにより、**network-**

environment.yaml 内に定義されたパラメーターおよびリソースが、他の環境ファイルで以前に定義された同じパラメーターおよびリソースをオーバーライドするようになります。

- 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。

付録A NETWORK_DATA ファイルの例

```
# 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'
```

付録B カスタムの NIC テンプレート

スパイン/リーフ型ネットワーク用のネットワークインターフェイステンプレートの設定を開始するためのテンプレートを以下に示します。**resources** のセクションは不完全で、お使いのインターフェイスの定義が必要である点に注意してください。

```
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
```

付録C ROLES_DATA ファイルの例

```
###############################################################################
# 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
```

付録D NETWORK_ENVIRONMENT ファイルの例

```
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')}"
```