# Red Hat OpenStack Platform 17.1

## スパイン/リーフ型ネットワークの設定

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: 2024-05-25

# Red Hat OpenStack Platform 17.1 スパイン/リーフ型ネットワークの設定

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

# 多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

# RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

### Jira でドキュメントのフィードバックを提供する

ドキュメントに関するフィードバックを提供するには、Create Issue フォームを使用します。Red Hat OpenStack Platform Jira プロジェクトで Jira Issue が作成され、フィードバックの進行状況を追跡できます。

1. Jira にログインしていることを確認してください。Jira アカウントをお持ちでない場合は、アカウントを作成してフィードバックを送信してください。
2. Create Issue をクリックして、**Create Issue** ページを開きます。
3. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
4. **Create** をクリックします。

# 第1章 スパイン/リーフ型ネットワーキングの概要

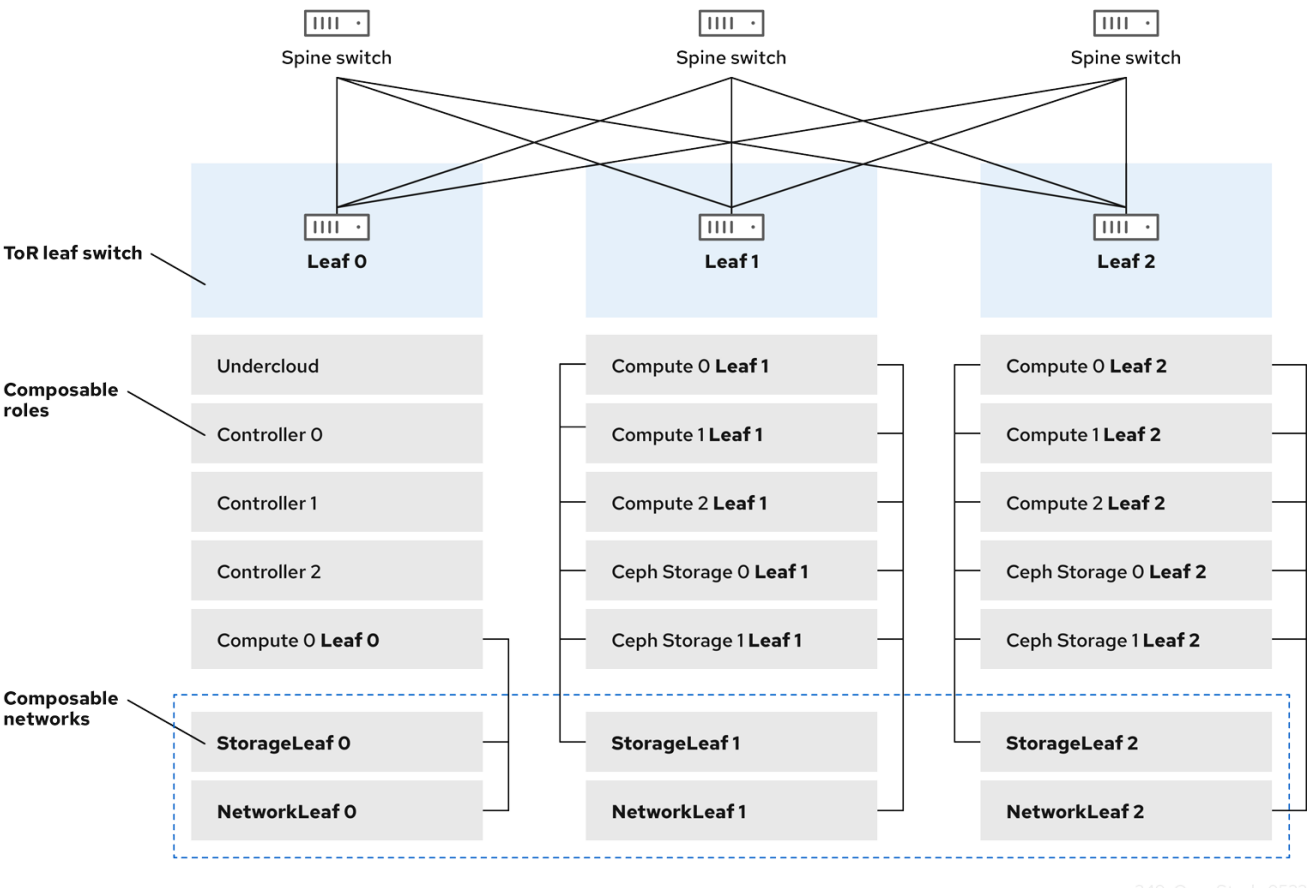
以下の章では、Red Hat OpenStack Platform 環境のスパイン/リーフ型ネットワークトポロジーの構築に関する情報を提供します。これには、完全なエンドツーエンドのシナリオと、お使いの環境でより広範囲なネットワークトポロジーを複製するのに役立つサンプルファイルが含まれます。

## 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/リーフ型トポロジーの例 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、以下の設定を示しています。

- **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、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. 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 |
|---|---|---|
| Provisioning / Ctlplane / 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 [1] | Controller、ComputeLeaf0 | 172.19.0.0/24 |
| External | Controller | 10.1.1.0/24 |

[1] テナントネットワークは、プロジェクトネットワークとしても知られています。

**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 |
|---|---|---|
| Provisioning / Ctlplane / Leaf1 | ComputeLeaf1、CephStorageLeaf1 | 192.168.11.0/24 |
| StorageLeaf1 | ComputeLeaf1、CephStorageLeaf1 | 172.16.1.0/24 |
| StorageMgmtLeaf1 | CephStorageLeaf1 | 172.17.1.0/24 |
| InternalApiLeaf1 | ComputeLeaf1 | 172.18.1.0/24 |
| TenantLeaf1 [1] | ComputeLeaf1 | 172.19.1.0/24 |

[1] テナントネットワークは、プロジェクトネットワークとしても知られています。

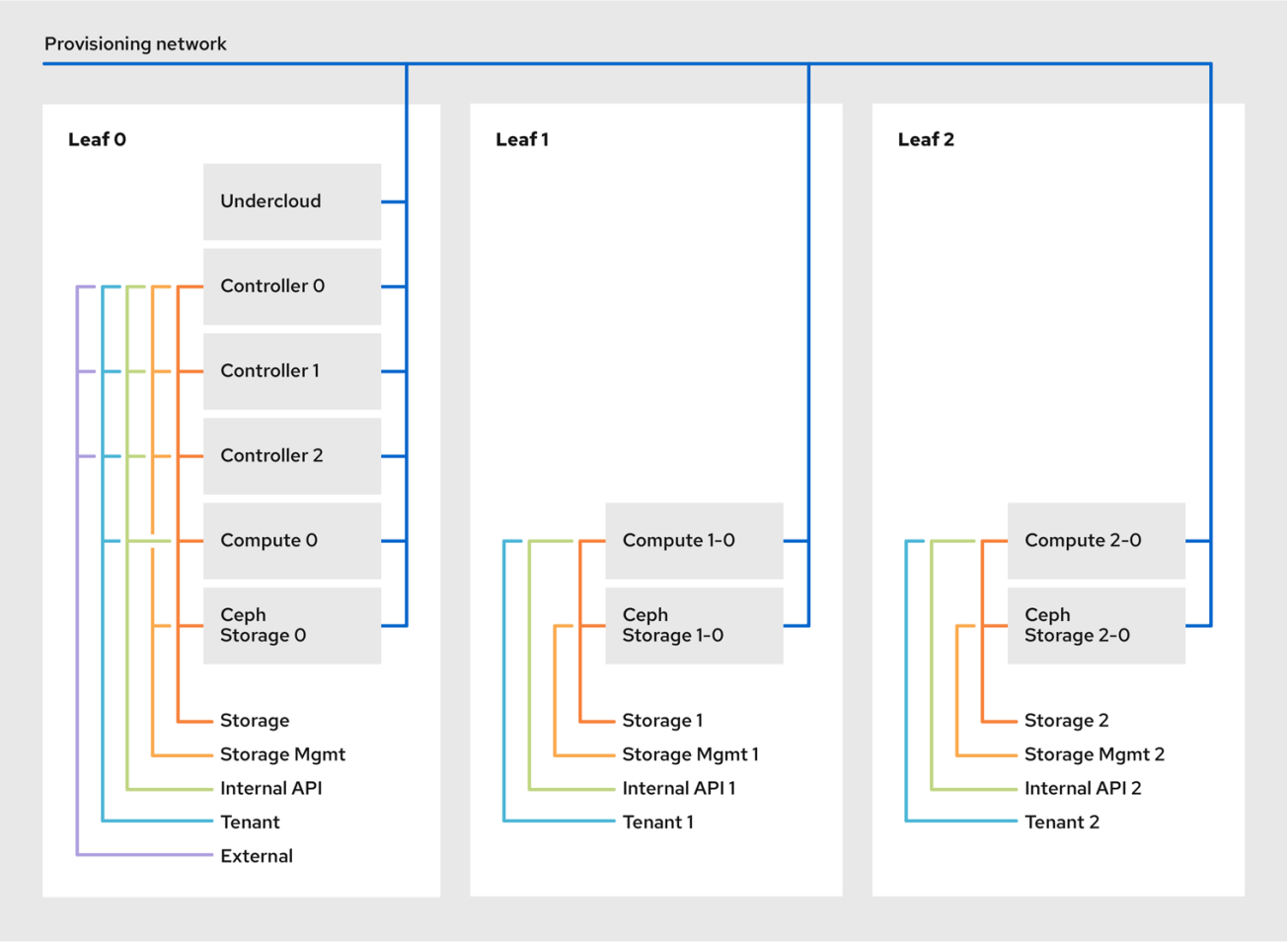
**表1.3 Leaf 2 ネットワーク**

| ネットワーク | アタッチされているロール | サブネット |
|---|---|---|
| Provisioning / Ctlplane / 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 [1] | ComputeLeaf2 | 172.19.2.0/24 |

[1] テナントネットワークは、プロジェクトネットワークとしても知られています。

**図1.2 スパイン/リーフ型ネットワークトポロジー**




249_OpenStack_0522

## 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**

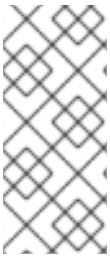
アンダークラウドは 2 つの DHCP サーバーを使用します。1 つは、ベアメタルノードのイントロスペクション用で、もう 1 つはオーバークラウドノードのデプロイ用です。**dhcp-relay** を設定する場合は、DHCP リレーの設定を読んで要件を理解するようにしてください。

## 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_host** 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_host = 10.1.1.1
```

   c. **undercloud_admin_host** 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_host = 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リストを定義します。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

DHCP リレーサービスは、リクエストを転送したいリモートネットワークセグメントに接続されているスイッチ、ルーター、またはサーバーで実行します。

**注記**

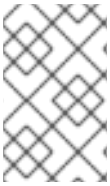
アンダークラウド上で DHCP リレーサービスを実行しないでください。

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 リレーは、アンダークラウド上の両方の DHCP サーバーに DHCP リクエストを転送するように設定する必要があります。

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。

**重要**

DHCP オプション 79 は、一部のリレー、特に DHCPv6 アドレスを提供するリレー、および元の MAC アドレスを渡さないリレーに必要です。詳細は、RFC6939 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、インターフェイスまたは 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され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 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イスは、**undercloud.conf** ファイルで **inspection_interface** として定義されるインターフェイスです。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イスは **br-ctlplane** になります。

**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** ファイルで **local_ip** として定義する IP アドレスは、**br-ctlplane** インターフェイス上にあります。

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。

- **dhcrelay** バージョン 4.2.x:

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

- **dhcrelay**バージョン 4.3.x 以降:

```
$ sudo dhcrelay -d --no-pid 192.168.10.10 192.168.10.1 \
  -iu eth0 -id eth1 -id eth2 -id 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. リーフノードのロールの指定

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。

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

2. ノードリストを取得して UUID を把握します。

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

3. リーフネットワークとロールを識別するカスタムリソースクラスを使用して、ロールに指定する各ベアメタルノードを割り当てます。

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

   - <ROLE> をロールを識別する名前に置き換えます。
   - <node> をベアメタルノードの ID に置き換えます。
     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。

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

4. 各ロールがまだ定義されていない場合は、**overcloud-baremetal-deploy.yaml** に追加します。

5. ロールのノードに割り当てるリソースクラスを定義します。

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

- <role> をロールの名前に置き換えます。
- <ROLE> をロールを識別する名前に置き換えます。

6. Baremetal-deploy.yaml ファイルで、ロールのノードに割り当てるリソースクラスを定義します。展開するロール、プロファイル、数量、および関連付けられているネットワークを指定します。

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

- <role> をロールの名前に置き換えます。
- <ROLE> をロールを識別する名前に置き換えます。

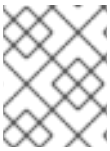
**注記**

**/home/stack/<stack>** に、デプロイするすべてのスタックに対して、**baremetal-deploy.yaml** 環境ファイルを作成する必要があります。

## 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>
```

6. オーバークラウドをデプロイする前に、ノードをイントロスペクションします。**--all-manageable** オプションと **-provide** オプションを付けて、デプロイ可能なノードとして設定します。

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

## 2.5. スパイン/リーフ型のプロビジョニングネットワークへの新規リーフの追加

新しい物理サイトの追加など、ネットワーク容量を増やす場合には、新しいリーフと、対応するサブネットを Red Hat OpenStack Platform のスパイン/リーフ型のプロビジョニングネットワークに追加する必要がある場合があります。オーバークラウドでリーフをプロビジョニングする場合には、対応するアンダークラウドのリーフが使用されます。

**前提条件**

- RHOSP デプロイメントでスパイン/リーフ型ネットワークトポロジーが使用されている。

**手順**

1. アンダークラウドホストに stack ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **/home/stack/undercloud.conf** ファイルで、以下の手順を実施します。
   a. **subnets** パラメーターを特定し、追加するリーフ用の新規サブネットを追加します。サブネットは、ルーティング対応のスパイン/リーフ内の L2 セグメントを表します。

      **例**

      以下の例では、新しいリーフ (**leaf3**) に新規サブネット (**leaf3**) が追加されます。

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

   b. 追加したサブネットのセクションを作成します。

      **例**

      以下の例では、新しいサブネット (**leaf3** に **[leaf3]** セクションが追加されます。

```
[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

[leaf3]
cidr = 192.168.13.0/24
dhcp_start = 192.168.13.10
```

```
dhcp_end = 192.168.13.90
inspection_iprange = 192.168.13.100,192.168.13.190
gateway = 192.168.13.1
masquerade = False
```

4. **undercloud.conf** ファイルを保存します。
5. アンダークラウドを再インストールします。

```
$ openstack undercloud install
```

**関連情報**

- スパイン/リーフ型デプロイメントへの新たなリーフの追加

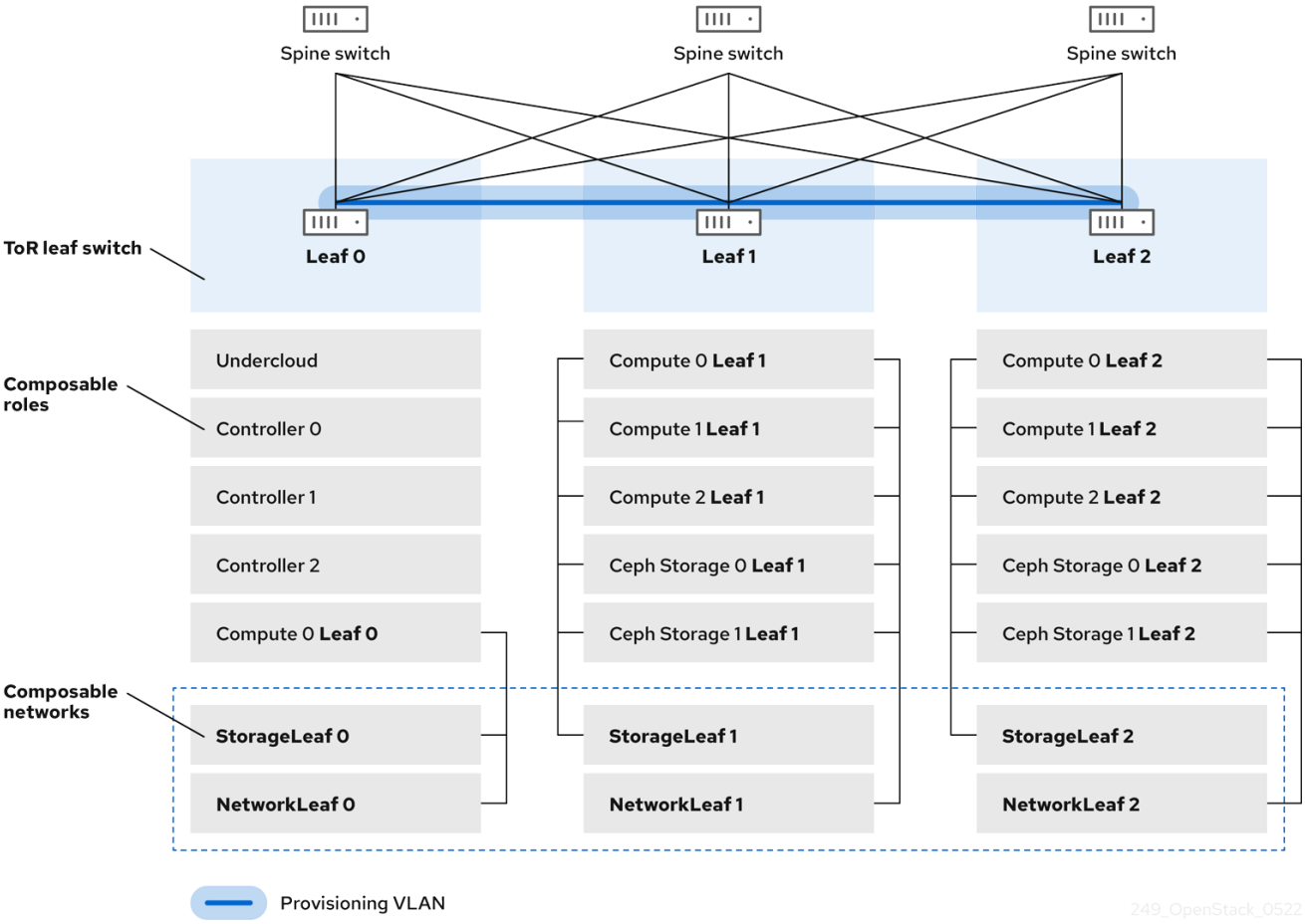
# 第3章 その他のプロビジョニングネットワーク設定手法

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、設定する任意のネットワークの例を示します。

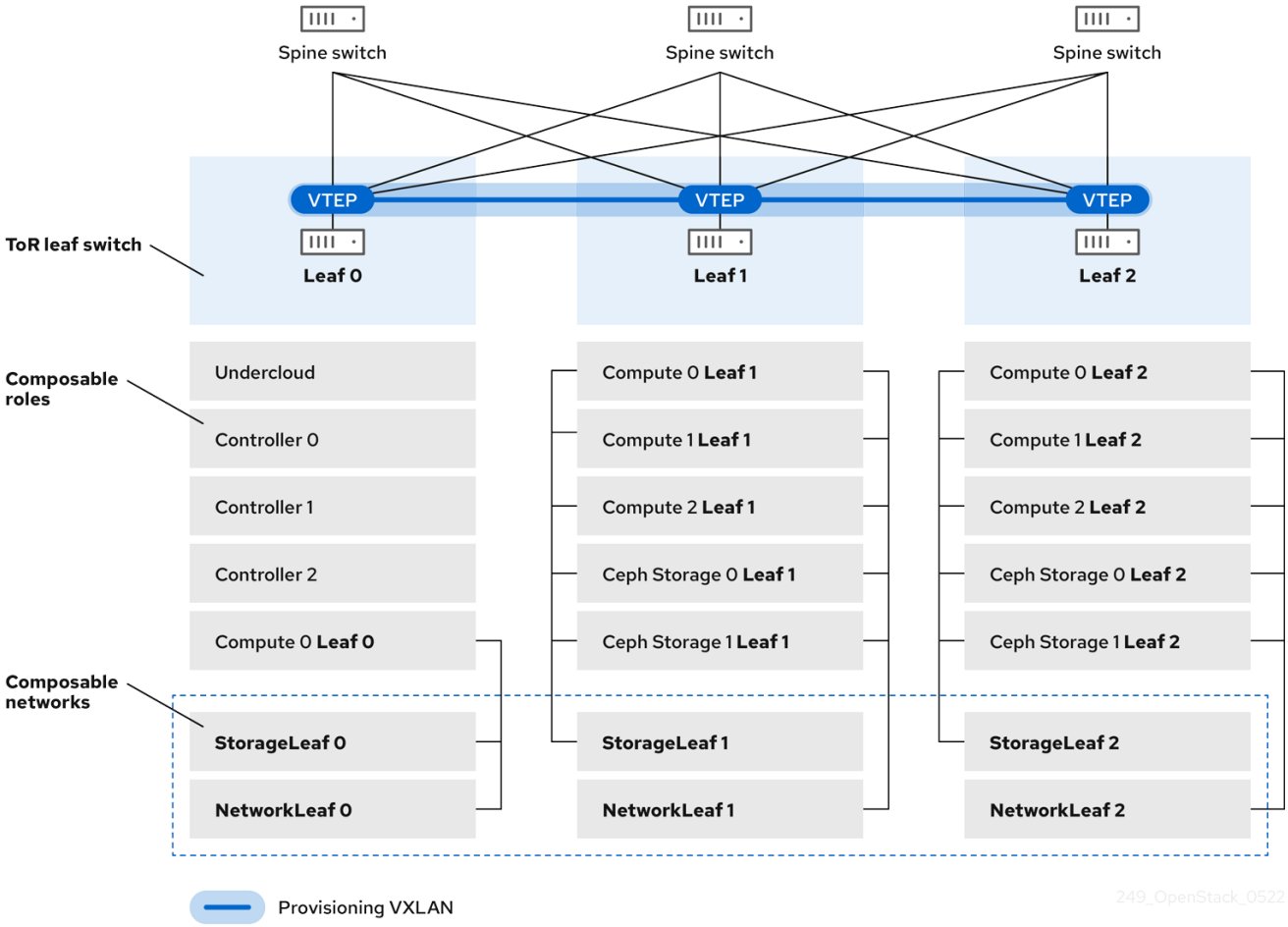
**図3.1 VLAN プロビジョニングネットワークトポロジー**




## 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。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

Red Hat OpenStack Platform (RHOSP) director を使用して、RHOSP オーバークラウドにスパイン/リーフネットワークをインストールおよび設定します。大まかな手順は次のとおりです。

1. リーフごとにオーバークラウドネットワークを定義します。
2. リーフごとにコンポーザブルロールを作成し、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します。
3. ロールごとに一意の NIC 設定を作成します。
4. 各リーフがそのリーフ上の特定のブリッジまたは VLAN を介してトラフィックをルーティングするように、ブリッジマッピングを変更します。
5. オーバークラウドエンドポイントの仮想 IP (VIP) を定義し、各仮想 IP のサブネットを特定します。
6. オーバークラウドネットワークとオーバークラウド仮想 IP をプロビジョニングします。
7. ベアメタルノードをオーバークラウドに登録します。

**注記**

事前にプロビジョニングされたベアメタルノードを使用している場合は、手順 7、8、および 9 をスキップします。

8. オーバークラウド内のベアメタルノードをイントロスペクトします。
9. ベアメタルノードをプロビジョニングします。
10. 前のステップで設定した設定を使用して、オーバークラウドをデプロイします。

## 4.1. リーフネットワークの定義

Red Hat OpenStack Platform (RHOSP) director は、作成した YAML 形式のカスタムネットワーク定義ファイルからオーバークラウドリーフネットワークを作成します。このカスタムネットワーク定義ファイルは、設定可能な各ネットワークとその属性をリストし、各リーフに必要なサブネットも定義します。

以下の手順を実行して、オーバークラウド上のスパイン/リーフネットワークの仕様を含む YAML 形式のカスタムネットワーク定義ファイルを作成します。その後、プロビジョニングプロセスにより、RHOSP オーバークラウドをデプロイするときに含めるネットワーク定義ファイルから heat 環境ファイルが作成されます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **/home/stack** の下に **templates** ディレクトリーを作成します。

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

4. デフォルトのネットワーク定義テンプレート、**routed-networks.yaml** をカスタムの **templates** ディレクトリーにコピーします。

**例**

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

5. ネットワーク定義テンプレートのコピーを編集して、各ベースネットワークおよび関連する各リーフサブネットを設定可能なネットワークアイテムとして定義します。

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの ネットワーク定義ファイル設定のオプション を参照してください。

**例**

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。

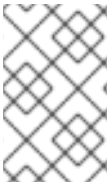
```
- name: InternalApi
  name_lower: internal_api
  vip: true
  mtu: 1500
  subnets:
    internal_api_subnet:
      ip_subnet: 172.16.32.0/24
      gateway_ip: 172.16.32.1
      allocation_pools:
        - start: 172.16.32.4
          end: 172.16.32.250
      vlan: 20
    internal_api_leaf1_subnet:
      ip_subnet: 172.16.33.0/24
      gateway_ip: 172.16.33.1
      allocation_pools:
        - start: 172.16.33.4
          end: 172.16.33.250
      vlan: 30
    internal_api_leaf2_subnet:
      ip_subnet: 172.16.34.0/24
      gateway_ip: 172.16.34.1
      allocation_pools:
        - start: 172.16.34.4
          end: 172.16.34.250
      vlan: 40
```

**注記**

コントロールプレーンネットワークは、アンダークラウドによってすでに作成されているため、カスタムネットワーク定義テンプレートでは定義しません。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。詳細は、アンダークラウドでのルーティング対応のスパイン/リーフの設定を参照してください。

**注記**

RHOSP は、ネットワークサブネットと **allocation_pools** の値の自動検証を実行しません。これらの値は、必ず一貫して定義し、既存のネットワークと競合しないようにしてください。



**注記**

コントローラーベースのサービスをホスティングするネットワークに対して、**vip** パラメーターを追加し、値を **true** に設定します。この例では、**InternalApi** ネットワークにこれらのサービスが含まれています。

**次のステップ**

1. 作成したカスタムネットワーク定義ファイルのパスとファイル名をメモします。この情報は、後で RHOSP オーバークラウド用にネットワークをプロビジョニングする際に必要になります。
2. 次のステップ リーフロールの定義とネットワークの接続 に進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの ネットワーク定義ファイル設定のオプション

## 4.2. リーフロールの定義とネットワークの接続

Red Hat OpenStack Platform (RHOSP) director は、リーフごとにコンポーザブルロールを作成し、作成したロールテンプレートからコンポーザブルネットワークをそれぞれのロールにアタッチします。まず、デフォルトの Controller、Compute、および Ceph Storage ロールを director コアテンプレートからコピーし、環境のニーズに合わせてこれらを変更します。個々のロールをすべて作成した後、**openstack overcloud role generated** コマンドを実行して、それらを 1 つの大きなカスタムロールデータファイルに連結します。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. RHOSP に同梱されている Controller、Compute、Ceph Storage ロールのデフォルトロールを **stack** ユーザーのホームディレクトリーにコピーします。ファイルがリーフ 0 であることを示すよう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/Controller0.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/Compute.yaml \
~/roles/Compute0.yaml
$ cp /usr/share/openstack-tripleo-heat-templates/roles/CephStorage.yaml \
~/roles/CephStorage0.yaml
```

4. リーフ 0 ファイルをコピーして、リーフ 1 およびリーフ 2 ファイルを作成します。

```
$ 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
```

5. 各ファイルのパラメーターを編集して、それぞれのリーフパラメーターに合わせます。

**ヒント**

ロールデータテンプレートのさまざまなパラメーターの詳細は、**Red Hat OpenStack Platform デプロイメントのカスタマイズ** ガイドの ロールパラメーターの検査 を参照してください。

**例 - ComputeLeaf0**

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

**例 - CephStorageLeaf0**

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

6. 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、リーフ 1 および リーフ 2 ファイルの **network** パラメーターを編集します。

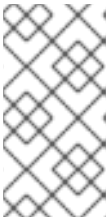
**例 - ComputeLeaf1**

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

**例 - CephStorageLeaf1**

```
- name: CephStorageLeaf1
```

```
networks:
  Storage:
    subnet: storage_leaf1
  StorageMgmt:
    subnet: storage_mgmt_leaf1
```



**注記**

この設定を行うのは、リーフ 1 および リーフ 2 だけです。リーフ 0 の **network** 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、リーフ 0 の内部 API は **internal_api_subnet** です。

7. コントローラー、コンピュート、および Networker (存在する場合) の各ロールファイルで、**ServicesDefault** パラメーターの下にあるサービスのリストに OVN BGP エージェントを追加します。

   **例**

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

8. ロールの設定が完了したら、**overcloud roles generate** 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。

   **例**

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

   これにより、それぞれのリーフネットワークのすべてのカスタムロールを含む 1 つのカスタムロールデータファイルが作成されます。

**次のステップ**

1. **overcloudrolesgenerate** コマンドによって作成されたカスタムロールデータファイルのパスとファイル名をメモします。このパスは、後でオーバークラウドをデプロイするときに使用します。
2. 次のステップ リーフロール用のカスタム NIC 設定の作成 に進みます。

**関連情報**

- Red Hat OpenStack Platform **デプロイメントのカスタマイズ** ガイドの ロールパラメーターの検討

## 4.3. リーフロール用のカスタム NIC 設定の作成

Red Hat OpenStack Platform (RHOSP) director が作成する各ロールには、固有の NIC 設定が必要です。次の手順を実行して、NIC テンプレートのカスタムセットと、カスタムテンプレートをそれぞれのロールにマッピングするカスタム環境ファイルを作成します。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。
- カスタムネットワーク定義ファイルがある。
- カスタムロールデータファイルがある。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. デフォルトの NIC テンプレートの 1 つからコンテンツをコピーして、NIC 設定のカスタムテンプレートを作成します。

**例**

この例では、NIC テンプレート **single-nic-vlans** をコピーし、NIC 設定のカスタムテンプレートとして使用します。

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

4. 前の手順で作成した各 NIC テンプレートで、スパイン/リーフトポロジーの詳細に一致するように NIC 設定を変更します。

**例**

```
{% set mtu_list = [ctlplane_mtu] %}
{% for network in role_networks %}
{{ mtu_list.append(lookup('vars', networks_lower[network] ~ '_mtu')) }}
{%- endfor %}
{% set min_viable_mtu = mtu_list | max %}
network_config:
- type: ovs_bridge
  name: {{ neutron_physical_bridge_name }}
  mtu: {{ min_viable_mtu }}
  use_dhcp: false
  dns_servers: {{ ctlplane_dns_nameservers }}
  domain: {{ dns_search_domains }}
  addresses:
  - ip_netmask: {{ ctlplane_ip }}/{{ ctlplane_subnet_cidr }}
  routes: {{ ctlplane_host_routes }}
  members:
  - type: interface
    name: nic1
    mtu: {{ min_viable_mtu }}
```

```
    # force the MAC address of the bridge to this interface
    primary: true
{% for network in role_networks %}
  - type: vlan
    mtu: {{ lookup('vars', networks_lower[network] ~ '_mtu') }}
    vlan_id: {{ lookup('vars', networks_lower[network] ~ '_vlan_id') }}
    addresses:
    - ip_netmask:
        {{ lookup('vars', networks_lower[network] ~ '_ip') }}/{{ lookup('vars',
networks_lower[network] ~ '_cidr') }}
    routes: {{ lookup('vars', networks_lower[network] ~ '_host_routes') }}
{% endfor %}
```

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの カスタムネットワークインターフェイステンプレートの定義 を参照してください。

5. カスタム NIC テンプレートを各カスタムロールにマッピングする **parameter_defaults** セクションを含む、**spine-leaf-nic-roles-map.yaml** などのカスタム環境ファイルを作成します。

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

**例**

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

**次のステップ**

1. カスタム NIC テンプレートのパスとファイル名、およびカスタム NIC テンプレートを各カスタムロールにマッピングするカスタム環境ファイルをメモします。このパスは、後でオーバークラウドをデプロイするときに使用します。
2. 次の リーフネットワークの設定 ステップに進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの カスタムネットワークインターフェイステンプレートの定義

## 4.4. リーフネットワークの設定

スパイン/リーフ型アーキテクチャーでは、各リーフは、そのリーフ上の特定のブリッジまたは VLAN を介してトラフィックをルーティングします。これは、エッジコンピューティングシナリオでよくあるケースです。そのため、Red Hat OpenStack Platform (RHOSP) コントローラーとコンピュートのネットワーク設定が **br-ex** ブリッジを使用するデフォルトのマッピングを変更する必要があります。

RHOSP director は、アンダークラウドの作成中にコントロールプレーンネットワークを作成します。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、デプロイメントに追加パラメーターを定義する必要があります。

以下の手順を実行して、個別のネットワークマッピングを含み、オーバークラウドのコントロールプレーンネットワークへのアクセスを設定するカスタムネットワーク環境ファイルを作成します。

**前提条件**

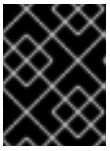
- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **spin-leaf-ctlplane.yaml** などの新しいカスタム環境ファイルで、**parameter_defaults** セクションを作成し、デフォルトの **br-ex** ブリッジを使用するリーフごとに **NeutronBridgeMappings** パラメーターを設定します。



**重要**

ネットワーク定義を含めるために作成するカスタム環境ファイルの名前は、**.yaml** または **.template** で終わる必要があります。

- 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。

**例**

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

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
```

```
  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"

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

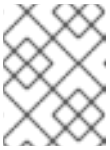
**ヒント**

詳細は、第 17 章ネットワーク (neutron) パラメーター (**オーバークラウドのパラメーター** ガイド) を参照してください。

- VLAN ネットワークマッピングの場合、**vlan** を **NeutronNetworkType** に追加し、**NeutronNetworkVLANRanges** を使用してリーフネットワークの VLAN をマッピングします。

  **例**

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

  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"

  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"

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



**注記**

スパイン/リーフトポロジーでは、フラットネットワークと VLAN の両方を使用できます。

4. **<role>ControlPlaneSubnet** パラメーターを使用して、各スパイン/リーフネットワークのコントロールプレーンサブネットマッピングを追加します。

   **例**

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    ControllerControlPlaneSubnet: leaf0
  Controller1Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Controller1ControlPlaneSubnet: leaf0
  Controller2Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Controller2ControlPlaneSubnet: leaf0
  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
    Compute0ControlPlaneSubnet: leaf0
  CephStorage0Parameters:
    CephStorage0ControlPlaneSubnet: leaf0
  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"
    Compute1ControlPlaneSubnet: leaf1
  CephStorage1Parameters:
    CephStorage1ControlPlaneSubnet: leaf1
  Compute2Parameters:
    NeutronBridgeMappings: "leaf2:br-ex"
    Compute2ControlPlaneSubnet: leaf2
  CephStorage2Parameters:
    CephStorage2ControlPlaneSubnet: leaf2
```

**次のステップ**

1. 作成したカスタムネットワーク環境ファイルのパスとファイル名をメモします。この情報は、後でオーバークラウドをデプロイするときに必要になります。
2. 次のステップ 仮想 IP アドレス用サブネットの設定 に進みます。

**関連情報**

- 第 17 章ネットワーク (neutron) パラメーター (**オーバークラウドのパラメーター** ガイド) を参照してください。

## 4.5. 仮想 IP アドレス用サブネットの設定

デフォルトでは、Red Hat Openstack Platform (RHOSP) コントローラーのロールは、各ネットワークの仮想 IP (VIP) アドレスをホストします。RHOSP オーバークラウドは、コントロールプレーンを除く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。

このドキュメントで使用されているスパイン/リーフの例では、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、値ペアの **subnet: leaf0** を **network:ctlplane** パラメーターに追加して、サブネットを **leaf0** にマップする必要があります。

以下の手順を実行して、オーバークラウド上の仮想 IP のオーバーライドを含む YAML 形式のカスタムネットワーク仮想 IP 定義ファイルを作成します。その後、プロビジョニングプロセスにより、RHOSP オーバークラウドをデプロイするときに含めるネットワーク仮想 IP 定義ファイルから heat 環境ファイルが作成されます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **spin-leaf-vip-data.yaml** などの新しいカスタムネットワーク仮想 IP 定義テンプレートで、コントローラーノードが使用する特定のサブネット上に作成する必要のある仮想 IP アドレスを一覧表示します。

**例**

```
- network: storage_mgmt
  subnet: storage_mgmt_subnet_leaf1
- network: internal_api
  subnet: internal_api_subnet_leaf1
- network: storage
  subnet: storage_subnet_leaf1
- network: external
  subnet: external_subnet_leaf1
  ip_address: 172.20.11.50
- network: ctlplane
  subnet: leaf0
- network: oc_provisioning
  subnet: oc_provisioning_subnet_leaf1
- network: storage_nfs
  subnet: storage_nfs_subnet_leaf1
```

**spine-leaf-vip-data.yaml** ファイルで、以下のパラメーターを使用できます。

**network**

neutron ネットワーク名を設定します。これは唯一の必須パラメーターです。

**ip_address**

VIP の IP アドレスを設定します。

**subnet**

neutron サブネット名を設定します。仮想 IP neutron ポートを作成するときにサブネットを指定するために使用します。このパラメーターは、展開でルーティングされたネットワークを使用する場合に必要です。

**dns_name**

FQDN (完全修飾ドメイン名) を設定します

**name**

仮想 IP 名を設定します。

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの コンポーザブルネットワークの追加 を参照してください。

**次のステップ**

1. 作成したカスタムネットワーク仮想 IP 定義テンプレートのパスとファイル名をメモします。このパスは、後で RHOSP オーバークラウド用にネットワーク仮想 IP をプロビジョニングするときに使用します。
2. 次のステップ オーバークラウド用のネットワークと仮想 IP のプロビジョニング に進みます。

## 4.6. オーバークラウド用のネットワークと仮想 IP のプロビジョニング

Red Hat OpenStack Platform (RHOSP) のプロビジョニングプロセスでは、ネットワーク定義ファイルを使用して、ネットワーク仕様を含む新しい heat 環境ファイルを作成します。デプロイメントで仮想 IP を使用する場合、RHOSP は仮想 IP 定義ファイルから新しい heat 環境ファイルを作成します。ネットワークと仮想 IP をプロビジョニングすると、後でオーバークラウドをデプロイするために使用する 2 つの heat 環境ファイルが作成されます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。
- ネットワーク設定テンプレートがある。
- 仮想 IP を使用している場合は、仮想 IP 定義テンプレートがある。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. オーバークラウドネットワークをプロビジョニングします。
**overcloud network professional** コマンドを使用して、前に作成したネットワーク定義ファイルへのパスを指定します。

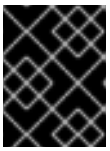
**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのネットワーク定義の設定とプロビジョニング を参照してください。

**例**

この例では、パスは **/home/stack/templates/spine-leaf-networks-data.yaml** です。**--output** 引数を使用して、コマンドによって作成されたファイルに名前を付けます。

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



**重要**

指定する出力ファイルの名前は、**.yaml** または **.template** で終わる必要があります。

4. オーバークラウド仮想 IP をプロビジョニングします。
**overcloud network vip professional** コマンドを **--stack** 引数とともに使用して、前に作成した仮想 IP 定義ファイルに名前を付けます。**--output** 引数を使用して、コマンドによって作成されたファイルに名前を付けます。

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのネットワーク仮想 IP の設定とプロビジョニング を参照してください。

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

**重要**

指定する出力ファイルの名前は、**.yaml** または **.template** で終わる必要があります。

5. 生成された出力ファイルのパスとファイル名に注意してください。この情報は、後でオーバークラウドをデプロイするときに使用します。

**検証**

- 以下のコマンドを使用して、コマンドによってオーバークラウドのネットワークとサブネットが作成されたことを確認できます。

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

<network>、<subnet>、<port> を、確認するネットワーク、サブネット、ポートの名前または UUID に置き換えます。

**次のステップ**

1. 事前にプロビジョニングされたノードを使用している場合は、オーバークラウドデプロイメントコマンドの実行 に進んでください。

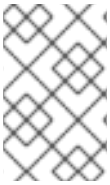
2. それ以外の場合は、次のステップ オーバークラウドへのベアメタルノードの登録 に進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのネットワーク定義の設定とプロビジョニング
- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのネットワーク仮想 IP の設定とプロビジョニング
- **コマンドラインインターフェイスリファレンス** の overcloud network provision
- **コマンドラインインターフェイスリファレンス** の overcloud network vip provision

## 4.7. オーバークラウドへのベアメタルノードの登録

Red Hat OpenStack Platform (RHOSP) director には、物理マシンのハードウェアおよび電源管理の詳細を指定するカスタムノード定義テンプレートが必要です。このテンプレートは、JSON または YAML 形式で作成できます。物理マシンをベアメタルノードとして登録したら、それらをイントロスペクトし、最後にプロビジョニングします。



**注記**

事前にプロビジョニングされたベアメタルノードを使用している場合は、ベアメタルノードの登録、イントロスペクト、およびプロビジョニングをスキップして、スパイン/リーフ対応のオーバークラウドのデプロイ に進むことができます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. 新しいノード定義テンプレート (**baremetal-nodes.yaml** など) を作成します。ハードウェアと電源管理の詳細を含む物理マシンのリストを追加します。

**例**

```
nodes:
  - name: "node01"
    ports:
      - address: "aa:aa:aa:aa:aa:aa"
        physical_network: ctlplane
        local_link_connection:
          switch_id: 52:54:00:00:00:00
          port_id: p0
    cpu: 4
    memory: 6144
```

```
    disk: 40
    arch: "x86_64"
    pm_type: "ipmi"
    pm_user: "admin"
    pm_password: "p@55w0rd!"
    pm_addr: "192.168.24.205"
  - name: "node02"
    ports:
      - address: "bb:bb:bb:bb:bb:bb"
        physical_network: ctlplane
        local_link_connection:
          switch_id: 52:54:00:00:00:00
          port_id: p0
    cpu: 4
    memory: 6144
    disk: 40
    arch: "x86_64"
    pm_type: "ipmi"
    pm_user: "admin"
    pm_password: "p@55w0rd!"
    pm_addr: "192.168.24.206"
```

**ヒント**

テンプレートパラメーター値と JSON の例の詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理** ガイドの オーバークラウドのノードの登録 を参照してください。

4. テンプレートのフォーマットと構文を確認します。

   **例**

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

5. エラーを修正し、ノード定義テンプレートを保存します。

6. ノード定義テンプレートを RHOSP director にインポートして、各ノードをテンプレートから director に登録します。

   **例**

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

**検証**

- ノードの登録と設定が完了したら、director がノードを正常に登録したことを確認します。

  ```
  $ openstack baremetal node list
  ```

  **baremetal node list** コマンドにはインポートされたノードが含まれており、ステータスが **manageable** である必要があります。

**次のステップ**

- 次のステップ オーバークラウド上のベアメタルノードのイントロスペクション に進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのノードの登録
- **コマンドラインインターフェイスリファレンス** の overcloud node import

## 4.8. オーバークラウド上のベアメタルノードのイントロスペクション

物理マシンをベアメタルノードとして登録した後、OpenStack Platform (RHOSP) director のイントロスペクションを使用して、ノードのハードウェア詳細を自動的に追加し、各イーサネット MAC アドレスのポートを作成できます。ベアメタルノードでイントロスペクションを実行したら、最後の手順はそれらをプロビジョニングすることです。

**注記**

事前にプロビジョニングされたベアメタルノードを使用している場合は、ベアメタルノードのイントロスペクトとイントロスペクトをスキップして、スパイン/リーフ対応オーバークラウドのデプロイ に進むことができます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。
- RHOSP でオーバークラウドのベアメタルノードを登録しました。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

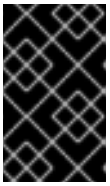
3. pre-introspection 検証グループを実行して、プリイントロスペクションの要件を確認します。

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

4. 検証レポートの結果を確認します。
5. オプション: 特定の検証からの詳細な出力を確認します。

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

<UUID> は、確認するレポートの特定の検証の UUID に置き換えます。



**重要**

検証結果が **FAILED** であっても、RHOSP のデプロイや実行が妨げられることはありません。ただし、**FAILED** の検証結果は、実稼働環境で問題が発生する可能性があることを意味します。

6. すべてのノードのハードウェア属性を検査します。

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

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理ガイド**の director のイントロスペクションを使用したベアメタルノードのハードウェア情報の収集 を参照してください。

別のターミナルウィンドウで、イントロスペクションの進捗ログを監視します。

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

**検証**

- イントロスペクション完了後には、すべてのノードが available の状態に変わります。

**次のステップ**

- 次のステップ オーバークラウドのベアメタルノードのプロビジョニング に進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理ガイド**の director のイントロスペクションを使用したベアメタルノードのハードウェア情報の収集
- **コマンドラインインターフェイスリファレンス** の overcloud node introspect

## 4.9. オーバークラウドのベアメタルノードのプロビジョニング

Red Hat OpenStack Platform (RHOSP) のベアメタルノードをプロビジョニングするには、デプロイするベアメタルノードの数と属性を定義し、これらのノードにオーバークラウドのロールを割り当てます。ノードのネットワークレイアウトも定義します。これらすべての情報を、YAML 形式のノード定義ファイルに追加します。

プロビジョニングプロセスにより、ノード定義ファイルから heat 環境ファイルが作成されます。この heat 環境ファイルには、ノード数、予測ノード配置、カスタムイメージ、カスタム NIC など、ノード定義ファイルで設定したノード仕様が含まれています。オーバークラウドをデプロイするときに、デプロイメントコマンドにこの heat 環境ファイルを含めます。プロビジョニングプロセスでは、ノード定義ファイル内の各ノードまたはロールに対して定義されたすべてのネットワークのポートリソースもプロビジョニングされます。

**注記**

事前にプロビジョニングされたベアメタルノードを使用している場合は、ベアメタルノードのプロビジョニングをスキップして、スパイン/リーフ対応オーバークラウドのデプロイ に進むことができます。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。
- ベアメタルノードは登録とイントロスペクトが行われ、プロビジョニングとデプロイメントに使用できます。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **spin-leaf-baremetal-nodes.yaml** などのベアメタルノード定義ファイルを作成し、プロビジョニングするロールごとにノード数を定義します。

**例**

```
- name: Controller
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
      default_route_network:
      - external
- name: Compute0
  count: 1
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
```

```
      subnet: tenant_subnet02
    - network: storage
      subnet: storage_subnet02
    network_config:
      template: /home/stack/templates/spine-leaf-nics/single-nic-vlans.j2
- name: Compute1
...
```

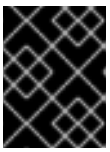
**ヒント**

ベアメタルノード定義ファイルを設定できるプロパティーの詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理** ガイドの オーバークラウドのベアメタルノードのプロビジョニング を参照してください。

4. **overcloud node provision** コマンドを使用して、オーバークラウドのベアメタルノードをプロビジョニングします。

**例**

```
$ openstack overcloud node provision \
 --stack spine_leaf_overcloud \
 --network-config \
 --output spine-leaf-baremetal-nodes-provisioned.yaml \
 /home/stack/templates/spine-leaf-baremetal-nodes.yaml
```



**重要**

指定する出力ファイルの名前は、**.yaml** または **.template** で終わる必要があります。

5. 別のターミナルでプロビジョニングの進捗をモニタリングします。プロビジョニングが成功すると、ノードの状態が **available** から **active** に変わります。

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

6. **metalsmith** ツールを使用して、割り当てやポートなどを含むノードの統合ビューを取得します。

```
$ metalsmith list
```

7. 生成された出力ファイルのパスとファイル名に注意してください。このパスは、後でオーバークラウドをデプロイするときに必要になります。

**検証**

- ノードとホスト名の関連付けを確認します。

```
$ openstack baremetal allocation list
```

**次のステップ**

- 次のステップ スパイン/リーフ対応オーバークラウドのデプロイ に進みます。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのベアメタルノードのプロビジョニング

## 4.10. スパイン/リーフ対応のオーバークラウドのデプロイ

Red Hat OpenStack Platform (RHOSP) オーバークラウドをデプロイする最後のステップは、**overcloud deploy** コマンドを実行することです。このコマンドに、作成したさまざまなオーバークラウドテンプレートと環境ファイルをすべて入力します。RHOSP director は、オーバークラウドのインストールと設定方法の計画としてこれらのテンプレートとファイルを使用します。

**前提条件**

- アンダークラウドホストへのアクセスと **stack** ユーザーの認証情報。
- このセクションの前の手順にリストされているすべてのステップを実行し、**overcloud deploy** コマンドの入力として使用するさまざまな heat テンプレートおよび環境ファイルをすべてアセンブルしました。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. オーバークラウド環境に必要なカスタム環境ファイルとカスタムテンプレートを照合します。このリストには、director インストールで提供される未編集の heat テンプレートファイルと、作成したカスタムファイルが含まれます。次のファイルへのパスがあることを確認します。
   - オーバークラウド上のスパイン/リーフネットワークの仕様を含むカスタムネットワーク定義ファイル (例: **spin-leaf-networks-data.yaml**)。
     詳細は、リーフネットワークの定義 を参照してください。
   - 各リーフのロールを定義するカスタムロールデータファイル。
     例: **spine-leaf-roles.yaml**

     詳細は、リーフのロールの定義とネットワークの接続 を参照してください。
   - ロールと各ロールのカスタム NIC テンプレートマッピングを含むカスタム環境ファイル。
     例: **spine-leaf-nic-roles-map.yaml**

     詳細は、リーフロール用のカスタム NIC 設定の作成 を参照してください。
   - 個別のネットワークマッピングを含み、オーバークラウドのコントロールプレーンネットワークへのアクセスを設定するカスタムネットワーク環境ファイル。
     例: **spine-leaf-ctlplane.yaml**

     詳細は、リーフネットワークの設定 を参照してください。
   - オーバークラウドネットワークのプロビジョニングからの出力ファイル。
     例: **spine-leaf-networks-provisioned.yaml**

詳細は、オーバークラウドのネットワークと仮想 IP のプロビジョニング を参照してください。

- オーバークラウド仮想 IP のプロビジョニングからの出力ファイル。
  例: **spine-leaf-vips-provisioned.yaml**

  詳細は、オーバークラウドのネットワークと仮想 IP のプロビジョニング を参照してください。

- 事前にプロビジョニングされたノードを使用していない場合は、ベアメタルノードのプロビジョニングからの出力ファイル。
  例: **spine-leaf-baremetal-nodes-provisioned.yaml**

  詳細は、オーバークラウド用のベアメタルノードのプロビジョニング を参照してください。

- その他のカスタム環境ファイル

4. コマンドへの入力であるカスタム環境ファイルとカスタムテンプレートを慎重に並べ替えて、**overcloud deploy** コマンドを入力します。
   一般的なルールは、未編集の heat テンプレートファイルを最初に指定し、次にカスタム環境ファイルと、デフォルトプロパティーのオーバーライドなどのカスタム設定を含むカスタムテンプレートを指定することです。

   **overclouddeploy** コマンドへの入力をリストする際には、次の順序に従います。

   a. 各ロールにマップされたカスタム NIC テンプレートを含むカスタム環境ファイルを含めます。
      例: **network-environment.yaml** の後に、**spine-leaf-nic-roles-map.yaml を追加します。**

      **network-environment.yaml** ファイルは、設定可能なネットワークパラメーターのデフォルトのネットワーク設定を提供します。これは、マッピングファイルによってオーバーライドされます。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。

   b. 他のスパイン/リーフネットワーク環境ファイルを作成した場合は、これらの環境ファイルをロールと NIC テンプレートのマッピングファイルの後に含めます。

   c. 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)。

      **例**

      **overcloud deploy** コマンド例からの以下の抜粋は、コマンドの入力の適切な順序を示しています。

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

```
-e /home/stack/templates/spine-leaf-vips-provisioned.yaml \
-e /home/stack/containers-prepare-parameter.yaml \
-e /home/stack/inject-trust-anchor-hiera.yaml \
-r /home/stack/templates/spine-leaf-roles-data.yaml
...
```

**ヒント**

詳細は、**director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドの作成 を参照してください。

5. **overcloud deploy** コマンドを実行します。
オーバークラウドの作成が完了すると、RHOSP director は、オーバークラウドへのアクセスに役立つ詳細を提供します。

**検証**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドのデプロイメントの検証 のステップを実行します。

**関連情報**

- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの オーバークラウドの作成
- **コマンドラインインターフェイスリファレンス** の overcloud deploy

## 4.11. スパイン/リーフ型デプロイメントへの新たなリーフの追加

ネットワーク容量を増やしたり、新しい物理サイトを追加したりする場合は、Red Hat OpenStack Platform (RHOSP) スパイン/リーフネットワークに新しいリーフを追加する必要があることもあります。

**前提条件**

- RHOSP デプロイメントでスパイン/リーフ型ネットワークトポロジーが使用されている。

**手順**

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. **/home/stack/templates/spine-leaf-networks-data.yaml** などのネットワーク定義テンプレートを開きます。適切なベースネットワークの下に、追加する新しいリーフの設定可能なネットワークアイテムとしてリーフサブネットを追加します。

**例**

以下の例では、新しいリーフ (**leaf3**) のサブネットエントリーが追加されました。

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

4. 追加する新しいリーフ用のロールデータファイルを作成します。

   a. 追加する新しいリーフ用にリーフ Compute およびリーフ Ceph Storage ファイルをコピーします。

      **例**

      この例では、**Compute1.yaml** および **CephStorage1.yaml** が新しいリーフ (**Compute3.yaml** および **CephStorage3.yaml**) にコピーされ、再度実行されます。

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

   b. 新しいリーフファイルのパラメーター **name** と **HostnameFormatDefault** を編集して、それぞれのリーフパラメーターと一致するようにします。

      **例**

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

      **例**

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

c. 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、新しい Leaf ファイルの networks パラメーターを編集します。

**例**

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
```

**例**

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
```

d. 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。ネットワークにすべてのリーフと、新たに追加するリーフを含めます。

**例**

以下の例では、leaf3 が leaf0、leaf1、および leaf2 に追加されます。

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

5. 追加するリーフ用のカスタム NIC 設定を作成します。

a. 追加する新しいリーフ用のリーフ Compute およびリーフ Ceph Storage NIC 設定ファイルをコピーします。

**例**

この例では、**computeleaf1.yaml** および **ceph-storageleaf1.yaml** が新しいリーフ (**computeleaf3.yaml** および **ceph-storageleaf3.yaml**) にコピーされます。

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

6. ロールと各ロールのカスタム NIC テンプレートマッピングを含むカスタム環境ファイル (例: spine-leaf-nic-roles-map.yaml) を開きます。追加する新しいリーフのロールごとにエントリーを挿入します。

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

**例**

この例では、エントリー **ComputeLeaf3NetworkConfigTemplate** および **CephStorage3NetworkConfigTemplate** が追加されています。

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

7. 個別のネットワークマッピングを含み、オーバークラウドのコントロールプレーンネットワークへのアクセスを設定するカスタムネットワーク環境ファイル (**spine-leaf-ctlplane.yaml** など) を開き、コントロールプレーンパラメーターを更新します。
**parameter_defaults** セクションで、新しいリーフネットワークのコントロールプレーンサブネットマッピングを追加します。また、新しいリーフネットワークの外部ネットワークマッピングも含めます。

- フラットネットワークのマッピングの場合には、**NeutronFlatNetworks** パラメーターの新しいリーフ (**leaf3**) をリスト表示し、新しいリーフの **NeutronBridgeMappings** パラメーターを設定します。

```
parameter_defaults:
  NeutronFlatNetworks: leaf0,leaf1,leaf2,leaf3
  Controller0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  Compute0Parameters:
    NeutronBridgeMappings: "leaf0:br-ex"
  Compute1Parameters:
    NeutronBridgeMappings: "leaf1:br-ex"
```

```
  Compute2Parameters:
    NeutronBridgeMappings: "leaf2:br-ex"
  Compute3Parameters:
    NeutronBridgeMappings: "leaf3:br-ex"
```

- VLAN 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、新しいリーフネットワーク用 (**leaf3**) に VLAN をマッピングします。

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

**例**

この例では、フラットネットワークマッピングが使用され、新しいリーフ (**leaf3**) エントリーが追加されます。

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

8. 変更したネットワークをプロビジョニングします。
詳細は、オーバークラウドネットワークとオーバークラウド VIP のプロビジョニング を参照してください。

9. 前に作成したベアメタルノード定義ファイル (例: **spine-leaf-baremetal-nodes.yaml**) で、**network_config_update** 変数が **true** に設定されていることを確認します。

**例**

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

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

10. 変更したノードをプロビジョニングします。
詳細は、ベアメタルノードのプロビジョニング を参照してください。

11. スパイン/リーフ対応オーバークラウドのデプロイの手順に従って、スパイン/リーフ対応オーバークラウド を再デプロイします。