Red Hat OpenStack Platform 17.1

Red Hat OpenStack Platform での動的ルーティングの設定

Red Hat OpenStack Platform での動的ルーティングの使用を可能にするため必要な、director による FRRouting および OVN BGP エージェントの設定

Last Updated: 2024-05-25

Red Hat OpenStack Platform 17.1 Red Hat OpenStack Platform での動的ルーティングの設定

Red Hat OpenStack Platform での動的ルーティングの使用を可能にするため必要な、director による FRRouting および OVN BGP エージェントの設定

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

これは、Red Hat OpenStack Platform で BGP 動的ルーティングをインストール、設定、操作、およびトラブルシューティングするためのガイドです。

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 RHOSP 動的ルーティングの概要

Red Hat OpenStack Platform (RHOSP) は、Border Gateway Protocol (BGP) を使用した動的ルーティングをサポートします。

このセクションに含まれるトピックは次のとおりです。

- RHOSP 動的ルーティングについて
- RHOSP 動的ルーティングで使用される BGP コンポーネント
- BGP 広告とトラフィックのリダイレクト

1.1. RHOSP 動的ルーティングについて

Red Hat OpenStack Platform (RHOSP) は、コントロールプレーンとデータプレーンで、ボーダーゲートウェイプロトコル (BGP) を使用した ML2/OVN 動的ルーティングをサポートします。純粋なレイヤー 3 (L3) データセンターにクラスターをデプロイすると、大規模な障害ドメイン、大量のブロードキャストトラフィック、障害復旧時の長いコンバージェンス時間など、従来のレイヤー 2 (L2) インフラストラクチャーのスケーリングの問題が克服されます。

RHOSP 動的ルーティングは、現時点でほとんどのインターネットサービスプロバイダーが採用している従来のアプローチとは異なる、負荷分散と高可用性のメカニズムを提供します。RHOSP 動的ルーティングを使用すると、コントローラーノード上の共有仮想 IP に L3 ルーティングを使用することで、高可用性が向上します。アベイラビリティーゾーンをまたいでデプロイするコントロールプレーンサーバーでは、個別の L2 セグメント、物理サイト、および電源サーキットを維持します。

RHOSP 動的ルーティングを使用すると、プロバイダーネットワーク上の仮想マシンおよびロードバランサーの IP アドレスを、作成時および起動時、またはそれらが Floating IP アドレスに関連付けられるたびに公開できます。特別なフラグが設定されている場合、プロジェクトネットワークで同じ機能を使用できます。

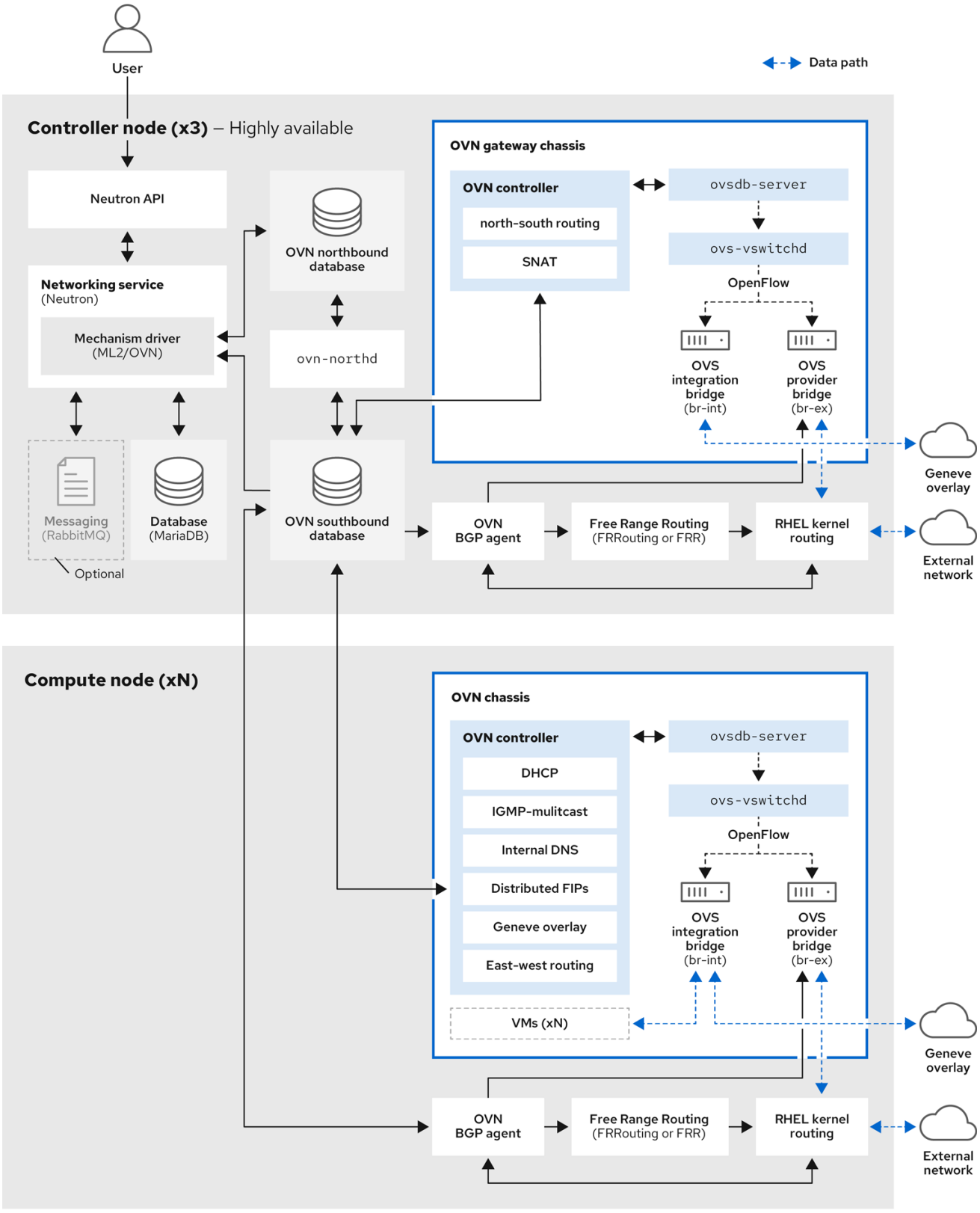
RHOSP 動的ルーティングには、次のような利点もあります。

- データプレーントラフィックの管理の改善。
- ロール間の違いが少なく、シンプルな設定。
- L3 境界にまたがる分散 L2 プロバイダー VLAN および Floating IP (FIP) サブネット。重複する CIDR がない場合、ラックをまたいで VLAN スパンニングを行う必要はありません。
- データセンターおよびエッジサイトのラックおよび L3 境界にまたがる分散コントローラー。
- あるサイトから別のサイトへのパブリックプロバイダー IP または FIP のサブネット全体のフェイルオーバー。
- 次世代のデータセンターとハイパースケールファブリックのサポート。

1.2. RHOSP 動的ルーティングで使用される BGP コンポーネント

Red Hat OpenStack Platform (RHOSP) は、いくつかのコンポーネントに依存して、レイヤー 3 データセンターへの動的ルーティングを提供します。

図1.1 RHOSP 動的ルーティングコンポーネント



OVN BGP エージェント (ovn-bgp-agent コンテナー)

各 RHOSP コントローラーおよびコンピュートノードの **ovn-controller** コンテナーで実行される Python ベースのデーモン。エージェントは、Open Virtual Network (OVN) サウスバウンドデータベースで特定の仮想マシンと Floating IP (FIP) イベントを監視します。これらのイベントが発生すると、エージェントは FRR BGP デーモン (**bgpd**) に通知して、VM に関連付けられた IP アドレスま

たは FIP をアドバタイズします。エージェントは、外部トラフィックを OVN オーバーレイにルーティングするアクションもトリガーします。エージェントはマルチドライバー実装を使用するため、RHOSP や Red Hat OpenShift Platform など、OVN 上で実行される特定のインフラストラクチャー用にエージェントを設定できます。

フリーレンジルーティング (FRRouting、または FRR) (frr コンテナー)

すべてのコンポーザブルロールの **frr** コンテナーで実行され、連携してルーティングテーブルを構築するデーモンの IP ルーティングプロトコルスイート。各プロトコルデーモンは異なる方法を使用して ECMP ポリシーを管理しますが、FRR は等コストマルチパスルーティング (ECMP) をサポートします。Red Hat Enterprise Linux (RHEL) で提供されている FRR スイートは、複数のデーモンを提供します。RHOSP は主に FRR **bgpd** デーモン、**bfdd** デーモン、および Zebra デーモンを使用します。

BGP デーモン (frr コンテナー)

Border Gateway Protocol (BGP) のバージョン 4 を実装するために **frr** コンテナーで実行されるデーモン (**bgpd**)。**bgpd** デーモンは、機能ネゴシエーションを使用して、リモートピアの機能を検出します。ピアが IPv4 ユニキャストネイバーとしてのみ設定されている場合、**bgpd** はケイパビリティーネゴシエーションパケットを送信しません。BGP デーモンは、Zebra デーモンを介してカーネルルーティングテーブルと通信します。

BFD デーモン (frr コンテナー)

Bidirectional Forwarding Detection (BFD) を実装する FRR スイートのデーモン (**bfdd**)。このプロトコルは、隣接する転送エンジン間の障害検出を提供します。

Zebra デーモン (frr コンテナー)

さまざまな FRR デーモンからの情報を調整し、ルーティングの決定をカーネルルーティングテーブルに直接伝達するデーモン。

VTY シェル (frr コンテナー)

FRR デーモン用のシェルである VTY シェル (**vtysh**) は、各デーモンで定義されたすべての CLI コマンドを集約し、単一のシェルで提供します。

関連情報

- FRRouting ドキュメント

1.3. BGP 広告とトラフィックのリダイレクト

Red Hat OpenStack Platform (RHOSP) 動的ルーティングを使用する場合、ネットワークトラフィックは、アドバタイズされたルートを使用して、仮想マシン、ロードバランサー、Floating IP (FIP) の間で送受信されます。トラフィックがノードに到達すると、OVN BGP エージェントは、Red Hat Enterprise Linux (RHEL) カーネルネットワークを使用して、トラフィックを OVS プロバイダーブリッジ (**br-ex**) にリダイレクトする IP ルール、ルート、OVS フロールールを追加します。

ネットワークルートをアドバタイズするプロセスは、OVN BGP エージェントが、フリーレンジルーティング (FRRouting、または FRR) をトリガーして、直接接続されたルートをアドバタイズおよび撤回することから始まります。OVN BGP エージェントは、次の手順を実行して、FRR を適切に設定し、IP アドレスが、**bgp-nic** インターフェイスに追加されるたびに、アドバタイズされるようにします。

1. FRR は、VTY シェルを起動して、FRR ソケットに接続します。

```
$ vtysh --vty_socket -c <command_file>
```

2. VTY シェルは、次のコマンドを含むファイルを渡します。

```
LEAK_VRF_TEMPLATE = '''
```

```
router bgp {{ bgp_as }}
  address-family ipv4 unicast
    import vrf {{ vrf_name }}
  exit-address-family

  address-family ipv6 unicast
    import vrf {{ vrf_name }}
  exit-address-family

router bgp {{ bgp_as }} vrf {{ vrf_name }}
  bgp router-id {{ bgp_router_id }}
  address-family ipv4 unicast
    redistribute connected
  exit-address-family

  address-family ipv6 unicast
    redistribute connected
  exit-address-family
...
```

VTY シェルが渡すコマンドは、次を実行します。

a. デフォルトで **bgp_vrf** という名前の VRF を作成します。

b. ダミーインターフェイスタイプを VRF に関連付けます。
デフォルトでは、ダミーインターフェイスの名前は **bgp-nic** です。

c. IP アドレスを OVS プロバイダーブリッジに追加して、Address Resolution Protocol (ARP) と Neighbor Discovery Protocol (NDP) が有効になっていることを確認します。

3. Zebra デーモンは、VM とロードバランサーの IP アドレスがローカルインターフェイスで追加および削除されるのを監視し、Zebra はルートをアドバタイズおよび撤回します。
FRR は、**redistribute connected** オプションを有効にして、設定されているため、ルートのアドバタイズと撤回は、ダミーインターフェイス **bgp-nic** からルートを公開または削除するだけで、設定されます。

注記

テナントネットワークに接続された仮想マシンの公開は、デフォルトで無効になっています。RHOSP 設定で有効にされている場合、OVN BGP エージェントは neutron ルーターゲートウェイポートを公開します。OVN BGP エージェントは、テナントネットワーク上の仮想マシンに流れるトラフィックを、**chassisredirect** 論理ルーターポート (**CR-LRP**) をホストするノードを介して OVN オーバーレイに注入します。

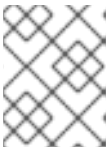
4. FRR は、VM またはロードバランサーをホスティングするノード、または OVN ルーターゲートウェイポートを含むノードの IP アドレスを公開します。

OVN BGP エージェントは、RHEL カーネルネットワークと OVS を使用して、OVN オーバーレイとの間でトラフィックをリダイレクトするために必要な設定を実行し、FRR は適切なノードで IP アドレスを公開します。

OVN BGP エージェントが起動すると、次の処理が実行されます。

1. OVS プロバイダーブリッジに IP アドレスを追加して、ARP と NDP を有効にします。

2. **/etc/iproute2/rt_tables** の各 OVS プロバイダーブリッジのルーティングテーブルにエントリーを追加します。



注記

RHEL カーネルの場合、ルーティングテーブルの最大数は 252 です。これにより、プロバイダーネットワークの数が 252 に制限されます。

3. VLAN デバイスをブリッジに接続し、ARP および NDP を有効にします (VLAN プロバイダーネットワークのみ)。

4. OVS プロバイダーブリッジで余分な OVS フローをクリーンアップします。

通常の再同期イベント中または起動中に、OVN BGP エージェントは次のアクションを実行します。

1. ルーティングテーブルに特定のルートを適用する IP アドレスルールを追加します。
次の例では、このルールは OVS プロバイダーブリッジに関連付けられています。

```
$ ip rule

0:      from all lookup local
1000:   from all lookup [l3mdev-table]
*32000:  from all to IP lookup br-ex*  # br-ex is the OVS provider bridge
*32000:  from all to CIDR lookup br-ex*  # for VMs in tenant networks
32766:  from all lookup main
32767:  from all lookup default
```

2. IP アドレスルートを OVS プロバイダーブリッジルーティングテーブルに追加して、トラフィックを OVS プロバイダーブリッジデバイスにルーティングします。

```
$ ip route show table br-ex

default dev br-ex scope link
*CIDR via CR-LRP_IP dev br-ex*  # for VMs in tenant networks
*CR-LRP_IP dev br-ex scope link*  # for the VM in tenant network redirection
*IP dev br-ex scope link*  # IPs on provider or FIPs
```

3. IPv4 または IPv6 のどちらを使用しているかに応じて、次のいずれかの方法を使用して、トラフィックを OVS プロバイダーブリッジ (**br-ex**) 経由で OVN にルーティングします。

   a. IPv4 の場合は、OVN ルーターゲートウェイポート **CR-LRP** の静的 ARP エントリーを追加します。これは、OVN がその L2 ネットワークの外部の ARP 要求に応答しないためです。

```
$ ip nei

...
CR-LRP_IP dev br-ex lladdr CR-LRP_MAC PERMANENT
...
```

   b. IPv6 の場合は、NDP プロキシーを追加します。

```
$ ip -6 nei add proxy CR-LRP_IP dev br-ex
```

4. OVS プロバイダーブリッジに新しいフローを追加して、OVN オーバーレイからカーネルネッ

トワークにトラフィックを送信します。これにより、宛先 MAC アドレスが OVS プロバイダーブリッジの MAC アドレスに変更されます (**actions=mod_dl_dst:OVN_PROVIDER_BRIDGE_MAC,NORMAL**)。

```
$ sudo ovs-ofctl dump-flows br-ex

cookie=0x3e7, duration=77.949s, table=0, n_packets=0, n_bytes=0, priority=
900,ip,in_port="patch-provnet-1" actions=mod_dl_dst:3a:f7:e9:54:e8:4d,NORMAL
cookie=0x3e7, duration=77.937s, table=0, n_packets=0, n_bytes=0, priority=
900,ipv6,in_port="patch-provnet-1" actions=mod_dl_dst:3a:f7:e9:54:e8:4d,NORMAL
```

第2章 RHOSP 動的ルーティングを使用するデプロイメントのプランニング

Red Hat OpenStack Platform 環境に動的ルーティングを実装する計画がある場合は、サポートされる機能、依存関係、制約、必要なネットワークトポロジーを評価してください。

このセクションに含まれるトピックは次のとおりです。

2.1. RHOSP 動的ルーティングのサポートマトリクス

以下の表は、Red Hat OpenStack Platform (RHOSP) 17.1 でサポートされる動的ルーティング機能の一覧を示しています。

注記

この機能が一覧にない場合、RHOSP 17.1 はこの機能をサポートしません。

表2.1 RHOSP 動的ルーティング機能のサポートマトリクス

機能	RHOSP 17.1? でのサポート状況
カーネルルーティング	はい
IPv6	はい
EVPN	いいえ
OVS-DPDK ルーティング	いいえ
SR-IOV ルーティング	いいえ
CIDR の重複	いいえ
フローティング IP の選択的公開	いいえ

2.2. RHOSP 動的ルーティングの要件

動的ルーティングを備えた Red Hat OpenStack Platform (RHOSP) には、以下のネットワークトポロジーおよびソフトウェアが必要です。

- スパイン/リーフ型ネットワークトポロジー

- 以下を備えた、RHOSP バージョン 17.0 以降を実行する環境。
  - ML2/OVN メカニズムドライバー。
  - ループバックインターフェイスの RHOSP アンダークラウドで設定された Border Gateway Protocol (BGP) および VIP。
  - RHOSP オーバークラウドにデプロイされた OVN BGP エージェント。
- BGP とピアリングするネットワークデバイスには、BGP の実装がインストールされている必要があります。(BGP はほとんどのデバイスの標準であり、デバイスベンダーによって提供されます。)

2.3. RHOSP 動的ルーティングの制約

Red Hat OpenStack Platform (RHOSP) 環境で動的ルーティングを計画する際には、以下の制約を考慮してください。

- どの仮想マシンとロードバランサー (LB) を公開するかを制御することはできません。プロバイダーネットワーク上の、またはFloating IP を持つ仮想マシンおよび LB はすべて公開されます。さらに、**expose_tenant_network** フラグが有効になっている場合、プロジェクトネットワークの仮想マシンが公開されます。
- 重複する CIDR はサポートされていないため、アドレススコープとサブネットプールを使用する必要があります。
- BGP は、IP ルートとルールによるカーネルルーティングを使用して、ネットワークトラフィックを誘導します。そのため、カーネル領域を使用しない OVS-DPDK はサポートされません。
- RHOSP ネットワークでは、ロードバランサーメンバーをプロバイダーネットワークに接続するルーターの場合、プロバイダー上の OVN-octavia 仮想 IP またはプロジェクトネットワーク上の VIP に関連付けられた FIP への north-south トラフィックは、neutron ルーターゲートウェイをホストするネットワークノードを通過する必要があります。

  注記

  これらのノードのポートは、**chassisredirect** 論理ルーターポート (**cr-lrp**) と呼ばれます。

  このため、OVN オーバーレイへのエントリーポイントは、これらのネットワークノードの 1 つにする必要があり、その結果、VIP および VIP への FIP はこれらのノードを介して公開されます。ネットワークノードから、トラフィックは通常のトンネルパス (Geneve トンネル) をたどって、選択したメンバーが配置されている RHOSP コンピュートノードに到達します。
- 現在、Floating IP (FIP) アドレスのポートへの転送が失敗するという既知の問題があります。代わりに、FIP 用のネットワークトラフィックは、ポート転送を実行するように設定されているポートの一覧に含まれるテナントポート IP アドレスに転送されます。この失敗は、OVN BGP エージェントが FIP にルートを公開していないことが原因で発生します。現在、回避策はありません。詳細は、BZ 2160481 を参照してください。
- 現在、Red Hat OpenStack Platform Compute サービスが、マルチキャスト IP アドレスの宛先に送信されたパケットをルーティングできないという既知の問題があります。したがって、マルチキャストグループに登録されている仮想マシンインスタンスは、送信されたパケットを受信できません。原因は、BGP マルチキャストルーティングがオーバークラウドノードで適切に設定されていないことです。現在、回避策はありません。詳細は、BZ 2163477 を参照してください。

- RHOSP 17.1 のバージョン間の更新中に、すべてのノードの FRR コンポーネントを再起動する必要があり、次のイベントが発生します。そのため、接続のダウンタイムが発生します。

  1. 各ノードは、BGP セッションをピアルーターで再確立し、コントロールプレーンとデータプレーントラフィックの両方に影響を与えます。

  2. BGP セッションが再確立された後、FRR はノードとの間でルートを取得して再アドバタイズします。つまり、ルートは数秒間は使用できなくなります。

  3. 最後に、**ovn-bgp-agent** は、実行中の FRR サービスに VRF 設定を追加し、BGP を使用してデータプレーントラフィックのルートをアドバタイズします。

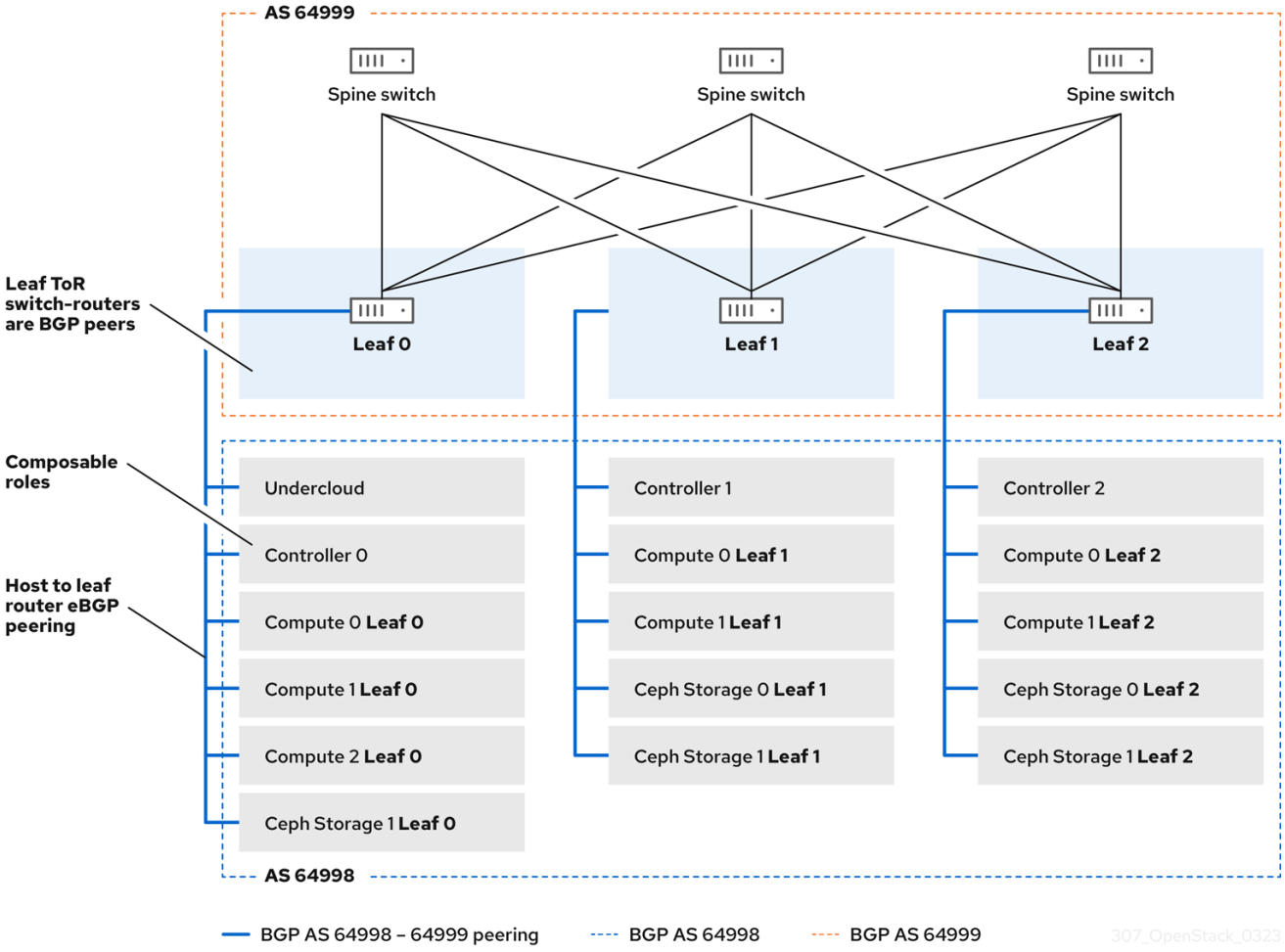
- ストレージノードを RHOSP の動的環境に追加する場合は、プロビジョニングネットワークを使用する必要があります。接続を確立するルーターを作成する前に、RHOSP director が Red Hat Ceph Storage を必要とします。

2.4. RHOSP 動的ルーティングトポロジーの例

以下の図は、Red Hat OpenStack Platform (RHOSP) ML2/OVN 環境で動的ルーティングを使用するネットワークトポロジーの例を示しています。

このネットワークトポロジーの例は、3 つのリーフを持つスパインを示しています。トップオブラックスイッチルーターは、スパイン上のスイッチとの BGP ピアです。

図2.1 リーフルーターへのオーバークラウドホストの BGP ピアリング



第3章 RHOSP 動的ルーティング用のアンダークラウドのデプロイ

アンダークラウドは、最終的な Red Hat OpenStack Platform (RHOSP) 環境 (オーバークラウドと呼ばれる) の設定、インストール、および管理をコントロールするノードです。アンダークラウドは、OVN BGP エージェントを含め、コンテナーで実行される OpenStack Platform コンポーネントサービスを使用します。これらのコンテナー化されたサービスは、オーバークラウドの作成および管理に使用する RHOSP director と呼ばれるツールを構成します。

このセクションに含まれるトピックは次のとおりです。

- RHOSP 動的ルーティング用のアンダークラウドのインストールと設定

3.1. RHOSP 動的ルーティング用のアンダークラウドのインストールと設定

Red Hat OpenStack Platform (RHOSP) director を使用して、RHOSP アンダークラウドに動的ルーティング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. (オプション) **frr-parameters.yaml** でアンダークラウドの BGP 設定値を設定します。
2. **undercloud.conf** でアンダークラウドのスパインリーフネットワークトポロジーの設定値を設定します。
3. **openstack undercloud install** コマンドを実行します。

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. BGP を使用して、他のラックとオーバークラウドノードに到達する予定の場合は、カスタムヒート環境ファイル **/home/stack/templates/frr-parameters.yaml** に以下のパラメーターを追加して、アンダークラウドにインストールされるように、FRRouting (FRR) を設定します。

注記

このパスを控えておいてください。後のステップで必要になります。

例

```
parameter_defaults:
  ContainerFrrImage: registry.redhat.io/rhosp-17.1/openstack-frr-rhel9:17.1.1
  FrrBfdEnabled: true
  FrrBgpEnabled: true
  FrrBgpAsn: 64999
  FrrBgpUplinks: ['nic2', 'nic3']
  FrrBgpUplinksScope: internal
  FrrLogLevel: debugging
  FrrBgpRouterID: 172.30.4.1
  FrrBgpIpv4SrcIp: 172.30.4.1
  FrrBgpIpv6SrcIp: fe80::5054:ff:fe74:73ce
```

ヒント

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

FrrBfdEnabled

true の場合、Bidirectional Forwarding Detection (BFD) を有効にします。デフォルトは **false** です。

FrrBgpEnabled

true の場合、Border Gateway Protocol (BGP) を有効にします。デフォルトは **true** です。

FrrBgpAsn

FRRouting 内で使用されるデフォルトの ASN。デフォルトは **65000** です。**FrrBgpAsn** は、使用されるロールごとに異なる値に設定できます。

FrrBgpUplinks

アップリンクネットワークインターフェイスのコンマ区切りリスト。デフォルトは **['nic1', 'nic2']** です。

FrrBgpUplinksScope

内部 (iBGP) または外部 (eBGP) ネイバーとピアリングします。デフォルトは **internal** です。

FrrLogLevel

emergencies、**alerts**、**critical**、**errors**、**warnings**、**notifications**、**informational**、**debugging** の値セットを使用して、FRR のログレベルを指定します。デフォルトは **informational** です。

FrrBgpRouterID

FRR で使用される BGP **router_id**。

FrrBgpIpv4SrcIp

IPv4 ネットワークトラフィックの送信元 IP アドレス。

FrrBgpIpv6SrcIp

IPv6 ネットワークトラフィックの送信元 IP アドレス。

tripleo_frr_bgp_peers

ピアリングする Free Range Routing (FRR) の IP アドレスまたはホスト名のリストを指定するために使用されるロール固有のパラメーター。

tripleo_frr_ovn_bgp_agent_enable

データプレーンルートが公開されない RHOSP ノード上で OVN BGP エージェントを有効または無効にするために使用されるロール固有のパラメーター。デフォルト値は **true** です。

4. **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。

```
$ cp /usr/share/python-tripleoclient/undercloud.conf.sample \
~/templates/undercloud.conf
```

5. **DEFAULT** セクションで、次の一般パラメーター値を設定します。

例

```
[DEFAULT]
# General
```

```
cleanup = false
container_images_file=/home/stack/templates/
\containers-prepare-parameter.yaml
overcloud_domain_name = {{ cloud_domain }}
undercloud_timezone = UTC
undercloud_hostname = undercloud-0.{{ cloud_domain }}

# BGP on undercloud
...

# TLS-e
...

# Networking
...

# Subnets
...
```

ヒント

詳細は、**director を使用した Red Hat OpenStack Platform のインストールおよび管理**の アンダークラウド設定パラメーター を参照してください。

overcloud_domain_name

オーバークラウドをデプロイする際に使用する DNS ドメイン名を指定します。次のステップでは、この値がオーバークラウドの **CloudDomain** パラメーターの値と一致することを確認する必要があります。

cleanup

一時的なファイルを削除します。デプロイメント中に使用される一時ファイルを保持するには、これを **false** に設定します。一時ファイルは、エラーが発生した場合にデプロイメントのデバッグに役立ちます。

container_images_file

コンテナーイメージ情報を含む Heat 環境ファイルを指定します。

container_insecure_registries

podman が使用するセキュアではないレジストリーのリスト。プライベートコンテナーレジストリー等の別のソースからイメージをプルする場合には、このパラメーターを使用します。

custom_env_files

アンダークラウドのインストールに追加する新たな環境ファイル

undercloud_hostname

アンダークラウドの完全修飾ホスト名を定義します。本パラメーターを指定すると、アンダークラウドのインストールでホスト名すべてに設定されます。指定しないと、アンダークラウドは現在のホスト名を使用しますが、システムのホスト名すべてを適切に設定しておく必要があります。

undercloud_timezone

アンダークラウド用ホストのタイムゾーン。タイムゾーンを指定しなければ、director は既存のタイムゾーン設定を使用します。

6. アンダークラウドに BGP をインストールする場合は、**DEFAULT** セクションでアンダークラウドの FRR を有効にし、前の手順で FRR パラメーター値を設定したカスタム環境ファイルを指定します。

例

```
[DEFAULT]
# General
...

# BGP on undercloud
enable_frr=true
custom_env_files=/home/stack/templates/frr-parameters.yaml

# TLS-e
...

# Networking
...

# Subnets
...
```

7. TLS-everywhere を使用している場合は、**[DEFAULT]** セクションで、次の TLS-everywhere パラメーター値を設定します。

例

```
[DEFAULT]
# General
...

# BGP on undercloud
...

# TLS-e
enable_novajoin = False
undercloud_nameservers = {{ freeipa_ip }}
generate_service_certificate = True
ipa_otp = {{ undercloud_otp }}

# Networking
...

# Subnets
...
```

ヒント

詳細は、**director を使用した Red Hat OpenStack Platform のインストールおよび管理**の アンダークラウド設定パラメーター を参照してください。

enable_novajoin

true の場合、novajoin サービスが TLS をデプロイできるようにします。

undercloud_nameservers

アンダークラウドのネームサーバーの DNS サーバーの現在の IP アドレスを指定します。この情報は **/etc/resolv.conf** にあります。

generate_service_certificate

アンダークラウドのインストール時に SSL/TLS 証明書を生成するかどうかを定義します。これは **undercloud_service_certificate** パラメーターに使用します。

ipa_otp

FreeIPA OTP ファクトを設定します。

8. **DEFAULT** セクションで、次のネットワークパラメーター値を設定します。

例

```
[DEFAULT]
# General
...

# BGP on undercloud
...

# TLS-e
...

# Networking
local_interface = eth0
local_ip = {{ undercloud_ctlplane }}/24
undercloud_public_host = {{ undercloud_public_host }}
undercloud_admin_host = {{ undercloud_admin_host }}

# Subnets
...
```

ヒント

詳細は、**director を使用した Red Hat OpenStack Platform のインストールおよび管理**の アンダークラウド設定パラメーター を参照してください。

local_interface

ローカルネットワーク用にブリッジするインターフェイス。

local_ip

leaf0 上のアンダークラウドの IP アドレス。

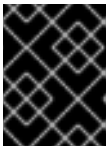
undercloud_public_host

アンダークラウドの外部向け IP アドレス。

undercloud_admin_host

アンダークラウドの管理 IP アドレス。この IP アドレスは、通常 leaf0 上にあります。

9. 以前に **subnets** パラメーターで定義したサブネットごとに新しいセクションを作成します。



重要

director がサブネットを作成した後、director はサブネットの IP アドレスを変更することはできません。

例

```
[DEFAULT]
# General
...

# BGP on undercloud
...

# TLS-e
...

# Networking
...

# Subnets
[r1]
# This subnet is used for overcloud nodes deployed on rack1.
cidr = 192.168.1.0/24
dhcp_start = 192.168.1.150
dhcp_end = 192.168.1.170
inspection_iprange = 192.168.1.171,192.168.1.185
gateway = 192.168.1.1
masquerade = False
[r2]
# This subnet is used for overcloud nodes deployed on rack2.
cidr = 192.168.2.0/24
dhcp_start = 192.168.2.150
dhcp_end = 192.168.2.170
inspection_iprange = 192.168.2.171,192.168.2.185
gateway = 192.168.2.1
masquerade = False
[r3]
# This subnet is used for overcloud nodes deployed on rack3.
cidr = 192.168.3.0/24
dhcp_start = 192.168.3.150
dhcp_end = 192.168.3.170
inspection_iprange = 192.168.3.171,192.168.3.185
gateway = 192.168.3.1
masquerade = False
[r4]
# This subnet is used for the underloud node and potentially FreeIPA
# that are deployed on rack4.
cidr = 192.168.4.0/24
dhcp_start = {{ undercloud_dhcp_start }}
dhcp_end = 192.168.4.170
inspection_iprange = 192.168.4.171,192.168.4.185
gateway = 192.168.4.1
masquerade = False
```

ヒント

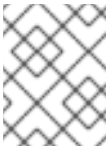
詳細は、**director を使用した Red Hat OpenStack Platform のインストールおよび管理**ガイドの サブネット を参照してください。

cidr

オーバークラウドインスタンスの管理に director が使用するネットワーク。これは、アンダークラウドの **neutron** サービスが管理するプロビジョニングネットワークです。プロビジョニングネットワークに別のサブネットを使用しない限り、この値はデフォルト (**192.168.24.0/24**) のままにします。

masquerade

外部アクセスのために、**cidr** で定義したネットワークをマスカレードするかどうかを定義します。このパラメーターにより、director 経由で外部アクセスすることができるように、プロビジョニングネットワークにネットワークアドレス変換 (NAT) のメカニズムが提供されます。



注記

director 設定は、適切な sysctl カーネルパラメーターを使用して IP フォワーディングも自動的に有効化します。

dhcp_start と dhcp_end

オーバークラウドノードの DHCP 割り当て範囲 (開始アドレスと終了アドレス)。ノードを割り当てるのに十分な IP アドレスがこの範囲に含まれるようにします。

dhcp_exclude

DHCP 割り当て範囲で除外する IP アドレス

dns_nameservers

サブネットに固有の DNS ネームサーバー。サブネットにネームサーバーが定義されていない場合には、サブネットは **undercloud_nameservers** パラメーターで定義されるネームサーバーを使用します。

gateway

オーバークラウドインスタンスのゲートウェイ。外部ネットワークにトラフィックを転送するアンダークラウドのホストです。director に別の IP アドレスを使用する場合または直接外部ゲートウェイを使用する場合以外は、この値はデフォルト (**192.168.24.1**) のままにします。

10. インストールコマンドを実行します。

```
$ openstack undercloud install
```

11. 各リーフおよびラックに到達するための追加のネットワークルートを含む、アンダークラウドのネットワーク設定が正しいことを確認します。
詳細は、**director を使用した Red Hat OpenStack Platform のインストールおよび管理**の director 設定パラメーター を参照してください。

検証

1. director 設定スクリプトは、すべてのサービスを自動的に開始します。RHOSP サービスコンテナーが実行中であることを確認します。

```
$ sudo podman ps -a --format "{{.Names}} {{.Status}}"
```

出力例

次のような出力が表示されるはずです。これは、RHOSP サービスコンテナーが **Up** の状態にあることを示します。

```
memcached Up 3 hours (healthy)
haproxy Up 3 hours
rabbitmq Up 3 hours (healthy)
mysql Up 3 hours (healthy)
iscsid Up 3 hours (healthy)
keystone Up 3 hours (healthy)
keystone_cron Up 3 hours (healthy)
neutron_api Up 3 hours (healthy)
logrotate_crond Up 3 hours (healthy)
neutron_dhcp Up 3 hours (healthy)
neutron_l3_agent Up 3 hours (healthy)
neutron_ovs_agent Up 3 hours (healthy)
ironic_api Up 3 hours (healthy)
ironic_conductor Up 3 hours (healthy)
ironic_neutron_agent Up 3 hours (healthy)
ironic_pxe_tftp Up 3 hours (healthy)
ironic_pxe_http Up 3 hours (unhealthy)
ironic_inspector Up 3 hours (healthy)
ironic_inspector_dnsmasq Up 3 hours (healthy)
neutron-dnsmasq-qdhcp-30d628e6-45e6-499d-8003-28c0bc066487 Up 3 hours
...
```

2. コマンドラインツールを使用するために **stack** ユーザーを初期化できることを確認します。

```
$ source ~/stackrc
```

プロンプトに **(undercloud)** が表示される場合、これは、OpenStack コマンドが認証され、アンダークラウドに対して実行されることを示しています。

出力例

```
(undercloud) [stack@director ~]$
```

director のインストールが完了しました。これで、director コマンドラインツールが使用できるようになりました。

関連情報

- **director を使用した Red Hat OpenStack Platform のインストールおよび管理**の アンダークラウド設定パラメーター
- director を使用した Red Hat OpenStack Platform のインストールと管理ガイド の **サブネット**
- **オーバークラウドのパラメーター** の ネットワーク (neutron) パラメーター
- **director を使用した Red Hat OpenStack Platform のインストールと管理**ガイドの デプロイメントコマンドのオプション

第4章 RHOSP 動的ルーティング用のオーバークラウドのデプロイ

Red Hat OpenStack Platform (RHOSP) director を使用して、オーバークラウドに RHOSP 動的ルーティング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 (Free Range Routing (FRR) ロールを含む) 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. 該当する場合は、オーバークラウドのエンドポイントに仮想 IP (VIP) を定義し、各 VIP 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. Ceph を動的ルーティング環境にデプロイします。
11. 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

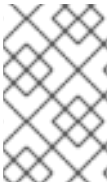
以下の例は、内部 API ネットワークおよびそのリーフネットワークを定義する方法を示しています。

```
- 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
```

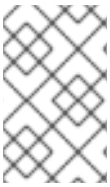
注記

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、RHOSP 動的ルーティング用のアンダークラウドのデプロイ 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/Controller0.yaml ~/roles/Controller1.yaml
$ cp ~/roles/Controller0.yaml ~/roles/Controller2.yaml
$ 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. 次のステップ リーフネットワークの設定 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 OVS プロバイダーブリッジ (**br-ex**) 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。

いくつかの基本的な FRRouting および OVN BGP エージェント設定を行う必要があります。

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

- RHOSP アンダークラウドにアクセスできる **stack** ユーザーである必要がある。
- アンダークラウドがインストールされる。

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. **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 OVS プロバイダーブリッジ (**br-ex**) 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。

例

```
parameter_defaults:
```

```
  NeutronFlatNetworks: provider1
  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

ヒント

詳細は、第 17 章ネットワーク (neutron) パラメータ (**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. 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: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

注記

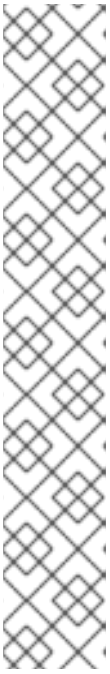
スパイン/リーフトポロジーでは、フラットネットワークと VLAN の両方を使用できます。

5. **<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:
  NeutronNetworkType: 'geneve,vlan'
  NeutronNetworkVLANRanges: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r1

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r2

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r3

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]

  ComputeRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
```

6. 各リーフの OVN BGP エージェント、FRRouting、CMS オプションを設定します。



注記

FRR サービスは、すべての RHOSP ノードで実行され、コントロールプレーンと、データプレーン全体のさまざまなノードで実行されているサービスとの間の接続を提供します。ただし、OVN BGP エージェントは、すべてのコンピュートノードと **enable-chassis-as-gw** で設定されたノードでのみ実行する必要があります。

データプレーンルートが公開されていない RHOSP ノードの場合、**tripleo_frr_ovn_bgp_agent_enable** パラメーターを **false** に設定して、これらのロールの OVN BGP エージェントを無効にします。デフォルトは **true** です。

例

```
parameter_defaults:
  DatabaseRack1ExtraGroupVars:
    tripleo_frr_ovn_bgp_agent_enable: false
```

例

```
parameter_defaults:
  NeutronNetworkType: 'geneve,vlan'
  NeutronNetworkVLANRanges: 'provider2:1:1000'

  ControllerRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r1
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ControllerRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r2
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ControllerRack3Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    ControlPlaneSubnet: r3
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
    FrrOvnBgpAgentExposeTenantNetworks: True
    OVNCMSOptions: "enable-chassis-as-gw"

  ComputeRack1Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'

  ComputeRack2Parameters:
    NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
    FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
```

```
ComputeRack3Parameters:
  NeutronBridgeMappings: ["provider1:br-ex", "provider2:br-vlan"]
  FrrOvnBgpAgentDriver: 'ovn_bgp_driver'
```

ヒント

詳細は、第 17 章ネットワーク (neutron) パラメータ (**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。

OVNCMSOptions

OVSDB で設定する CMS オプション

FrrOvnBgpAgentReconcileInterval

ステータスをチェックする頻度を定義して、正しい場所で正しい IP のみが公開されるようにします。デフォルト: 120

FrrOvnBgpAgentOvsdbConnection

ネイティブ OVSDB バックエンドの接続文字列。TCP 接続には **tcp:<IP_address>:<port>** を使用します。デフォルト: **tcp:127.0.0.1:6640**

FrrOvnBgpAgentExposeTenantNetworks

MP-BGP IPv4 および IPv6 ユニキャストを介してテナントネットワーク上の VM IP を公開します。BGP ドライバーが必要です (THT パラメーター FrrOvnBgpAgentDriver を参照)。デフォルト: false

FrrOvnBgpAgentDriver

VM IP が BGP 経由で公開される方法を設定します。EVPN ドライバーは、プロバイダーネットワーク上の VM IP と、MP-BGP IPv4 および IPv6 ユニキャストを介してテナントネットワーク上の VM に関連付けられた FIP を公開します。BGP ドライバーは、MP-BGP EVPN VXLAN 経由でテナントネットワーク上の VM IP を公開します。デフォルト: **ovn_evpn_driver**

FrrOvnBgpAgentAsn

BGP モードでの実行時にエージェントが使用する Autonomous System 番号 (ASN)。デフォルト: 64999**FrrOvnBgpAgentAsn** は、使用されるロールごとに異なる値に設定できます。

FrrLogLevel

ログレベル。デフォルト: **情報提供**

FrrBgpAsn

FRR 内で使用されるデフォルトの ASN。デフォルトは 65000 です。**FrrBgpAsn** は、使用されるロールごとに異なる値に設定できます。

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。
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 VIP 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。

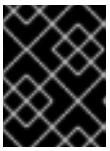
ヒント

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

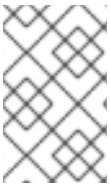
1. 事前にプロビジョニングされたノードを使用している場合は、オーバークラウドデプロイメントコマンドの実行 にスキップしてください。
2. それ以外の場合は、次のステップ オーバークラウドにベアメタルノードを登録する に進みます。

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

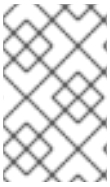
- 次のステップ オーバークラウド上のベアメタルノードのイントロスペクション に進みます。

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
```

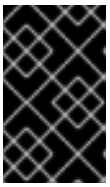
3. pre-introspection 検証グループを実行して、プリイントロスペクションの要件を確認します。

```
$ 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 オーバークラウド用のベアメタルノードのプロビジョニング に進みます。

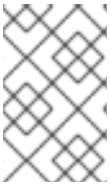
関連情報

- **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: ControllerRack1
  count: 1
  hostname_format: ctrl-1-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r1.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network1
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: ctrl-1-0
    name: ctrl-1-0
    capabilities:
      node: ctrl-1-0
    networks:
    - network: ctlplane
```

```
      vif: true
    - network: left_network
      fixed_ip: 100.65.1.2
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.2
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.1
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0001
      subnet: main_network_ipv6_r1
- name: ComputeRack1
  count: 2
  hostname_format: cmp-1-%index%
  defaults:
    network_config:
      default_route_network:
        - ctlplane
      template: /home/stack/tht/nics_r1.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network1
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: cmp-1-0
    name: cmp-1-0
    capabilities:
      node: cmp-1-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.1.6
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.6
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.2
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0004
      subnet: main_network_ipv6_r1
  - hostname: cmp-1-1
    name: cmp-1-1
    capabilities:
      node: cmp-1-1
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
```

```
      fixed_ip: 100.65.1.10
      subnet: left_network_r1
    - network: right_network1
      fixed_ip: 100.64.0.10
      subnet: right_network1_sub
    - network: main_network
      fixed_ip: 172.30.1.3
      subnet: main_network_r1
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0005
      subnet: main_network_ipv6_r1
- name: ControllerRack2
  count: 1
  hostname_format: ctrl-2-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r2.yaml
    networks:
      - network: ctlplane
        vif: true
      - network: left_network
      - network: right_network2
      - network: main_network
      - network: main_network_ipv6
  instances:
  - hostname: ctrl-2-0
    name: ctrl-2-0
    capabilities:
      node: ctrl-2-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.2
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.2
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.1
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0002
      subnet: main_network_ipv6_r1
- name: ComputeRack2
  count: 2
  hostname_format: cmp-2-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r2.yaml
    networks:
    - network: ctlplane
```

```
      vif: true
    - network: left_network
    - network: right_network2
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: cmp-2-0
    name: cmp-2-0
    capabilities:
      node: cmp-2-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.6
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.6
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.2
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0006
      subnet: main_network_ipv6_r1
  - hostname: cmp-2-1
    name: cmp-2-1
    capabilities:
      node: cmp-2-1
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.2.10
      subnet: left_network_r2
    - network: right_network2
      fixed_ip: 100.64.0.10
      subnet: right_network2_sub
    - network: main_network
      fixed_ip: 172.30.2.3
      subnet: main_network_r2
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0007
      subnet: main_network_ipv6_r1
- name: ControllerRack3
  count: 1
  hostname_format: ctrl-3-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r3.yaml
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
```

```
    - network: right_network3
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: ctrl-3-0
    name: ctrl-3-0
    capabilities:
      node: ctrl-3-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.2
      subnet: left_network_r3
    - network: right_network3
      fixed_ip: 100.64.0.2
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.1
      subnet: main_network_r3
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0003
      subnet: main_network_ipv6_r1
- name: ComputeRack3
  count: 2
  hostname_format: cmp-3-%index%
  defaults:
    network_config:
      default_route_network:
      - ctlplane
      template: /home/stack/tht/nics_r3.yaml
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
    - network: right_network3
    - network: main_network
    - network: main_network_ipv6
  instances:
  - hostname: cmp-3-0
    name: cmp-3-0
    capabilities:
      node: cmp-3-0
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.6
      subnet: left_network_r3
    - network: right_network3
      fixed_ip: 100.64.0.6
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.2
      subnet: main_network_r3
    - network: main_network_ipv6
```

```
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0008
      subnet: main_network_ipv6_r1
  - hostname: cmp-3-1
    name: cmp-3-1
    capabilities:
      node: cmp-3-1
    networks:
    - network: ctlplane
      vif: true
    - network: left_network
      fixed_ip: 100.65.3.10
      subnet: left_networ10_r3
    - network: right_network3
      fixed_ip: 100.64.0.10
      subnet: right_network3_sub
    - network: main_network
      fixed_ip: 172.30.3.3
      subnet: main_network_r3
    - network: main_network_ipv6
      fixed_ip: f00d:f00d:f00d:f00d:f00d:f00d:f00d:0009
      subnet: main_network_ipv6_r1
```

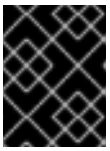
ヒント

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

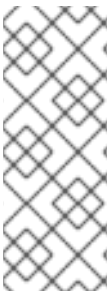
- 次のステップ 動的ルーティング環境への Ceph のデプロイ に進みます。

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. 動的ルーティング環境への CEPH のデプロイ

通常の設定をいくつか調整すると、動的ルーティングを使用する Red Hat OpenStack Platform (RHOSP) 環境に Red Hat Ceph Storage をデプロイできます。



注記

動的ルーティングを使用する Red Hat OpenStack Platform 環境に Red Hat Ceph Storage をインストールする場合は、オーバークラウドをデプロイする前に Ceph をインストールする必要があります。次の設定例では、プロビジョニングネットワークを使用して Ceph をデプロイします。最適なパフォーマンスを得るために、RHOSP 動的ルーティング環境に Ceph Storage をデプロイする場合は、専用の NIC とネットワークハードウェアを使用することを推奨します。

手順

1. Red Hat Ceph Storage をインストールする方法については、**director を使用した Red Hat Ceph Storage および Red Hat OpenStack Platform のデプロイ** 手順に従ってください。Ceph Storage クラスターを設定するステップで、次の追加のステップを実行します。

2. Ceph 設定ファイルを作成し、**[global]** という名前のセクションを追加します。

   例

   この例では、Ceph 設定ファイルの名前は、**initial-ceph.conf** です。

```
$ echo "[global]" > initial-ceph.conf
```

3. **[global]** セクションに、**public_network** パラメーターと **cluster_network** パラメーターを追加し、**undercloud.conf** ファイルにリストされているサブネット CIDR をそれぞれに追加します。オーバークラウドノードに対応するサブネット CIDR のみを含めてください。

ヒント

追加するサブネット CIDR は、RHOSP 動的ルーティング用のアンダークラウドのインストールと設定 で説明されているものです。

例

この例では、オーバークラウドノードに対応する **undercloud.conf** ファイル内のサブネットを **public_network** パラメーターと **cluster_network** パラメーターに追加します。

```
[global]
public_network="192.168.1.0/24,192.168.2.0/24,192.168.3.0/24"
cluster_network="192.168.1.0/24,192.168.2.0/24,192.168.3.0/24"
```

4. Ceph をデプロイする準備ができたら、**overcloud ceph deploy** コマンドに必ず **Initial-ceph.conf** を含めます。

例

```
$ openstack overcloud ceph deploy --config initial-ceph.conf \
<other_arguments> \
/home/stack/templates/overcloud-baremetal-deployed.yaml
```

重要

動的ルーティングはまだ利用できません。したがって、Ceph ノードが NTP サーバーに到達するためにルーティングを必要とする場合、Ceph ノードの NTP 設定が遅れる可能性があります。

サイトで NTP サーバーを使用している場合は、**openstack overcloud ceph deploy** コマンドに **--skip-ntp** を追加してください。

Ceph が必要な NTP サーバーに到達できるように BGP ルートが確立されるまでは、Ceph クラスターを実稼働環境に導入しないでください。オーバークラウドがデプロイされ NTP が設定されないと、NTP が設定されていない Ceph クラスターにより、さまざまな異常が発生する可能性があります。たとえば、デーモンが受信したメッセージを無視したり、タイムスタンプが古かったり、メッセージが時間内に受信されなかった場合にトリガーされるタイムアウトが早すぎたり遅すぎたりすることがあります。

5. **overcloud ceph deploy** コマンドの実行により生成された出力ファイルのパスとファイル名をメモします。この例では、**/home/stack/templates/overcloud-baremetal-deployed.yaml** です。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。

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 と共に Red Hat Ceph Storage および Red Hat OpenStack Platform をデプロイする**

4.11. 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 VIP 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 VIP 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。

- **overcloud ceph deploy** コマンドからの出力ファイル。
  例: **overcloud-baremetal-deployed.yaml**。

  詳細は、動的ルーティング環境への Ceph のデプロイ 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/templates/overcloud-baremetal-deployed.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

第5章 RHOSP 動的ルーティングのトラブルシューティング

動的ルーティングを使用する Red Hat OpenStack Platform (RHOSP) 環境で問題を診断する際は、まず適切なログを調べ、VTY シェルを使用して各種 FRRouting コンポーネントに対してクエリーを実行します。

このセクションに含まれるトピックは次のとおりです。

- ML2/OVN BGP エージェントと FRRouting ログ
- BGP のトラブルシューティングに VTY シェルコマンドを使用する

5.1. ML2/OVN BGP エージェントと FRROUTING ログ

Red Hat OpenStack Platform (RHOSP) OVN BGP エージェントは、コンピュートノードと Networker ノードの **/var/log/containers/stdouts/ovn_bgp_agent.log** にログを書き込みます。

BGP、BFD、Zebra デーモンなどの Free Range Routing (FRRouting、または FRR) コンポーネントは、すべての RHOSP ノードの **/var/log/containers/frr/frr.log** にログを書き込みます。

5.2. BGP のトラブルシューティングに VTY シェルコマンドを使用する

Virtual Terminal Interface (VTY シェル) のシェルを使用して、Free Range Routing (FRRouting、または FRR) デーモンと対話できます。Red Hat OpenStack Platform では、**bgpd** などの FRR デーモンがコンテナー内で実行されます。VTY シェルを使用すると、BGP ルーティングの問題のトラブルシューティングに役立ちます。

前提条件

- VTY シェルコマンドを実行するホストで sudo 権限が必要です。

手順

1. BGP デーモン **bgpd** のトラブルシューティングを行うホストにログインします。通常、**bgpd** は、すべてのオーバークラウドノードで実行されます。
2. FRR コンテナーに入ります。

```
$ sudo podman exec -it frr bash
```

3. VTY シェルコマンドを実行するには、次の 2 つのオプションがあります。
   - インタラクティブモード
     sudo vtysh を 1 回入力してインタラクティブモードに入り、複数の VTY シェルコマンドを実行します。

     例

```
$ sudo vtysh
> show bgp summary
```

   - Direct モード
     各 VTY シェルコマンドの前に **sudo vtysh -c** を付けます。

例

```
$ sudo vtysh -c 'show bgp summary'
```

4. 以下に、VTY シェル BGP デーモンのトラブルシューティングコマンドをいくつか示します。

ヒント

IPv6 を使用している場合は、**ip** 引数を省略します。

特定のルーティングテーブルまたはすべてのルーティングテーブルを表示します。

```
> show ip bgp <IPv4_address> | all
> show bgp <IPv6_address> | all
```

BGP ピアにアドバタイズされたスロールート

```
> show ip bgp neighbors <router-ID> <advertised-routes>
```

BGP ピアから受信したスロールート

```
> show ip bgp neighbors <router-ID> <received-routes>
```

関連情報

- **FRRouting ユーザーガイド** の BGP 情報の表示