



Red Hat OpenStack Platform 13

スパイン/リーフ型ネットワーク

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

Red Hat OpenStack Platform 13 スパイン/リーフ型ネットワーク

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

OpenStack Team
rhos-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

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

目次

第1章 はじめに	3
1.1. スパイン/リーフ型ネットワーク	3
1.2. ネットワークトポロジー	3
1.3. スパイン/リーフ型ネットワークの要件	6
1.4. スパイン/リーフの制限事項	6
第2章 アンダークラウドの設定	7
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	7
2.2. DHCP リレーの設定	8
2.3. リーフネットワーク向けのフレーバーの作成とノードのタグ付け	10
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	11
第3章 オーバークラウドの設定	13
3.1. ネットワークデータファイルの作成	13
3.2. カスタム NIC 設定の作成	13
3.3. コントローラー用のカスタム NIC 設定の作成	14
3.4. コンピュート用のカスタム NIC 設定の作成	16
3.5. CEPH STORAGE 用のカスタム NIC 設定の作成	18
3.6. ネットワーク環境ファイルの作成	20
3.7. NIC テンプレートへのネットワークリソースのマッピング	21
3.8. スパイン/リーフのルーティング	21
3.9. コンポーザブルネットワークにルートを割り当てます。	22
3.10. コントロールプレーンのパラメーターの設定	23
3.11. ロールデータファイルの作成	25
3.12. スパイン/リーフ対応のオーバークラウドのデプロイ	27
付録A NETWORK_DATA ファイルの例	29
付録B カスタムの NIC テンプレート	31
付録C ROLES_DATA ファイルの例	34

第1章 はじめに

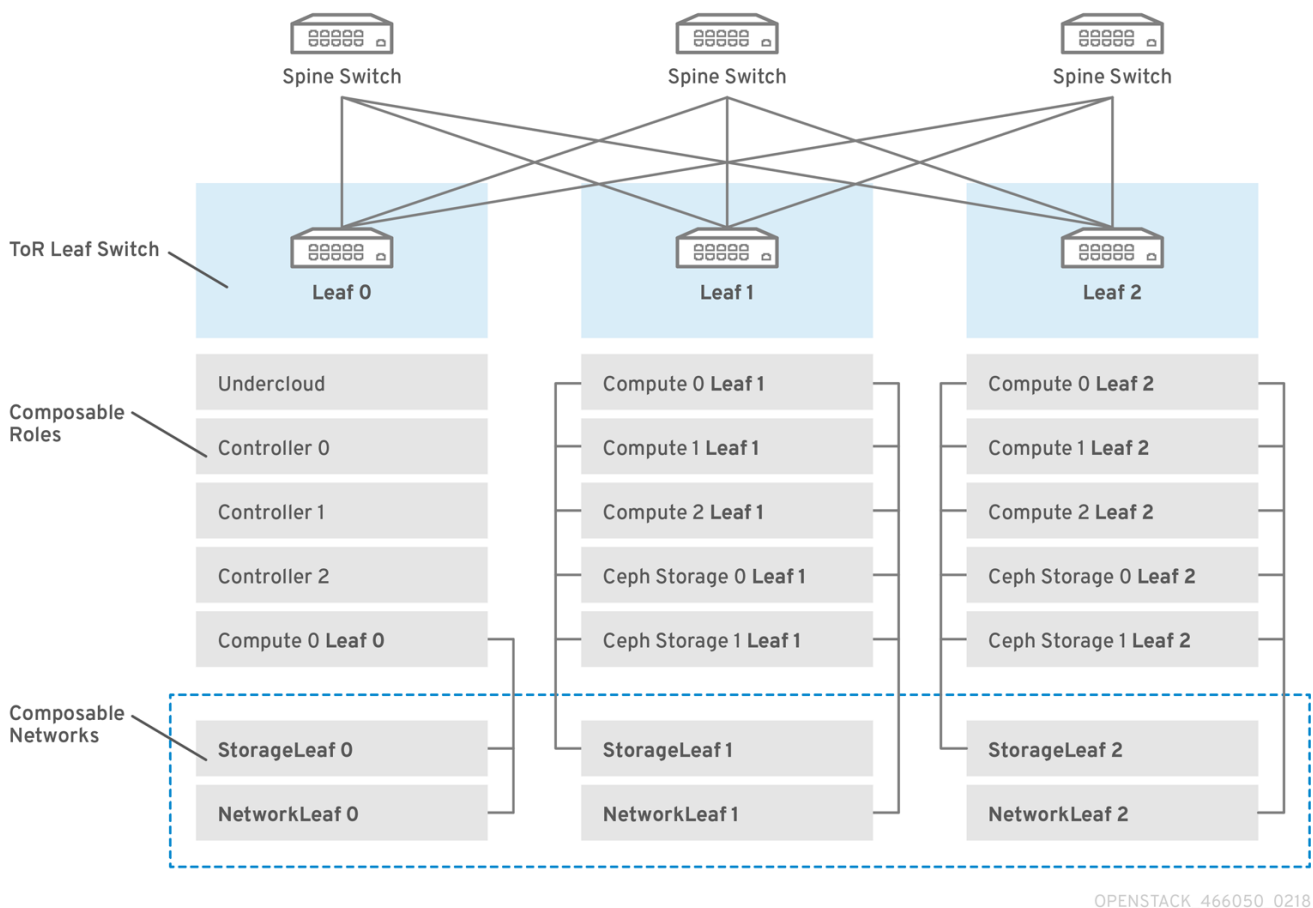
本ガイドは、Red Hat OpenStack Platform 環境に向けたスパイン/リーフ型ネットワークのトポロジーの構築方法についての情報を提供します。これには、完全なエンドツーエンドのシナリオと、お使いの環境でより広範囲なネットワークトポロジーを複製するのに役立つサンプルファイルが含まれます。

1.1. スパイン/リーフ型ネットワーク

Red Hat OpenStack Platform のコンポーザブルネットワークアーキテクチャにより、ネットワークを、一般的なルーティング対応のスパイン/リーフ型データセンタ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、リーフはコンピュートまたはストレージ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リーフ/スパイントポロジーの例」](#)に示したように、通常はデータセンターのラック内にあります。**Leaf 0** ラックにはアンダークラウドノード、コントローラー、コンピュートノードがあります。コンポーザブルネットワークはコンポーザブルロールに割り当てられたノードに提示されます。以下の図では、

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

図1.1 ルーティング対応のリーフ/スパイントポロジーの例



1.2. ネットワークトポロジー

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

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

このシナリオでは、以下のネットワークを使用します。

表1.1 Leaf 0 ネットワーク

ネットワーク	アタッチされているロール	インターフェース	ブリッジ	サブネット
プロビジョニング / コントロールプレーン	すべて	nic1	br-ctlplane (アンダークラウド)	192.168.10.0/24
Storage	Controller	nic2		172.16.0.0/24
Storage Mgmt	Controller	nic3		172.17.0.0/24
Internal API	Controller	nic4		172.18.0.0/24
Tenant	Controller	nic5		172.19.0.0/24
External	Controller	nic6	br-ex	10.1.1.0/24

表1.2 Leaf 1 Networks

ネットワーク	アタッチされているロール	インターフェース	ブリッジ	サブネット
プロビジョニング / コントロールプレーン	すべて	nic1	br-ctlplane (アンダークラウド)	192.168.11.0/24
Storage1	Compute1、Ceph1	nic2		172.16.1.0/24
Storage Mgmt1	Ceph1	nic3		172.17.1.0/24
Internal API1	Compute1	nic4		172.18.1.0/24
Tenant1	Compute1	nic5		172.19.1.0/24

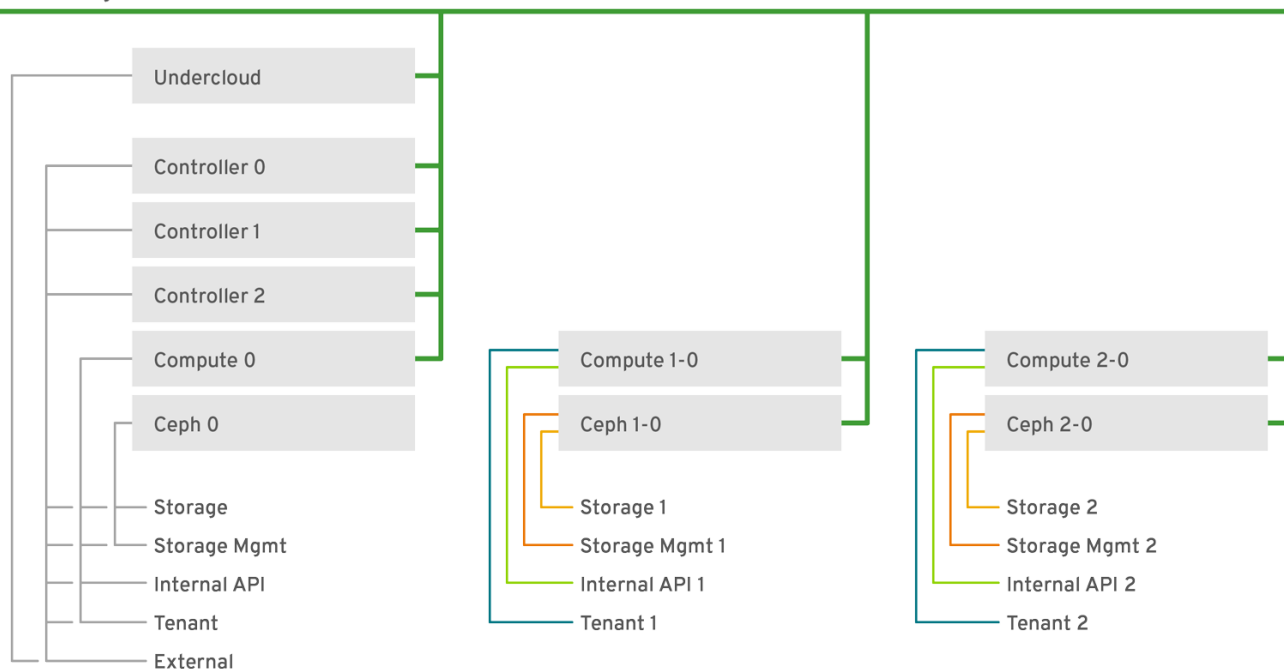
表1.3 Leaf 2 Networks

ネットワーク	アタッチされている ロール	インターフェース	ブリッジ	サブネット
プロビジョニング / コントロールプレーン	すべて	nic1	br-ctlplane (アン ダーククラウド)	192.168.12.0/24
Storage2	Compute2、Ceph2	nic2		172.16.2.0/24
Storage Mgmt2	Ceph2	nic3		172.17.2.0/24
Internal API2	Compute2	nic4		172.18.2.0/24
Tenant2	Compute2	nic5		172.19.2.0/24

表1.4 Supernet Routes

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

Provisioning Network



OPENSTACK_466050_0218

1.3. スパイン/リーフ型ネットワークの要件

レイヤー 3 ルーティング対応アーキテクチャーを使用したネットワーク上でオーバークラウドをデプロイするには、以下の要件を満たす必要があります。

レイヤー 3 ルーティング

異なるレイヤー 2 セグメント間のトラフィックを有効にするには、ネットワークインフラストラクチャーでルーティングを設定する必要があります。これは、静的または動的に設定することができます。

DHCP リレー

アンダークラウドにローカルでない各レイヤー 2 セグメントには、**dhcp-relay** を指定する必要があります。DHCP 要求は、アンダークラウドが接続されているプロビジョニングネットワークのセグメントでアンダークラウドに対して送信する必要があります。

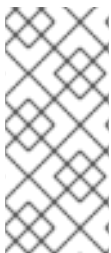


注記

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

1.4. スパイン/リーフの制限事項

- コントローラー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、ノード間のレイヤー 2 ネットワーク接続が必要です。それらのノードはすべて同じリーフ内に配置されます。
- Networker ノードにも同様の制限が適用されます。ネットワークサービスは、Virtual Router Redundancy Protocol (VRRP) を使用するネットワーク内にデフォルトの高可用性パスを実装します。VRRP は仮想ルーターの IP アドレスを使用するので、マスターとバックアップノードを同じ L2 ネットワークセグメントに接続する必要があります。
- テナントまたはプロバイダーネットワークを VLAN セグメンテーションと共に使用する場合には、すべての Networker ノードおよびコンピュータード間で特定の VLAN を共有する必要があります。



注記

ネットワークサービスは、複数の Networker ノードセットで設定することが可能です。各セットはそれらのネットワークのルートを共有し、VRRP が Networker ノードの各セット内の高可用性のデフォルトパスを提供します。このような構成では、ネットワークを共有する全 Networker ノードが同じ L2 ネットワークセグメント上にある必要があります。

第2章 アンダークラウドの設定

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

2.1. スパン/リーフ用のプロビジョニングネットワークの設定

スパン/リーフインフラストラクチャー用のプロビジョニングネットワークを設定するには、**undercloud.conf** ファイルを編集して、以下の手順で定義されているように関連するパラメーターを設定します。

手順

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

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

3. **undercloud.conf** を編集します。
4. **[DEFAULT]** セクションを以下のように編集します。
 - a. **enable_routed_networks** を **true** に設定します。

```
enable_routed_networks = true
```

- b. **subnets** パラメーターでサブネットの一覧を定義します。ルーティング対応のスパン/リーフ内の各レイヤー 2 セグメントにサブネットを 1 つ定義します。

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

- c. **local_subnet** パラメーターでアンダークラウドにローカルな物理レイヤー 2 セグメントに関連付けられるサブネットを指定します。

```
local_subnet = leaf0
```

5. **subnets** パラメーターで定義されている各サブネットごとに新しいセクションを作成します。

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

[leaf1]
cidr = 192.168.11.0/24
dhcp_start = 192.168.11.10
dhcp_end = 192.168.11.90
inspection_iprange = 192.168.11.100,192.168.11.190
```

```
gateway = 192.168.11.1
masquerade = False

[leaf2]
cidr = 192.168.12.0/24
dhcp_start = 192.168.12.10
dhcp_end = 192.168.12.90
inspection_iprange = 192.168.12.100,192.168.12.190
gateway = 192.168.12.1
masquerade = False
```

6. **undercloud.conf** ファイルを保存します。

7. アンダークラウドをインストールするコマンドを実行します。

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

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

アンダークラウドに対する DHCP 要求が適切にリレーされるようにするには、DHCP リレーを設定する必要があります場合があります。次の項では、DHCP リレーの設定方法について説明します。

2.2. DHCP リレーの設定

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

- イントロスペクション用 x 1
- プロビジョニング用 x 1

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

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



注記

特定のデバイス種別での DHCP リレーの設定は、本ガイドの対象範囲外となっています。本ガイドでは参考として、ISC DHCP ソフトウェアの実装を使用した DHCP リレー設定の例を以下に記載しています。この実装の使用方法に関する詳しい情報は、`dhcrelay(8)` の man ページを参照してください。

ブロードキャストを使用する DHCP リレー

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

インターフェース

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

IP ネットワークアドレス

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

ユニキャストを使用する DHCP リレー

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

イントロスペクションに使用されるインターフェースは、**undercloud.conf** の `inspection_interface` で定義されているインターフェースです。



注記

br-ctlplane インターフェースをイントロスペクションに使用するのは一般的です。**undercloud.conf** の `local_ip` で定義されている IP アドレスは、**br-ctlplane** インターフェース上です。

Neutron DHCP 名前空間に割り当てられてる IP アドレスは、**undercloud.conf** の `local_subnet` で設定されている IP 範囲内で利用可能な最初のアドレスです。IP 範囲内の最初のアドレスは、設定の **dhcp_start** で定義されているアドレスです。たとえば、以下の設定が使用されている場合には、**172.20.0.10** がその IP アドレスとなります

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

[leaf0]
cidr = 172.20.0.0/26
dhcp_start = 172.20.0.10
dhcp_end = 172.20.0.19
inspection_iprange = 172.20.0.20,172.20.0.29
gateway = 172.20.0.62
masquerade = False
```



警告

DHCP 名前空間の IP アドレスは自動的に割り当てられます。大半の場合は、IP 範囲の最初のアドレスです。アンダークラウドで以下のコマンドを実行して、確認するようにしてください。

```
$ openstack port list --device-owner network:dhcp -c "Fixed
IP Addresses"
+-----+
+-----+
| Fixed IP Addresses |
+-----+
+-----+
| ip_address='172.20.0.10', subnet_id='7526fbe3-f52a-4b39-
a828-ec59f4ed12b2' |
+-----+
+-----+
$ openstack subnet show 7526fbe3-f52a-4b39-a828-
ec59f4ed12b2 -c name
+-----+-----+
| Field | Value |
+-----+-----+
| name  | leaf0  |
+-----+-----+
```

DHCP リレーの設定例

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

- DHCP の受信要求をリレーするインターフェースは **eth1**、**eth2**、**eth3** です。
- ネットワークセグメント上のアンダークラウドの DHCP サーバーが接続されているインターフェースは **eth0** です。
- イントロスペクションに使用される DHCP サーバーがリッスンしている IP アドレスは **172.20.0.1** 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 **172.20.0.10** です。

これで、**dhcrelay** コマンドは以下のようになります。

```
$ sudo dhcrelay -d --no-pid 172.20.0.10 172.20.0.1 \
-iu eth0 -id eth1 -id eth2 -id eth3
```

これでプロビジョニングネットワークの設定が完了したので、残りのオーバークラウドリーフネットワークを設定することができます。これは、一連の設定ファイルを使用して行います。

2.3. リーフネットワーク向けのフレーバーの作成とノードのタグ付け

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

手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. 各カスタムロール用のフレーバーを作成します。

```
$ ROLES="control0 compute0 compute1 compute2 ceph-storage0 ceph-storage1 ceph-storage2"
$ for ROLE in $ROLES; do openstack flavor create --id auto --ram 4096 --disk 40 --vcpus 1 $ROLE ; done
$ for ROLE in $ROLES; do openstack flavor set --property "cpu_arch"="x86_64" --property "capabilities:boot_option"="local" --property "capabilities:profile"="$ROLE" $ROLE ; done
```

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

```
$ openstack baremetal node set --property capabilities='profile:compute2,boot_option:local' 58c3d07e-24f2-48a7-bbb6-6843f0e8ee13
```

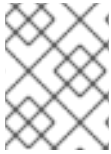
4. フレーバーからロールへのマッピングが含まれた環境ファイル (**~/templates/node-data.yaml**) を作成します。

```
parameter_defaults:
  OvercloudController0Flavor: control0
  OvercloudController0Count: 3
  OvercloudCompute0Flavor: compute0
  OvercloudCompute0Count: 3
  OvercloudCompute1Flavor: compute1
  OvercloudCompute1Count: 3
  OvercloudCompute2Flavor: compute2
  OvercloudCompute2Count: 3
  OvercloudCephStorage0Flavor: ceph-storage0
  OvercloudCephStorage0Count: 3
  OvercloudCephStorage1Flavor: ceph-storage1
  OvercloudCephStorage1Count: 3
  OvercloudCephStorage2Flavor: ceph-storage2
  OvercloudCephStorage2Count: 3
```

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

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

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



注記

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

手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. ベアメタルノードをチェックします。

```
$ openstack baremetal node list
```

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

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

4. baremetal ポートが関連付けられている baremetal ノードを確認します。以下に例を示します。

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

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

```
$ openstack baremetal port set --physical-network ctlplane <port-uuid>
$ openstack baremetal port set --physical-network leaf1 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
$ openstack baremetal port set --physical-network leaf2 <port-uuid>
```

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

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


第3章 オーバークラウドの設定

アンダークラウドの設定が完了したので、残りのオーバークラウドリーフネットワークを設定することができます。これは、一連の設定ファイルを使用して行います。この作業が終わった後には、オーバークラウドをデプロイします。デプロイされる環境には、ルーティングを利用できるネットワークが複数セット実装されます。

3.1. ネットワークデータファイルの作成

リーフネットワークを定義するには、ネットワークデータファイルを作成します。これには、コンポーザブルネットワークとその属性の一覧が YAML 形式で記載されます。デフォルトのネットワークのデータは、アンダークラウドの `/usr/share/openstack-tripleo-heat-templates/network_data.yaml` にあります。

手順

1. **stack** ユーザーのローカルディレクトリーに新しい **network_data_spine_leaf.yaml** ファイルを作成します。
2. **network_data_spine_leaf.yaml** ファイルで、各ネットワークとリーフネットワークをコンポーザブルネットワーク項目として定義する YAML リストを作成します。たとえば、内部 API ネットワークとそのリーフネットワークは、以下の構文を使用して定義します。

```
# Internal API
- name: InternalApi0
  name_lower: internal_api0
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: true
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: true
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]
```



注記

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

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

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

3.2. カスタム NIC 設定の作成

各ロールには、独自の NIC 設定が必要です。NIC テンプレートの基本セットのコピーを作成して、現在の NIC 設定に合わせて変更します。

手順

1. NIC テンプレートを保管する新規ディレクトリーを作成します。以下に例を示します。

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

2. **base.yaml** という名前の基本テンプレートを作成します。「[付録B カスタムの NIC テンプレート](#)」から boilerplate の内容を使用します。このテンプレートは、各ロール用のテンプレートをコピーするベースとして使用します。

リソース

- NIC テンプレートのカスタマイズに関する詳しい情報は、『[オーバークラウドの高度なカスタマイズ](#)』ガイドの「[カスタムのインターフェーステンプレートの作成](#)」を参照してください。

3.3. コントローラー用のカスタム NIC 設定の作成

以下の手順では、Leaf0 上のみでコントローラーノードの YAML 構成を作成します。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. Leaf0 用のベーステンプレート (**base.yaml**) をコピーします。以下に例を示します。

```
$ cp base.yaml controller0.yaml
```

3. **controller0.yaml** のテンプレートを編集して、以下のような内容が記載されているネットワーク設定のセクションまでスクロールします。

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
        params:
          $network_config:
            network_config:
```

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

```

network_config:
- type: interface
  name: nic1
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
  - ip_netmask:
      list_join:
      - /
      - - get_param: ControlPlaneIp
        - get_param: ControlPlane0SubnetCidr
  routes:
  - ip_netmask: 169.254.169.254/32
    next_hop:
      get_param: Leaf0EC2MetadataIp
  - ip_netmask: 192.168.10.0/24
    next_hop:
      get_param: ControlPlane0DefaultRoute

```

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

5. 外部ブリッジの新しいインターフェースを定義します。

```

- type: ovs_bridge
  name: br-ex
  use_dhcp: false
  addresses:
  - ip_netmask:
      get_param: ExternalIpSubnet
  routes:
  - default: true
    next_hop:
      get_param: ExternalInterfaceDefaultRoute
  members:
  - type: interface
    name: nic2
    primary: true

```

members セクションには、使用するネットワーク用の VLAN 設定も含まれます。

6. 各 VLAN を **members** セクションに追加します。たとえば、Storage ネットワークを追加するには、以下のように編集します。

```

members:
- type: interface
  name: nic2
  primary: true
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:

```

```

- ip_netmask:
  get_param: Storage0IpSubnet
routes:
- ip_netmask:
  get_param: StorageSupernet
  next_hop:
    get_param: Storage0InterfaceDefaultRoute

```

各インターフェースの構成には、Leaf0 固有のパラメーター (**Storage0NetworkVlanID**、**Storage0IpSubnet**、**Storage0InterfaceDefaultRoute**) と supernet ルート (**StorageSupernet**) が使用されます。

Storage、**StorageMgmt**、**InternalApi**、**Tenant** のコントローラーネットワークの VLAN 構成を追加します。

7. このファイルを保存します。

3.4. コンピュート用のカスタム NIC 設定の作成

以下の手順では、Leaf0、Leaf1、Leaf2 上のコンピュータノードの YAML 構成を作成します。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. Leaf0 用のベーステンプレート (**base.yaml**) をコピーします。以下に例を示します。

```
$ cp base.yaml compute0.yaml
```

3. **compute0.yaml** のテンプレートを編集して、以下のような内容が記載されているネットワーク設定のセクションまでスクロールします。

```

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:

```

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

```

network_config:
- type: interface
  name: nic1

```

```

use_dhcp: false
dns_servers:
  get_param: DnsServers
addresses:
- ip_netmask:
  list_join:
    - /
    - - get_param: ControlPlaneIp
      - get_param: ControlPlane0SubnetCidr
routes:
- ip_netmask: 169.254.169.254/32
  next_hop:
    get_param: Leaf0EC2MetadataIp
- ip_netmask: 192.168.10.0/24
  next_hop:
    get_param: ControlPlane0DefaultRoute

```

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

5. 外部ブリッジの新しいインターフェースを定義します。

```

- type: ovs_bridge
  name: br-ex
  use_dhcp: false
  members:
  - type: interface
    name: nic2
    primary: true

```

members セクションには、使用するネットワーク用の VLAN 設定も含まれます。

6. 各 VLAN を **members** セクションに追加します。たとえば、Storage ネットワークを追加するには、以下のように編集します。

```

members:
- type: interface
  name: nic2
  primary: true
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
    get_param: Storage0IpSubnet
  routes:
  - ip_netmask:
    get_param: StorageSupernet
    next_hop:
      get_param: Storage0InterfaceDefaultRoute

```

各インターフェースの構成には、Leaf0 固有のパラメーター (**Storage0NetworkVlanID**、**Storage0IpSubnet**、**Storage0InterfaceDefaultRoute**) と supernet ルート (**StorageSupernet**) が使用されます。

Storage、**InternalApi**、**Tenant** のコントローラーネットワークの VLAN 構成を追加します。

7. このファイルを保存します。
8. Leaf1 および Leaf2 で使用するためにこのファイルをコピーします。

```
$ cp compute0.yaml compute1.yaml
$ cp compute0.yaml compute2.yaml
```

9. **compute1.yaml** を編集して **network_config** セクションにスクロールします。Leaf0 パラメーターを Leaf1 パラメーターに置き換えます。これには、**Control Plane**、**Storage**、**InternalApi**、**Tenant** のネットワーク用のパラメーターが含まれます。編集が終わったらファイルを保存してください。
10. **compute2.yaml** を編集して **network_config** セクションにスクロールします。Leaf0 パラメーターを Leaf2 パラメーターに置き換えます。これには、**Control Plane**、**Storage**、**InternalApi**、**Tenant** のネットワーク用のパラメーターが含まれます。編集が終わったらファイルを保存してください。

3.5. CEPH STORAGE 用のカスタム NIC 設定の作成

この手順では、Leaf0、Leaf1、Leaf2 上の Ceph Storage ノードの YAML 構成を作成します。

手順

1. カスタムの NIC ディレクトリーに移動します。

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

2. Leaf0 用のベーステンプレート (**base.yaml**) をコピーします。以下に例を示します。

```
$ cp base.yaml compute0.yaml
```

3. **ceph-storage0.yaml** のテンプレートを編集して、以下のような内容が記載されているネットワーク設定のセクションまでスクロールします。

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
```

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

```
network_config:
- type: interface
  name: nic1
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
  - ip_netmask:
    list_join:
      - /
      - - get_param: ControlPlaneIp
        - get_param: ControlPlane0SubnetCidr
  routes:
  - ip_netmask: 169.254.169.254/32
    next_hop:
      get_param: Leaf0EC2MetadataIp
  - ip_netmask: 192.168.10.0/24
    next_hop:
      get_param: ControlPlane0DefaultRoute
```

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

5. 外部ブリッジの新しいインターフェースを定義します。

```
- type: ovs_bridge
  name: br-ex
  use_dhcp: false
  members:
  - type: interface
    name: nic2
    primary: true
```

members セクションには、使用するネットワーク用の VLAN 設定も含まれます。

6. 各 VLAN を **members** セクションに追加します。たとえば、Storage ネットワークを追加するには、以下のように編集します。

```
members:
- type: interface
  name: nic2
  primary: true
- type: vlan
  vlan_id:
    get_param: Storage0NetworkVlanID
  addresses:
  - ip_netmask:
    get_param: Storage0IpSubnet
  routes:
```

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

各インターフェースの構成には、Leaf0 固有のパラメーター (**Storage0NetworkVlanID**、**Storage0IpSubnet**、**Storage0InterfaceDefaultRoute**) と supernet ルート (**StorageSupernet**) が使用されます。

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

7. このファイルを保存します。

8. Leaf1 および Leaf2 で使用するためにこのファイルをコピーします。

```
$ cp ceph-storage0.yaml ceph-storage1.yaml
$ cp ceph-storage0.yaml ceph-storage2.yaml
```

9. **ceph-storage1.yaml** を編集して **network_config** セクションにスクロールします。Leaf0 パラメーターを Leaf1 パラメーターに置き換えます。これには、**Control Plane**、**Storage**、**InternalApi**、**Tenant** のネットワーク用のパラメーターが含まれます。編集が終わったらファイルを保存してください。

10. **ceph-storage2.yaml** を編集して **network_config** セクションにスクロールします。Leaf0 パラメーターを Leaf2 パラメーターに置き換えます。これには、**Control Plane**、**Storage**、**InternalApi**、**Tenant** のネットワーク用のパラメーターが含まれます。編集が終わったらファイルを保存してください。

3.6. ネットワーク環境ファイルの作成

この手順では、後で使用する基本的なネットワーク環境ファイルを作成します。

手順

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

```
resource_registry:

parameter_defaults:
```

以下の点に注意してください。

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

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

3.7. NIC テンプレートへのネットワークリソースのマッピング

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

手順

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

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

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

```
resource_registry:
  OS::TripleO::Controller0::Net::SoftwareConfig: ./spine-leaf-
    nics/controller0.yaml
  OS::TripleO::Compute0::Net::SoftwareConfig: ./spine-leaf-
    nics/compute0.yaml
  OS::TripleO::Compute1::Net::SoftwareConfig: ./spine-leaf-
    nics/compute1.yaml
  OS::TripleO::Compute2::Net::SoftwareConfig: ./spine-leaf-
    nics/compute2.yaml
  OS::TripleO::CephStorage0::Net::SoftwareConfig: ./spine-leaf-
    nics/CephStorage0.yaml
  OS::TripleO::CephStorage1::Net::SoftwareConfig: ./spine-leaf-
    nics/CephStorage1.yaml
  OS::TripleO::CephStorage2::Net::SoftwareConfig: ./spine-leaf-
    nics/CephStorage2.yaml
```

3. **network-environment.yaml** ファイルを保存します。

3.8. スパイン/リーフのルーティング

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

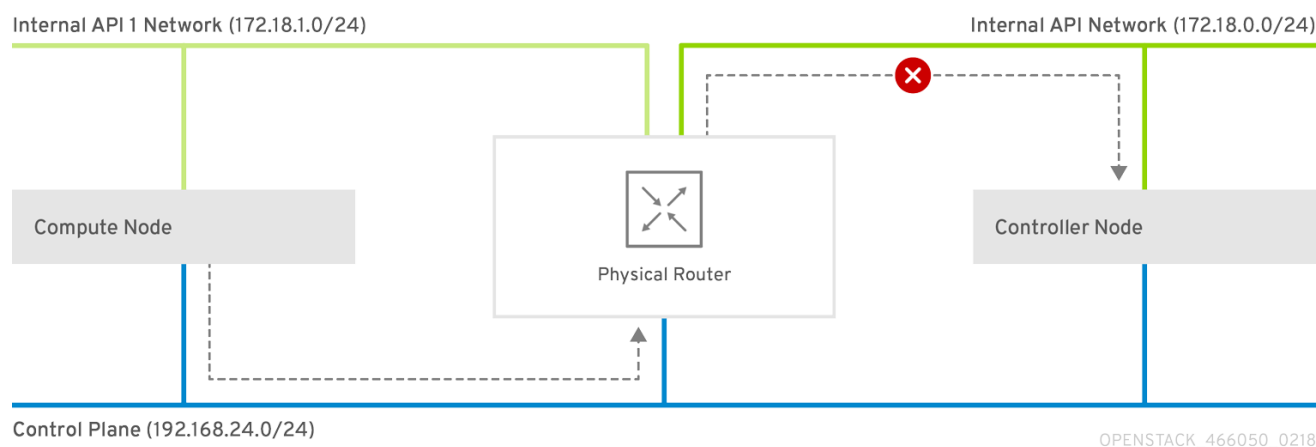
supernet ルートは、各ロール上の全分離ネットワークに適用して、デフォルトのゲートウェイ経由でトラフィックが送信されるのを防ぎます。これは、デフォルトでは、コントローラー以外のロールの場合には **Control Plane** ネットワークで、コントローラー上の場合には **External** ネットワークです。

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

下図には、コントロールプレーンを経由してトラフィックのルーティングを試みて、成功しない例を示

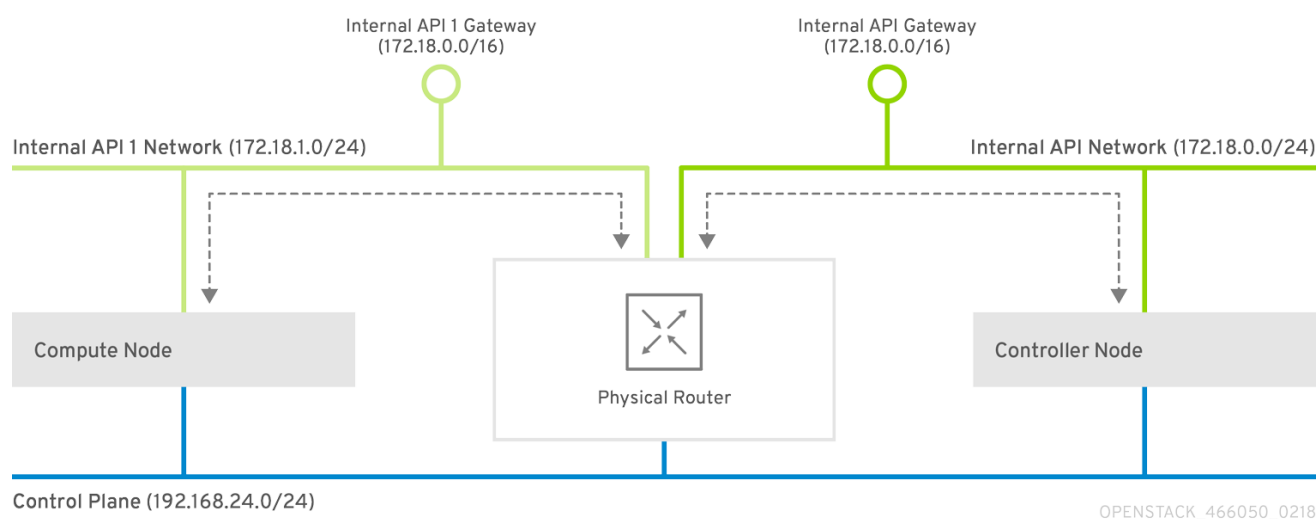
しています。ルーターからコントローラーノードに戻るルートは、VIP がリスンしているインターフェースとは一致しないので、パケットは破棄されます。**192.168.24.0/24** は直接コントローラーに接続され、**Control Plane** に対してローカルであると見なされます。

図3.1 コントロールプレーン経由のトラフィックルーティング



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

図3.2 Internal API 経由のトラフィックルーティング



3.9. コンポーザブルネットワークにルートを割り当てます。

この手順では、リーフネットワークのルーティングを定義します。

手順

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

```
parameter_defaults:
  StorageSupernet: 172.16.0.0/16
  StorageMgmtSupernet: 172.17.0.0/16
```

InternalApiSupernet: 172.18.0.0/16
 TenantSupernet: 172.19.0.0/16

注記

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

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

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

```
parameter_defaults:
  ...
  Compute1ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api1')}}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant1')}}"
  Compute2ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api2')}}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant2')}}"
  Compute3ExtraConfig:
    nova::vncproxy::host: "%{hiera('internal_api3')}}"
    neutron::agents::ml2::ovs::local_ip: "%{hiera('tenant3')}}"
  CephAnsibleExtraConfig:
    public_network: '172.16.0.0/24,172.16.1.0/24,172.16.2.0/24'
    cluster_network: '172.17.0.0/24,172.17.1.0/24,172.17.2.0/24'
```

- コンピュートの **ExtraConfig** パラメーターの場合:
 - VNC プロキシに使用する IP アドレスを定義します。
 - ML2 エージェントに使用する IP アドレスを定義します。
- **CephAnsibleExtraConfig** の場合:
 - **public_network** の設定には、すべてのストレージネットワーク (1 リーフにつき 1 つ) がリストされます。
 - **cluster_network** のエントリーにはストレージ管理ネットワーク (1 リーフにつき 1 つ) がリストされます。

3.10. コントロールプレーンのパラメーターの設定

分離されたスパイン/リーフネットワークのネットワーク詳細の定義には通常 **network_data** ファイルを使用します。コントロールプレーンネットワークは例外で、これはアンダークラウドによって作成されます。ただし、オーバークラウドには、各リーフのコントロールプレーンへのアクセスが必要です。これには、追加のパラメーターをいくつか設定する必要がありますが、**network-environment.yaml** ファイルで定義します。たとえば、以下のスニペットは、Leaf0 上のコントローラーノード用の NIC テンプレートのサンプルの抜粋です。

```
- type: interface
  name: nic1
  use_dhcp: false
  dns_servers:
    get_param: DnsServers
  addresses:
    - ip_netmask:
        list_join:
          - /
          - - get_param: ControlPlaneIp
            - get_param: ControlPlane0SubnetCidr
  routes:
    - ip_netmask: 169.254.169.254/32
      next_hop:
        get_param: Leaf0EC2MetadataIp
    - ip_netmask: 192.168.10.0/24
      next_hop:
        get_param: ControlPlane0DefaultRoute
```

この場合は、Leaf 0 上のコントロールプレーンネットワークに対応するIP、サブネット、メタデータIP、デフォルトルートを定義する必要があります。

手順

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

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

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

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

- c. 各リーフにコントロールプレーンのルートを追加します。

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

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

d. EC2 メタデータ IP 用のパラメーターの追加

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

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

3. **network-environment.yaml** ファイルを保存します。

3.11. ロールデータファイルの作成

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

手順

1. **stack** ユーザーのローカルディレクトリー内にカスタム **roles** のディレクトリーを作成します。

```
$ mkdir ~/roles
```

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

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

3. **Controller0.yaml** ファイルを編集します。

```
$ vi ~/roles/Controller0.yaml
```

4. このファイルで **name**、**networks**、**HostnameFormatDefault** のパラメーターを編集して、Leaf 0 固有のパラメーターと一致するようにします。以下に例を示します。

```
- name: Controller0
...
networks:
  - External
  - InternalApi0
  - Storage0
  - StorageMgmt0
  - Tenant0
...
HostnameFormatDefault: '%stackname%-controller0-%index%'
```

このファイルを保存します。

5. **Compute0.yaml** ファイルを編集します。

```
$ vi ~/roles/Compute0.yaml
```

6. このファイルで **name**、**networks**、**HostnameFormatDefault** のパラメーターを編集して、Leaf 0 固有のパラメーターと一致するようにします。以下に例を示します。

```
- name: Compute0
...
networks:
  - InternalApi0
  - Tenant0
  - Storage0
HostnameFormatDefault: '%stackname%-compute0-%index%'
```

このファイルを保存します。

7. **CephStorage0.yaml** ファイルを編集します。

```
$ vi ~/roles/CephStorage0.yaml
```

8. このファイルで **name** および **networks** のパラメーターを編集して、その値が Leaf 0 固有のパラメーターと一致するようにします。また、**HostnameFormatDefault** パラメーターを追加して、Ceph Storage ノード用の Leaf 0 のホスト名を定義します。以下の例を示します。

```
- name: CephStorage0
...
networks:
  - Storage0
  - StorageMgmt0
HostnameFormatDefault: '%stackname%-cephstorage0-%index%'
```

このファイルを保存します。

9. Leaf 0 の Compute と Ceph Storage のファイルをコピーして、Leaf 1 および Leaf 2 のファイルのベースにします。

```
$ 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
```

10. Leaf 1 および Leaf 2 のファイルで **name**、**networks**、**HostnameFormatDefault** のパラメーターを編集して、対応するリーフネットワークのパラメーターに一致するようにします。たとえば、Leaf 1 Compute ファイルには以下の値を使用します。

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

Leaf 1 Ceph Storage パラメーターには以下の値を使用します。

```
- name: CephStorage1
...
networks:
  - Storage1
  - StorageMgmt1
HostnameFormatDefault: '%stackname%-cephstorage1-%index%'
```

11. ロールの準備が整ったら、以下のコマンドで完全なロールデータファイルを生成します。

```
$ openstack overcloud roles generate --roles-path ~/roles -o
roles_data_spine_leaf.yaml Controller0 Compute0 Compute1 Compute2
CephStorage0 CephStorage1 CephStorage2
```

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

このファイルの完全なサンプルは、「[付録C roles_data ファイルの例](#)」に記載しています。

3.12. スパイン/リーフ対応のオーバークラウドのデプロイ

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

手順

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



注記

ネットワークサブネットと **allocation_pools** の値に対して実行される検証は現在ありません。これらの値が一致するように定義して、既存のネットワークと競合が発生しないことを確認してください。

2. `~/templates/spine-leaf-nics/` に含まれている NIC テンプレートをチェックして、各リーフ上の各ロールのインターフェースが正しく定義されていることを確認します。
3. `network-environment.yaml` 環境ファイルをチェックして、ネットワークデータファイルでは制御できない全カスタムパラメーターが定義されていることを確認します。これには、ルート、コントロールプレーンのパラメーター、各ロールのカスタム NIC テンプレートを参照する `resource_registry` セクションが含まれます。
4. `/home/stack/templates/roles_data_spine_leaf.yaml` の値をチェックして、各リーフにロールが定義されていることを確認します。
5. `/home/stack/templates/nodes_data.yaml` ファイルをチェックして、全ロールにフレーバーとノード数が割り当てられていることを確認します。また、各リーフの全ノードが正しくタグ付けされていることも確認してください。
6. `openstack overcloud deploy` コマンドを実行して、スパイン/リーフの設定を適用します。以下に例を示します。

```
openstack overcloud deploy --templates \  
-n /home/stack/template/network_data_spine_leaf.yaml \  
-r /home/stack/templates/roles_data_spine_leaf.yaml \  
-e /usr/share/openstack-tripleo-heat-templates/environments/network-  
isolation.yaml \  
-e /home/stack/templates/network-environment.yaml \  
-e /home/stack/templates/nodes_data.yaml \  
-e [OTHER ENVIRONMENT FILES]
```

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

7. スパイン/リーフ対応のオーバークラウドがデプロイされるまで待ちます。

付録A NETWORK_DATA ファイルの例

```
# Storage
- name: Storage0
  vip: true
  name_lower: storage0
  ip_subnet: '172.16.0.0/24'
  allocation_pools: [{'start': '172.16.0.4', 'end': '172.16.0.250'}]
- name: Storage1
  vip: true
  name_lower: storage1
  ip_subnet: '172.16.1.0/24'
  allocation_pools: [{'start': '172.16.1.4', 'end': '172.16.1.250'}]
- name: Storage2
  vip: true
  name_lower: storage2
  ip_subnet: '172.16.2.0/24'
  allocation_pools: [{'start': '172.16.2.4', 'end': '172.16.2.250'}]

# StorageMgmt
- name: StorageMgmt0
  name_lower: storage_mgmt0
  vip: true
  ip_subnet: '172.17.0.0/24'
  allocation_pools: [{'start': '172.17.0.0', 'end': '172.17.0.250'}]
- name: StorageMgmt1
  name_lower: storage_mgmt1
  vip: true
  ip_subnet: '172.17.1.0/24'
  allocation_pools: [{'start': '172.17.1.4', 'end': '172.17.1.250'}]
- name: StorageMgmt2
  name_lower: storage_mgmt2
  vip: true
  ip_subnet: '172.17.2.0/24'
  allocation_pools: [{'start': '172.17.2.4', 'end': '172.17.2.250'}]

# Internal API
- name: InternalApi0
  name_lower: internal_api0
  vip: true
  ip_subnet: '172.18.0.0/24'
  allocation_pools: [{'start': '172.18.0.4', 'end': '172.18.0.250'}]
- name: InternalApi1
  name_lower: internal_api1
  vip: true
  ip_subnet: '172.18.1.0/24'
  allocation_pools: [{'start': '172.18.1.4', 'end': '172.18.1.250'}]
- name: InternalApi2
  name_lower: internal_api2
  vip: true
  ip_subnet: '172.18.2.0/24'
  allocation_pools: [{'start': '172.18.2.4', 'end': '172.18.2.250'}]

# Tenant
- name: Tenant0
  vip: false # Tenant network does not use VIPs
```

```
    name_lower: tenant0
    ip_subnet: '172.19.0.0/24'
    allocation_pools: [{'start': '172.19.0.4', 'end': '172.19.0.250'}]
- name: Tenant1
  vip: false # Tenant network does not use VIPs
  name_lower: tenant1
  ip_subnet: '172.19.1.0/24'
  allocation_pools: [{'start': '172.19.1.4', 'end': '172.19.1.250'}]
- name: Tenant2
  vip: false # Tenant network does not use VIPs
  name_lower: tenant2
  ip_subnet: '172.19.2.0/24'
  allocation_pools: [{'start': '172.19.2.4', 'end': '172.19.2.250'}]

- name: External
  vip: true
  name_lower: external
  ip_subnet: '10.0.0.0/24'
  allocation_pools: [{'start': '10.0.0.4', 'end': '10.0.0.250'}]
  gateway_ip: '10.0.0.1'
```

付録B カスタムの NIC テンプレート

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

```
heat_template_version: queens

parameters:
  # Supernets
  StorageSupernet:
    type: string
  StorageMgmtSupernet:
    type: string
  InternalApiSupernet:
    type: string
  TenantSupernet:
    type: string
  ExternalSupernet:
    type: string

  # Default Routes
  ControlPlane0DefaultRoute:
    type: string
  ControlPlane1DefaultRoute:
    type: string
  ControlPlane2DefaultRoute:
    type: string
  Storage0InterfaceDefaultRoute:
    type: string
  Storage1InterfaceDefaultRoute:
    type: string
  Storage2InterfaceDefaultRoute:
    type: string
  StorageMgmt0InterfaceDefaultRoute:
    type: string
  StorageMgmt1InterfaceDefaultRoute:
    type: string
  StorageMgmt2InterfaceDefaultRoute:
    type: string
  InternalApi0InterfaceDefaultRoute:
    type: string
  InternalApi1InterfaceDefaultRoute:
    type: string
  InternalApi2InterfaceDefaultRoute:
    type: string
  Tenant0InterfaceDefaultRoute:
    type: string
  Tenant1InterfaceDefaultRoute:
    type: string
  Tenant2InterfaceDefaultRoute:
    type: string
  ExternalInterfaceDefaultRoute:
    type: string

  # IP subnets
```

```
Storage0IpSubnet:
  type: string
Storage1IpSubnet:
  type: string
Storage2IpSubnet:
  type: string
StorageMgmt0IpSubnet:
  type: string
StorageMgmt1IpSubnet:
  type: string
StorageMgmt2IpSubnet:
  type: string
InternalApi0IpSubnet:
  type: string
InternalApi1IpSubnet:
  type: string
InternalApi2IpSubnet:
  type: string
Tenant0IpSubnet:
  type: string
Tenant1IpSubnet:
  type: string
Tenant2IpSubnet:
  type: string
ExternalIpSubnet:
  type: string
ManagementIpSubnet:
  type: string

# VLAN IDs
Storage0NetworkVlanID:
  type: number
Storage1NetworkVlanID:
  type: number
Storage2NetworkVlanID:
  type: number
StorageMgmt0NetworkVlanID:
  type: number
StorageMgmt1NetworkVlanID:
  type: number
StorageMgmt2NetworkVlanID:
  type: number
InternalApi0NetworkVlanID:
  type: number
InternalApi1NetworkVlanID:
  type: number
InternalApi1NetworkVlanID:
  type: number
Tenant0NetworkVlanID:
  type: number
Tenant1NetworkVlanID:
  type: number
Tenant2NetworkVlanID:
  type: number
ExternalNetworkVlanID:
  type: number
```

```
ManagementNetworkVlanID:
  type: number

# Subnet CIDR
ControlPlane0SubnetCidr:
  type: string
ControlPlane1SubnetCidr:
  type: string
ControlPlane1SubnetCidr:
  type: string

ControlPlaneIp:
  type: string
DnsServers:
  type: comma_delimited_list

# EC2 metadata server IPs
Leaf0EC2MetadataIp:
  type: string
Leaf1EC2MetadataIp:
  type: string
Leaf2EC2MetadataIp:
  type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-
templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                [NETWORK CONFIG HERE]

outputs:
  OS::stack_id:
    description: The OsNetConfigImpl resource.
    value:
      get_resource: OsNetConfigImpl
```

付録C ROLES_DATA ファイルの例

```
#####
####
# Role: Controller0
#
#####
####
- name: Controller0
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
    - primary
    - controller
  networks:
    - External
    - InternalApi0
    - Storage0
    - StorageMgmt0
    - Tenant0
  default_route_networks: ['External']
  HostnameFormatDefault: '%stackname%-controller0-%index%'
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  deprecated_nic_config_name: 'controller.yaml'
  ServicesDefault:
    - OS::Triple0::Services::Aide
    - OS::Triple0::Services::AodhApi
    - OS::Triple0::Services::AodhEvaluator
    - OS::Triple0::Services::AodhListener
    - OS::Triple0::Services::AodhNotifier
    - OS::Triple0::Services::AuditD
    - OS::Triple0::Services::BarbicanApi
    - OS::Triple0::Services::BarbicanBackendSimpleCrypto
    - OS::Triple0::Services::BarbicanBackendDogtag
    - OS::Triple0::Services::BarbicanBackendKmip
    - OS::Triple0::Services::BarbicanBackendPkcs11Crypto
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CeilometerApi
    - OS::Triple0::Services::CeilometerCollector
    - OS::Triple0::Services::CeilometerExpirer
    - OS::Triple0::Services::CeilometerAgentCentral
    - OS::Triple0::Services::CeilometerAgentNotification
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::CephMds
    - OS::Triple0::Services::CephMgr
    - OS::Triple0::Services::CephMon
    - OS::Triple0::Services::CephRbdMirror
    - OS::Triple0::Services::CephRgw
    - OS::Triple0::Services::CertmongerUser
    - OS::Triple0::Services::CinderApi
    - OS::Triple0::Services::CinderBackendDellPs
```

- OS::Triple0::Services::CinderBackendDellSc
- OS::Triple0::Services::CinderBackendDellEMCUnity
- OS::Triple0::Services::CinderBackendDellEMCVMAXISCSI
- OS::Triple0::Services::CinderBackendDellEMCVNX
- OS::Triple0::Services::CinderBackendDellEMCXTREMIOSCSI
- OS::Triple0::Services::CinderBackendNetApp
- OS::Triple0::Services::CinderBackendScaleIO
- OS::Triple0::Services::CinderBackendVRTSHyperScale
- OS::Triple0::Services::CinderBackup
- OS::Triple0::Services::CinderHPELeftHandISCSI
- OS::Triple0::Services::CinderScheduler
- OS::Triple0::Services::CinderVolume
- OS::Triple0::Services::Clustercheck
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::Congress
- OS::Triple0::Services::Docker
- OS::Triple0::Services::Ec2Api
- OS::Triple0::Services::Etc
- OS::Triple0::Services::ExternalSwiftProxy
- OS::Triple0::Services::Fluentd
- OS::Triple0::Services::GlanceApi
- OS::Triple0::Services::GlanceRegistry
- OS::Triple0::Services::GnocchiApi
- OS::Triple0::Services::GnocchiMetricd
- OS::Triple0::Services::GnocchiStatsd
- OS::Triple0::Services::HAproxy
- OS::Triple0::Services::HeatApi
- OS::Triple0::Services::HeatApiCloudwatch
- OS::Triple0::Services::HeatApiCfn
- OS::Triple0::Services::HeatEngine
- OS::Triple0::Services::Horizon
- OS::Triple0::Services::Ipsec
- OS::Triple0::Services::IronicApi
- OS::Triple0::Services::IronicConductor
- OS::Triple0::Services::IronicPxe
- OS::Triple0::Services::Isccsid
- OS::Triple0::Services::Keepalived
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::Keystone
- OS::Triple0::Services::LoginDefs
- OS::Triple0::Services::ManilaApi
- OS::Triple0::Services::ManilaBackendCephFs
- OS::Triple0::Services::ManilaBackendIsilon
- OS::Triple0::Services::ManilaBackendNetapp
- OS::Triple0::Services::ManilaBackendUnity
- OS::Triple0::Services::ManilaBackendVNX
- OS::Triple0::Services::ManilaBackendVMAX
- OS::Triple0::Services::ManilaScheduler
- OS::Triple0::Services::ManilaShare
- OS::Triple0::Services::Memcached
- OS::Triple0::Services::MistralApi
- OS::Triple0::Services::MistralEngine
- OS::Triple0::Services::MistralExecutor
- OS::Triple0::Services::MistralEventEngine
- OS::Triple0::Services::MongoDb
- OS::Triple0::Services::MySQL

- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::NeutronApi
- OS::Triple0::Services::NeutronBgpVpnApi
- OS::Triple0::Services::NeutronSfcApi
- OS::Triple0::Services::NeutronCorePlugin
- OS::Triple0::Services::NeutronDhcpAgent
- OS::Triple0::Services::NeutronL2gwAgent
- OS::Triple0::Services::NeutronL2gwApi
- OS::Triple0::Services::NeutronL3Agent
- OS::Triple0::Services::NeutronLbaasv2Agent
- OS::Triple0::Services::NeutronLbaasv2Api
- OS::Triple0::Services::NeutronLinuxbridgeAgent
- OS::Triple0::Services::NeutronMetadataAgent
- OS::Triple0::Services::NeutronML2FujitsuCfab
- OS::Triple0::Services::NeutronML2FujitsuFossw
- OS::Triple0::Services::NeutronOvsAgent
- OS::Triple0::Services::NeutronVppAgent
- OS::Triple0::Services::NovaApi
- OS::Triple0::Services::NovaConductor
- OS::Triple0::Services::NovaConsoleauth
- OS::Triple0::Services::NovaIronic
- OS::Triple0::Services::NovaMetadata
- OS::Triple0::Services::NovaPlacement
- OS::Triple0::Services::NovaScheduler
- OS::Triple0::Services::NovaVncProxy
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::ContainersLogrotateCron
- OS::Triple0::Services::OctaviaApi
- OS::Triple0::Services::OctaviaDeploymentConfig
- OS::Triple0::Services::OctaviaHealthManager
- OS::Triple0::Services::OctaviaHousekeeping
- OS::Triple0::Services::OctaviaWorker
- OS::Triple0::Services::OpenDaylightApi
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::OVNDBs
- OS::Triple0::Services::OVNController
- OS::Triple0::Services::Pacemaker
- OS::Triple0::Services::PankoApi
- OS::Triple0::Services::RabbitMQ
- OS::Triple0::Services::Redis
- OS::Triple0::Services::Rhsm
- OS::Triple0::Services::RsyslogSidecar
- OS::Triple0::Services::SaharaApi
- OS::Triple0::Services::SaharaEngine
- OS::Triple0::Services::Securetty
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::SkydiveAgent
- OS::Triple0::Services::SkydiveAnalyzer
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::SwiftProxy
- OS::Triple0::Services::SwiftDispersion
- OS::Triple0::Services::SwiftRingBuilder
- OS::Triple0::Services::SwiftStorage
- OS::Triple0::Services::Tacker
- OS::Triple0::Services::Timezone


```

- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::Tuned
- OS::Triple0::Services::Vpp
- OS::Triple0::Services::Zaqaar
- OS::Triple0::Services::Ptp
#####
#####
# Role: Compute0
#
#####
#####
- name: Compute0
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi0
    - Tenant0
    - Storage0
  HostnameFormatDefault: '%stackname%-compute0-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::Aide
    - OS::Triple0::Services::AuditD
    - OS::Triple0::Services::CACerts
    - OS::Triple0::Services::CephClient
    - OS::Triple0::Services::CephExternal
    - OS::Triple0::Services::CertmongerUser
    - OS::Triple0::Services::Collectd
    - OS::Triple0::Services::ComputeCeilometerAgent
    - OS::Triple0::Services::ComputeNeutronCorePlugin
    - OS::Triple0::Services::ComputeNeutronL3Agent
    - OS::Triple0::Services::ComputeNeutronMetadataAgent
    - OS::Triple0::Services::ComputeNeutronOvsAgent
    - OS::Triple0::Services::Docker
    - OS::Triple0::Services::Fluentd
    - OS::Triple0::Services::Ipsec
    - OS::Triple0::Services::Iscsid
    - OS::Triple0::Services::Kernel
    - OS::Triple0::Services::LoginDefs
    - OS::Triple0::Services::MySQLClient
    - OS::Triple0::Services::NeutronBgpVpnBagpipe
    - OS::Triple0::Services::NeutronLinuxbridgeAgent
    - OS::Triple0::Services::NeutronVppAgent
    - OS::Triple0::Services::NovaCompute
    - OS::Triple0::Services::NovaLibvirt
    - OS::Triple0::Services::NovaMigrationTarget

```

```

- OS::TripleO::Services::Ntp
- OS::TripleO::Services::ContainersLogrotateCron
- OS::TripleO::Services::OpenDaylightOvs
- OS::TripleO::Services::Rhsm
- OS::TripleO::Services::RsyslogSidecar
- OS::TripleO::Services::Securetty
- OS::TripleO::Services::SensuClient
- OS::TripleO::Services::SkydiveAgent
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Sshd
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::Tuned
- OS::TripleO::Services::Vpp
- OS::TripleO::Services::OVNController
- OS::TripleO::Services::OVNMetadataAgent
- OS::TripleO::Services::Ptp
#####
#####
# Role: Compute1
#
#####
#####
- name: Compute1
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi1
    - Tenant1
    - Storage1
  HostnameFormatDefault: '%stackname%-compute1-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephClient
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::ComputeCeilometerAgent
    - OS::TripleO::Services::ComputeNeutronCorePlugin
    - OS::TripleO::Services::ComputeNeutronL3Agent
    - OS::TripleO::Services::ComputeNeutronMetadataAgent
    - OS::TripleO::Services::ComputeNeutronOvsAgent
    - OS::TripleO::Services::Docker

```

```

- OS::Triple0::Services::Fluentd
- OS::Triple0::Services::Ipsec
- OS::Triple0::Services::Isccsid
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::LoginDefs
- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::NeutronBgpVpnBagpipe
- OS::Triple0::Services::NeutronLinuxbridgeAgent
- OS::Triple0::Services::NeutronVppAgent
- OS::Triple0::Services::NovaCompute
- OS::Triple0::Services::NovaLibvirt
- OS::Triple0::Services::NovaMigrationTarget
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::ContainersLogrotateCron
- OS::Triple0::Services::OpenDaylightOvs
- OS::Triple0::Services::Rhsm
- OS::Triple0::Services::RsyslogSidecar
- OS::Triple0::Services::Securetty
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::SkydiveAgent
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::Tuned
- OS::Triple0::Services::Vpp
- OS::Triple0::Services::OVNController
- OS::Triple0::Services::OVNMetadataAgent
- OS::Triple0::Services::Ptp
#####
#####
# Role: Compute2
#
#####
#####
- name: Compute2
  description: |
    Basic Compute Node role
  CountDefault: 1
  networks:
    - InternalApi2
    - Tenant2
    - Storage2
  HostnameFormatDefault: '%stackname%-compute2-%index%'
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  deprecated_nic_config_name: 'compute.yaml'
  disable_upgrade_deployment: True
  ServicesDefault:
    - OS::Triple0::Services::Aide

```

```

- OS::TripleO::Services::AuditD
- OS::TripleO::Services::CACerts
- OS::TripleO::Services::CephClient
- OS::TripleO::Services::CephExternal
- OS::TripleO::Services::CertmongerUser
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::ComputeCeilometerAgent
- OS::TripleO::Services::ComputeNeutronCorePlugin
- OS::TripleO::Services::ComputeNeutronL3Agent
- OS::TripleO::Services::ComputeNeutronMetadataAgent
- OS::TripleO::Services::ComputeNeutronOvsAgent
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Fluentd
- OS::TripleO::Services::Ipsec
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Kernel
- OS::TripleO::Services::LoginDefs
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronBgpVpnBagpipe
- OS::TripleO::Services::NeutronLinuxbridgeAgent
- OS::TripleO::Services::NeutronVppAgent
- OS::TripleO::Services::NovaCompute
- OS::TripleO::Services::NovaLibvirt
- OS::TripleO::Services::NovaMigrationTarget
- OS::TripleO::Services::Ntp
- OS::TripleO::Services::ContainersLogrotateCronD
- OS::TripleO::Services::OpenDaylightOvs
- OS::TripleO::Services::Rhsm
- OS::TripleO::Services::RsyslogSidecar
- OS::TripleO::Services::Securetty
- OS::TripleO::Services::SensuClient
- OS::TripleO::Services::SkydiveAgent
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Sshd
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::Tuned
- OS::TripleO::Services::Vpp
- OS::TripleO::Services::OVNController
- OS::TripleO::Services::OVNMetadataAgent
- OS::TripleO::Services::Ptp

```

```

#####
#####

```

```
# Role: CephStorage0
```

```
#
```

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

```

```
- name: CephStorage0
```

```
  description: |
```

```
    Ceph OSD Storage node role
```

```
  networks:
```

```
    - Storage0
```

```
    - StorageMgmt0
```

```
  HostnameFormatDefault: '%stackname%-cephstorage0-%index%'
```

```
  uses_deprecated_params: False
```

```
deprecated_nic_config_name: 'ceph-storage.yaml'
```

```
ServicesDefault:
```

- OS::Triple0::Services::Aide
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::CACerts
- OS::Triple0::Services::CephOSD
- OS::Triple0::Services::CertmongerUser
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::Docker
- OS::Triple0::Services::Fluentd
- OS::Triple0::Services::Ipsec
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::LoginDefs
- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::Ntp
- OS::Triple0::Services::ContainersLogrotateCron
- OS::Triple0::Services::Rhsm
- OS::Triple0::Services::RsyslogSidecar
- OS::Triple0::Services::Securetty
- OS::Triple0::Services::SensuClient
- OS::Triple0::Services::Snmp
- OS::Triple0::Services::Sshd
- OS::Triple0::Services::Timezone
- OS::Triple0::Services::TripleoFirewall
- OS::Triple0::Services::TripleoPackages
- OS::Triple0::Services::Tuned
- OS::Triple0::Services::Ptp

```
#####
```

```
#####
```

```
# Role: CephStorage1
```

```
#
```

```
#####
```

```
#####
```

```
- name: CephStorage1
```

```
description: |
```

```
    Ceph OSD Storage node role
```

```
networks:
```

- Storage1
- StorageMgmt1

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

```
uses_deprecated_params: False
```

```
deprecated_nic_config_name: 'ceph-storage.yaml'
```

```
ServicesDefault:
```

- OS::Triple0::Services::Aide
- OS::Triple0::Services::AuditD
- OS::Triple0::Services::CACerts
- OS::Triple0::Services::CephOSD
- OS::Triple0::Services::CertmongerUser
- OS::Triple0::Services::Collectd
- OS::Triple0::Services::Docker
- OS::Triple0::Services::Fluentd
- OS::Triple0::Services::Ipsec
- OS::Triple0::Services::Kernel
- OS::Triple0::Services::LoginDefs
- OS::Triple0::Services::MySQLClient
- OS::Triple0::Services::Ntp

```

- OS::TripleO::Services::ContainersLogrotateCron
- OS::TripleO::Services::Rhsm
- OS::TripleO::Services::RsyslogSidecar
- OS::TripleO::Services::Securetty
- OS::TripleO::Services::SensuClient
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Sshd
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::Tuned
- OS::TripleO::Services::Ptp
#####
#####
# Role: CephStorage2
#
#####
#####
- name: CephStorage2
  description: |
    Ceph OSD Storage node role
  networks:
    - Storage2
    - StorageMgmt2
  HostnameFormatDefault: '%stackname%-cephstorage2-%index%'
  uses_deprecated_params: False
  deprecated_nic_config_name: 'ceph-storage.yaml'
  ServicesDefault:
    - OS::TripleO::Services::Aide
    - OS::TripleO::Services::AuditD
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephOSD
    - OS::TripleO::Services::CertmongerUser
    - OS::TripleO::Services::Collectd
    - OS::TripleO::Services::Docker
    - OS::TripleO::Services::Fluentd
    - OS::TripleO::Services::Ipsec
    - OS::TripleO::Services::Kernel
    - OS::TripleO::Services::LoginDefs
    - OS::TripleO::Services::MySQLClient
    - OS::TripleO::Services::Ntp
    - OS::TripleO::Services::ContainersLogrotateCron
    - OS::TripleO::Services::Rhsm
    - OS::TripleO::Services::RsyslogSidecar
    - OS::TripleO::Services::Securetty
    - OS::TripleO::Services::SensuClient
    - OS::TripleO::Services::Snmp
    - OS::TripleO::Services::Sshd
    - OS::TripleO::Services::Timezone
    - OS::TripleO::Services::TripleoFirewall
    - OS::TripleO::Services::TripleoPackages
    - OS::TripleO::Services::Tuned
    - OS::TripleO::Services::Ptp

```

