Red Hat OpenStack Platform 16.2

ベアメタルプロビジョニング

Bare Metal サービス (ironic) のインストール、設定、および使用方法

Last Updated: 2022-04-23

Red Hat OpenStack Platform 16.2 ベアメタルプロビジョニング

Bare Metal サービス (ironic) のインストール、設定、および使用方法

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

法律上の通知

概要

Red Hat OpenStack Platform 環境のオーバークラウドで Bare Metal Provisioning サービスをインストール、設定、および使用します。

目次

前書き

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、弊社の CTO、Chris Wright のメッセージ を参照してください。

RED HAT ドキュメントへのフィードバックの提供

弊社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。

ドキュメントへのダイレクトフィードバック (DDF) 機能の使用 (英語版のみ)

特定の文章、段落、またはコードブロックに対して直接コメントを送付するには、DDF の **Add Feedback** 機能を使用してください。なお、この機能は英語版のドキュメントでのみご利用いただけます。

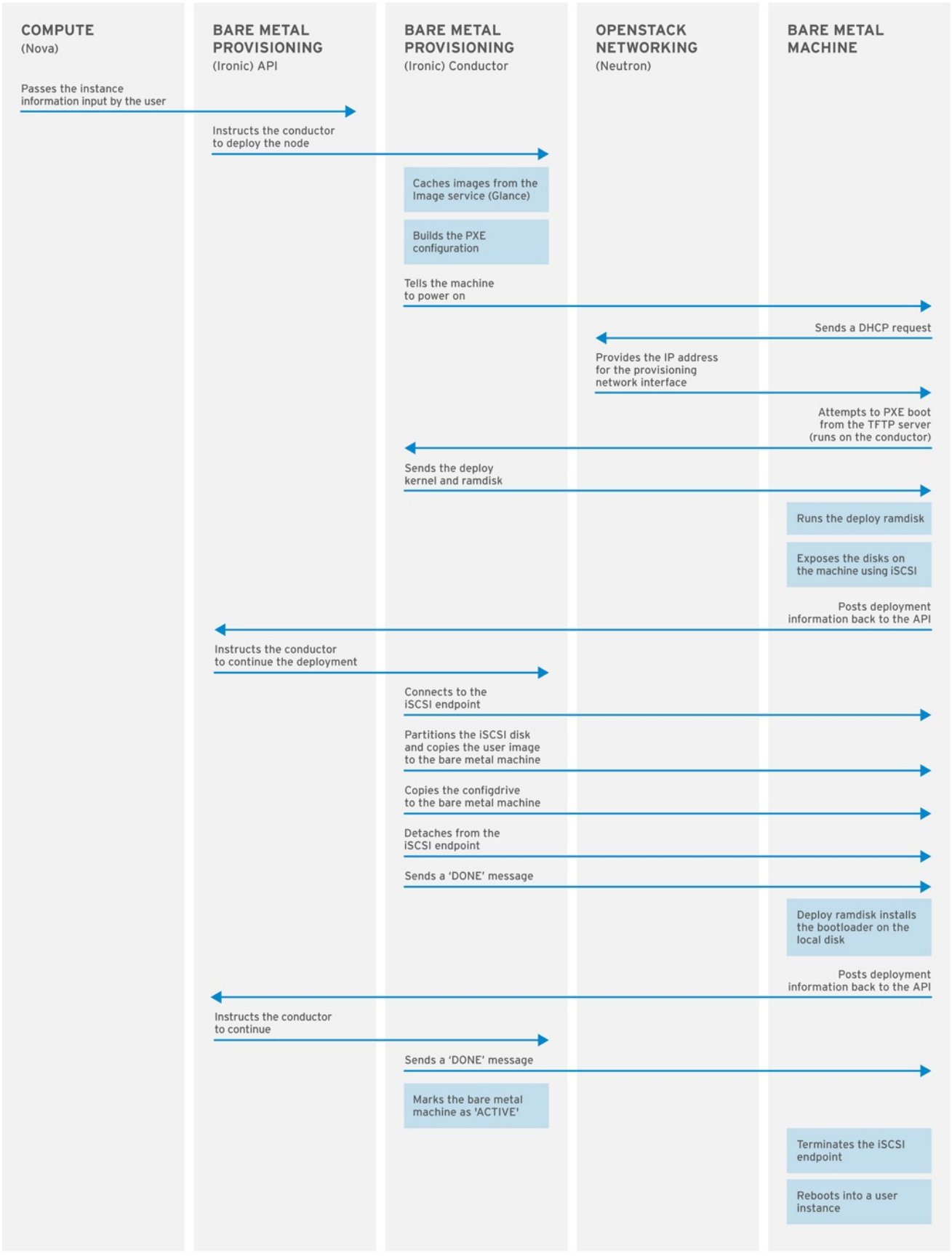
1. **Multi-page HTML** 形式でドキュメントを表示します。
2. ドキュメントの右上隅に **Feedback** ボタンが表示されていることを確認してください。
3. コメントするテキスト部分をハイライト表示します。
4. **Add Feedback** をクリックします。
5. **Add Feedback** フィールドにコメントを入力します。
6. (オプション) ドキュメントチームが連絡を取り問題についてお伺いできるように、ご自分のメールアドレスを追加します。
7. **Submit** をクリックします。

第1章 BARE METAL PROVISIONING サービス

Bare Metal Provisioning サービス (ironic) は、エンドユーザー向けの物理マシンのプロビジョニングと管理に使用できるコンポーネントを提供します。オーバークラウドの Bare Metal Provisioning サービスは、以下の OpenStack サービスと対話します。

- OpenStack Compute (nova) は、スケジューリング、テナントレベルのクォータ設定、IP の割り当ての機能と、仮想マシンインスタンスを管理するためのユーザー向けの API を提供します。一方、Bare Metal Provisioning サービスは、ハードウェア管理のための管理 API を提供します。
- OpenStack Identity (keystone) は、要求の認証機能を提供し、Bare Metal Provisioning サービスが他の OpenStack サービスを特定するのを補助します。
- OpenStack Image サービス (glance) は、イメージとイメージのメタデータを管理します。
- OpenStack Networking (neutron) は、DHCP とネットワーク設定を提供します。
- OpenStack Object Storage (swift) は、一部のドライバーの一時イメージ URL を公開します。

Bare Metal Provisioning サービスは、iPXE を使用して物理マシンをプロビジョニングします。以下の図は、デフォルトのドライバーを使用して新規マシンを起動した場合、プロビジョニングプロセス中に OpenStack のサービスがどのように対話するかを概説しています。



OPENSTACK_377593_1215

第2章 ベアメタルプロビジョニングの前提条件

ベアメタルのプロビジョニングを開始する前に、環境に必要なインストール、ハードウェア、およびネットワーク設定が含まれていることを確認してください。

- インストール要件の詳細は、「インストール要件」を参照してください。
- ハードウェア要件の詳細は、「ハードウェア要件」を参照してください。
- ネットワーク要件の詳細は、「ネットワーク要件」を参照してください。

2.1. インストール要件

- アンダークラウドノードに director がインストールされている。director のインストールについての詳しい情報は、「Installing the Undercloud」を参照してください。
- 残りのオーバークラウドと共に Bare Metal Provisioning サービスをインストールする準備が整っている。

注記

ベアメタルノードは、Red Hat OpenStack Platform (RHOSP) インストール環境のコントロールプレーンネットワークに直接アクセスできるため、オーバークラウドの Bare Metal Provisioning サービスは、信頼済みのテナント環境向けに設計されています。ユーザーがコントロールプレーンにアクセスせずに済むように、オーバークラウドの Ironic サービス用にカスタムのコンポーザブルネットワークを実装することもできます。

2.2. ハードウェア要件

オーバークラウドの要件

Bare Metal Provisioning サービスを有効にしたオーバークラウドのハードウェア要件は、標準のオーバークラウドと同じです。詳しい情報は、**『Director Installation and Usage』**の「Overcloud Requirements」を参照してください。

ベアメタルマシンの要件

プロビジョニングするベアメタルマシンのハードウェア要件は、インストールするオペレーティングシステムによって異なります。

- Red Hat Enterprise Linux 8 の場合は、『Red Hat Enterprise Linux 8 標準的な RHEL インストールの実行』を参照してください。
- Red Hat Enterprise Linux 7 の場合は、『Red Hat Enterprise Linux 7 インストールガイド』を参照してください。
- Red Hat Enterprise Linux 6 の場合は、『Red Hat Enterprise Linux 6 インストールガイド』を参照してください。

プロビジョニングするベアメタルマシンには、すべて以下に示す項目が必要です。

- ベアメタルネットワークに接続するための NIC 1 つ。
- **ironic-conductor** サービスから到達可能なネットワークに接続された電源管理インターフェース (例: IPMI)。コンポーザブルロールを使用して **ironic-conductor** を別の場所で実行する場合以外は、デフォルトでは **ironic-conductor** は全コントローラーノード上で実行されます。

- ベアメタルネットワーク上での PXE ブート。デプロイメント内のその他すべての NIC については PXE ブートを無効にしてください。

2.3. ネットワーク要件

ベアメタルネットワーク:

Bare Metal Provisioning サービスは、このプライベートネットワークを使用して以下の操作を行います。

- オーバークラウド上のベアメタルマシンのプロビジョニングと管理
- 再デプロイ前のベアメタルノードのクリーニング
- ベアメタルノードへのテナントアクセス

ベアメタルネットワークは、ベアメタルシステムを検出するための DHCP および PXE ブートの機能を提供します。このネットワークは、Bare Metal Provisioning サービスが PXE ブートと DHCP 要求に対応できるように、トランキングされたインターフェースでネイティブの VLAN を使用する必要があります。

ベアメタルネットワークを設定するには、2 とおりの方法があります。

- Ironic Conductor サービス用にフラットなベアメタルネットワークを使用する。このネットワークは、コントロールプレーン上の Ironic サービスにルーティングする必要があります。分離したベアメタルネットワークを定義すると、ベアメタルノードは PXE ブートすることができません。
- カスタムのコンポーザブルネットワークを使用して、オーバークラウドに Bare Metal Provisioning サービスを実装する。

注記

ベアメタルノードは、Red Hat OpenStack Platform (RHOSP) インストール環境のコントロールプレーンネットワークに直接アクセスできるため、オーバークラウドの Bare Metal Provisioning サービスは、信頼済みのテナント環境向けに設計されています。ユーザーがコントロールプレーンにアクセスせずに済むように、オーバークラウドの Ironic サービス用にカスタムのコンポーザブルネットワークを実装することもできます。

ネットワークのタグ付け:

- コントロールプレーンネットワーク (director のプロビジョニングネットワーク) は常にタグなしです。
- ベアメタルネットワークは、プロビジョニングのためにタグなしである必要があり、また Ironic API にアクセスできなければなりません。
- その他のネットワークはタグ付けすることができます。

オーバークラウドコントローラー:

Bare Metal Provisioning サービスをホストするコントローラーノードは、ベアメタルネットワークにアクセス可能である必要があります。

ベアメタルノード:

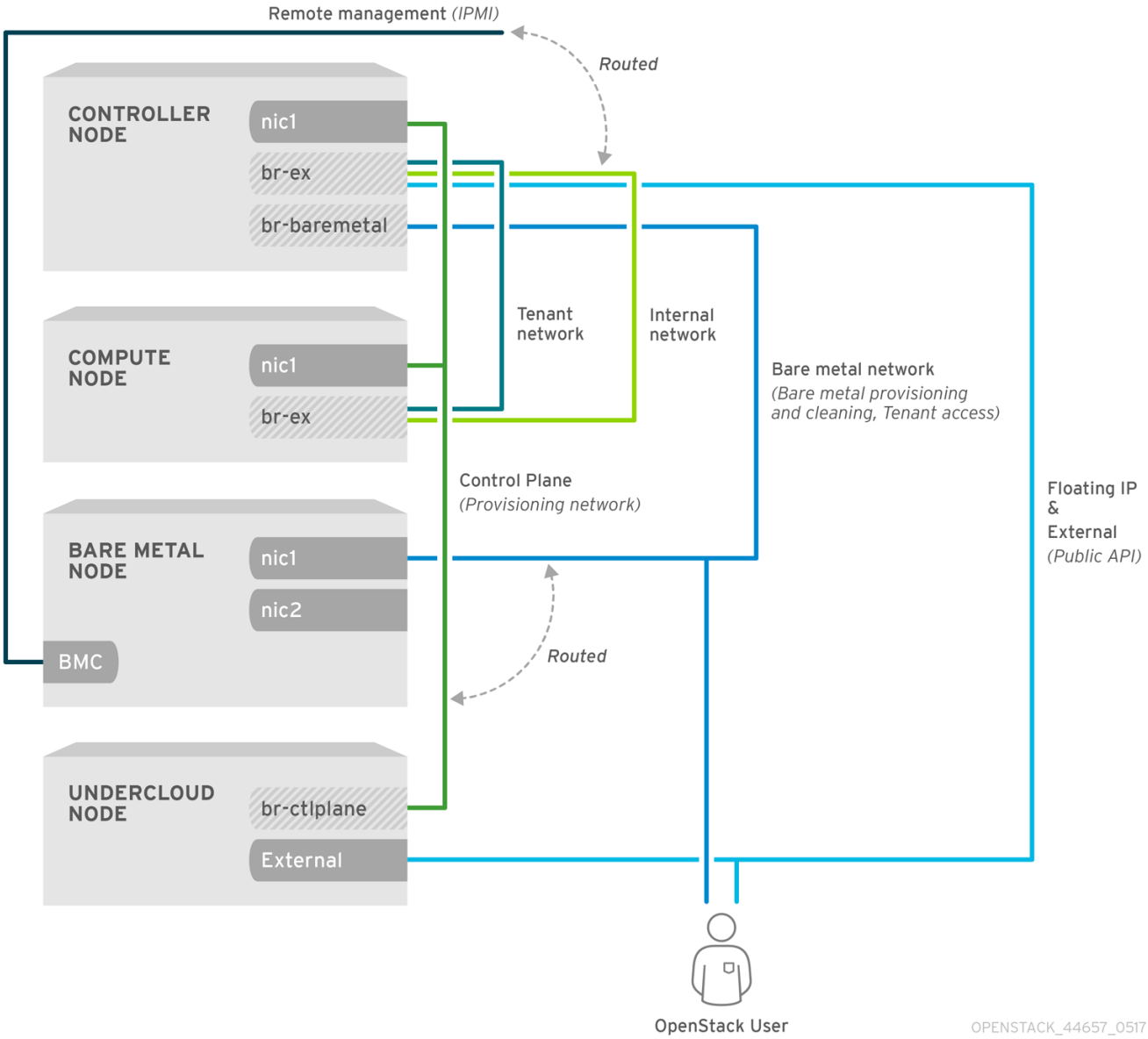
ベアメタルノードの PXE ブートに使用するように設定されている NIC は、ベアメタルネットワークにアクセス可能でなければなりません。

2.3.1. デフォルトのベアメタルネットワーク

このアーキテクチャーでは、ベアメタルネットワークはコントロールプレーンネットワークとは分離されています。ベアメタルネットワークは、テナントネットワークとしても機能するフラットネットワークです。

- ベアメタルネットワークは、OpenStack のオペレーターが作成します。このネットワークには、director のプロビジョニングネットワークへのルートが必要です。
- Bare Metal Provisioning サービスのユーザーは、パブリックの OpenStack API とベアメタルネットワークにアクセスすることができます。ベアメタルネットワークは、director のプロビジョニングネットワークにルーティングされるので、ユーザーはコントロールプレーンにも間接的にアクセスできます。
- Bare Metal Provisioning サービスは、ノードのクリーニングにベアメタルネットワークを使用します。

デフォルトのベアメタルネットワークアーキテクチャー図

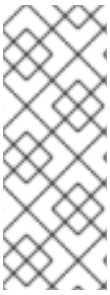


2.3.2. カスタムコンポーザブルベアメタルネットワーク

このアーキテクチャーでは、ベアメタルネットワークはコントロールプレーンにアクセスできないカスタムコンポーザブルネットワークです。コントロールプレーンへのアクセスを制限する場合、カスタムコンポーザブルネットワークを作成することができます。

- カスタムコンポーザブルベアメタルネットワークは、OpenStack のオペレーターが作成します。
- Ironic ユーザーは、パブリックの OpenStack API とカスタムコンポーザブルベアメタルネットワークにアクセスすることができます。
- Ironic は、ノードのクリーニングにカスタムコンポーザブルベアメタルネットワークを使用します。

第3章 BARE METAL PROVISIONING サービスを有効にした IPV4 オーバークラウドのデプロイ



注記

OVN を使用する場合、Bare Metal Provisioning サービス (ironic) は **ironic-overcloud.yaml** ファイルからの neutron DHCP エージェントでのみサポートされます。現在、OVN 上のビルトインの DHCP サーバーは、ベアメタルノードのプロビジョニングやプロビジョニングネットワーク用の DHCP を提供することができません。iPXE のチェーンブートにはタグ付け (dnsmasq の --dhcp-match) が必要ですが、OVN DHCP サーバーではサポートされていません。

以下の手順では、Bare Metal Provisioning サービス (ironic) に固有のデプロイメント手順を説明します。director を使用したオーバークラウドのデプロイメントについての詳しい情報は、『Director Installation and Usage』を参照してください。

前提条件

- お使いの環境が最小要件を満たしていること。詳細は、「2章 *ベアメタルプロビジョニングの前提条件*」を参照してください。

3.1. ベアメタルテンプレートの作成

環境ファイルを使用して、Bare Metal Provisioning サービスを有効にしたオーバークラウドをデプロイします。director ノードの **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic-overcloud.yaml** にあるテンプレートの例を使用できます。

前提条件

- アンダークラウドの正常なインストール。詳しくは、『director のインストールと使用方法』を参照してください。

テンプレート作成の完了

提供されているテンプレートまたは追加の yaml ファイル (例: **~/templates/ironic.yaml**) で、追加の設定を指定することができます。

- ベアメタルと仮想インスタンスの両方を備えたハイブリッドのデプロイメントでは、**NovaSchedulerDefaultFilters** の一覧に **AggregateInstanceExtraSpecsFilter** を追加する必要があります。**NovaSchedulerDefaultFilters** をどこにも設定していない場合には、**ironic.yaml** に設定することができます。例として、「テンプレートの例」を参照してください。

  注記

  SR-IOV を使用している場合には、**NovaSchedulerDefaultFilters** はすでに **tripleo-heat-templates/environments/neutron-sriov.yaml** で設定されています。このリストに **AggregateInstanceExtraSpecsFilter** を追記してください。

- 初回のデプロイメントおよび再デプロイメントの前に実行されるクリーニングの種別は、**IronicCleaningDiskErase** で設定されます。デフォルトでは、これは **deployment/ironic/ironic-conductor-container-puppet.yaml** によって「**full**」に設定されま

す。パーティションテーブルのみを消去するため、これを **metadata** に設定すると処理速度を大幅に向上させることができます。ただし、マルチテナント環境ではデプロイメントのセキュリティーレベルが低くなるため、信頼済みのテナント環境でのみこの操作を実施します。

- **IronicEnabledHardwareTypes** パラメーターを使用してドライバーを追加することができます。デフォルトでは、ipmi および **redfish** は有効になっています。

設定パラメーターの全一覧は、『**Overcloud Parameters**』の「Bare Metal」を参照してください。

3.1.1. テンプレートの例

テンプレートファイルの例を以下に示します。このファイルは、お使いの環境の要件を満たさない可能性があります。このサンプルを使用する前には、お使いの環境内の既存の設定に干渉しないことを確認してください。この例では、以下の設定を行います。

- **AggregateInstanceExtraSpecsFilter** は、ハイブリッドデプロイメント向けに、仮想インスタンスとベアメタルインスタンスの両方を許可します。
- 初回のデプロイメントまたは再デプロイメントの前に実行されるディスククリーニングでは、パーティションテーブル (metadata) のみが消去されます。

~/templates/ironic.yaml

```
parameter_defaults:

  NovaSchedulerDefaultFilters:
    - AggregateInstanceExtraSpecsFilter
    - AvailabilityZoneFilter
    - ComputeFilter
    - ComputeCapabilitiesFilter
    - ImagePropertiesFilter

  IronicCleaningDiskErase: metadata
```

3.2. ネットワーク設定

デフォルトのフラットベアメタルネットワークを使用する場合には、Bare Metal Provisioning サービス (ironic) が使用するブリッジ **br-baremetal** を作成する必要があります。このブリッジは、追加のテンプレートで指定することができます。

~/templates/network-environment.yaml

```
parameter_defaults:
  NeutronBridgeMappings: datacentre:br-ex,baremetal:br-baremetal
  NeutronFlatNetworks: datacentre,baremetal
```

コントローラーのプロビジョニングネットワーク (コントローラープレーン) をベアメタルネットワークとして再利用できるように、このネットワークにブリッジを設定するか、専用のネットワークを追加してブリッジを設定することができます。設定の要件は同じですが、ベアメタルネットワークはプロビジョニングに使用するので VLAN タグ付けはできません。

~/templates/nic-configs/controller.yaml

```
network_config:
```

```
-
  type: ovs_bridge
    name: br-baremetal
    use_dhcp: false
    members:
      -
        type: interface
        name: eth1
```

注記

ベアメタルノードは、Red Hat OpenStack Platform (RHOSP) インストール環境のコントロールプレーンネットワークに直接アクセスできるため、オーバークラウドの Bare Metal Provisioning サービスは、信頼済みのテナント環境向けに設計されています。

3.2.1. カスタムの IPv4 プロビジョニングネットワークの設定

テナントがアンダークラウドネットワークと干渉する場合があるので、デフォルトのフラットプロビジョニングネットワークにより、お客様の環境でセキュリティー上の問題が発生する可能性があります。このリスクを避けるために、コントロールプレーンにアクセスすることのできない、ironic サービス用のカスタムコンポーザブルベアメタルプロビジョニングネットワークを設定することができます。

前提条件

- アンダークラウドの正常なインストール。詳しくは、『Director Installation and Usage』を参照してください。

手順

1. Identity サービス (keystone) に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/stackrc
```

2. **network_data.yaml** ファイルをコピーします。

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/network_data.yaml .
```

3. 新しい **network_data.yaml** ファイルを編集し、IPv4 オーバークラウドプロビジョニング用の新規ネットワークを追加します。

```
# custom network for overcloud provisioning
- name: OcProvisioning
name_lower: oc_provisioning
vip: true
vlan: 205
ip_subnet: '172.23.3.0/24'
allocation_pools: [{'start': '172.23.3.10', 'end': '172.23.3.200'}]
```

4. 新規ネットワークを使用するために、**network_environments.yaml** ファイルおよび **nic-configs/controller.yaml** ファイルを更新します。

   a. **network_environments.yaml** ファイルで Ironic ネットワークを再マッピングします。

      ■

```
ServiceNetMap:
  IronicApiNetwork: oc_provisioning
  IronicNetwork: oc_provisioning
```

b. **nic-configs/controller.yaml** ファイルにおいて、インターフェースおよび必要なパラメーターを追加します。

```
$network_config:
    - type: vlan
      vlan_id:
        get_param: OcProvisioningNetworkVlanID
      addresses:
      - ip_netmask:
          get_param: OcProvisioningIpSubnet
```

5. **roles_data.yaml** ファイルをコピーします。

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/roles_data.yaml .
```

6. 新しい **roles_data.yaml** を編集し、コントローラー用の新規ネットワークを追加します。

```
networks:
 ...
  OcProvisioning:
    subnet: oc_provisioning_subnet
```

7. デプロイコマンドに新しい **network_data.yaml** ファイルと **roles_data.yaml** ファイルを追加します。

```
-n /home/stack/network_data.yaml \
-r /home/stack/roles_data.yaml \
```

3.3. オーバークラウドでのベアメタルイントロスペクションの有効化

ベアメタルのイントロスペクションを有効にするには、以下のファイルの両方をデプロイメントコマンドに追加します。

OVN を使用するデプロイメントの場合:

- **ironic-overcloud.yaml**
- **ironic-inspector.yaml**

注記

スパイン/リーフ型ルーティング対応デプロイメントでは、ToR ルーター上の DHCP リレーや、各サブネットに DHCP エージェントが必要な場合があります。メタデータサービスには、メタデータサーバーへの静的ルートが必要です。OVN は、デフォルトではベアメタルノードでこのルートを提供しません。

OVS を使用するデプロイメントの場合:

- **ironic.yaml**
- **ironic-inspector.yaml**

これらのファイルは、**/usr/share/openstack-tripleo-heat-templates/environments/services** ディレクトリーにあります。以下の例を使用して、実際の環境に対応する ironic インスペクターの設定の詳細情報を追加します。

```
parameter_defaults:
  IronicInspectorSubnets:
    - ip_range: <ip_range>
  IPAImageURLs: '["http://<ip_address>:<port>/agent.kernel", "http://<ip_address>:
<port>/agent.ramdisk"]'
  IronicInspectorInterface: 'br-baremetal'
```

IronicInspectorSubnets

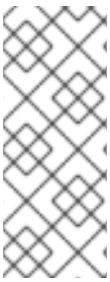
このパラメーターには複数の IP 範囲を含めることができ、スパインおよびリーフの両方に使用することができます。

IPAImageURLs

このパラメーターには、IPA カーネルおよび ramdisk に関する詳細が含まれます。多くの場合、アンダークラウドで使用するイメージと同じものを使用することができます。このパラメーターを省略する場合には、各コントローラーに代わりの URL を追加する必要があります。

IronicInspectorInterface

このパラメーターを使用して、ベアメタルのネットワークインターフェースを指定します。



注記

コンポーザブル Ironic ロールまたは IronicConductor ロールを使用する場合には、ロールファイルの Ironic ロールに **IronicInspector** サービスを含める必要があります。

```
ServicesDefault:
  OS::TripleO::Services::IronicInspector
```

3.4. オーバークラウドのデプロイ

Bare Metal Provisioning サービスを有効にするには、オーバークラウドの初回または再デプロイメントの時に、**-e** オプションを使用して ironic の環境ファイルを残りのオーバークラウド設定と共に追加します。以下の例を目安にしてください。

```
$ openstack overcloud deploy \
  --templates \
  -e ~/templates/node-info.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e ~/templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-overcloud.yaml \
  -e ~/templates/ironic.yaml \
```

関連資料

- オーバークラウドのデプロイについての詳しい情報は、『**director のインストール** と使用方法』の「デプロイメントコマンドオプション」および「オーバークラウド作成時の環境ファイル の追加」を参照してください。
- IPv6 を使用したオーバークラウドのデプロイに関する詳しい情報は、**『IPv6 Networking for the Overcloud』**の「Setting up your environment」および「Creating the overcloud」を参照してください。

3.5. BARE METAL PROVISIONING サービスのテスト

OpenStack Integration Test Suite を使用して、Red Hat OpenStack デプロイメントを検証することができます。詳しい情報は、『OpenStack Integration Test Suite Guide』を参照してください。

Bare Metal Provisioning サービスの追加検証方法:

1. Identity に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. **nova-compute** サービスがコントローラーノードで実行中であることを確認します。

```
$ openstack compute service list -c Binary -c Host -c Status
```

3. デフォルトの ironic ドライバーを変更した場合には、必要なドライバーを必ず有効にしてください。

```
$ openstack baremetal driver list
```

4. ironic のエンドポイントがリストされていることを確認します。

```
$ openstack catalog list
```

第4章 BARE METAL PROVISIONING サービスを有効にした IPV6 オーバークラウドのデプロイ

注記

OVN を使用する場合、Bare Metal Provisioning サービス (ironic) は **ironic-overcloud.yaml** ファイルからの neutron DHCP エージェントでのみサポートされます。現在、OVN 上のビルトインの DHCP サーバーは、ベアメタルノードのプロビジョニングやプロビジョニングネットワーク用の DHCP を提供することができません。iPXE のチェーンブートにはタグ付け (dnsmasq の --dhcp-match) が必要ですが、OVN DHCP サーバーではサポートされていません。

以下の手順では、Bare Metal Provisioning サービス (ironic) に固有のデプロイメント手順を説明します。director を使用したオーバークラウドのデプロイメントについての詳しい情報は、『Director Installation and Usage』を参照してください。

前提条件

- お使いの環境が最小要件を満たしていること。詳細は、「2章 *ベアメタルプロビジョニングの前提条件*」を参照してください。

4.1. ベアメタルテンプレートの作成

環境ファイルを使用して、Bare Metal Provisioning サービスを有効にしたオーバークラウドをデプロイします。director ノードの **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic-overcloud.yaml** にあるテンプレートの例を使用できます。

前提条件

- アンダークラウドの正常なインストール。詳しくは、『director のインストールと使用方法』を参照してください。

テンプレート作成の完了

提供されているテンプレートまたは追加の yaml ファイル (例: **~/templates/ironic.yaml**) で、追加の設定を指定することができます。

- ベアメタルと仮想インスタンスの両方を備えたハイブリッドのデプロイメントでは、**NovaSchedulerDefaultFilters** の一覧に **AggregateInstanceExtraSpecsFilter** を追加する必要があります。**NovaSchedulerDefaultFilters** をどこにも設定していない場合には、**ironic.yaml** に設定することができます。例として、「テンプレートの例」を参照してください。

  注記

  SR-IOV を使用している場合には、**NovaSchedulerDefaultFilters** はすでに **tripleo-heat-templates/environments/neutron-sriov.yaml** で設定されています。このリストに **AggregateInstanceExtraSpecsFilter** を追記してください。

- 初回のデプロイメントおよび再デプロイメントの前に実行されるクリーニングの種別は、**IronicCleaningDiskErase** で設定されます。デフォルトでは、これは **deployment/ironic/ironic-conductor-container-puppet.yaml** によって「**full**」に設定されま

す。パーティションテーブルのみを消去するため、これを **metadata** に設定すると処理速度を大幅に向上させることができます。ただし、マルチテナント環境ではデプロイメントのセキュリティーレベルが低くなるため、信頼済みのテナント環境でのみこの操作を実施します。

- **IronicEnabledHardwareTypes** パラメーターを使用してドライバーを追加することができます。デフォルトでは、ipmi および **redfish** は有効になっています。

設定パラメーターの全一覧は、『**Overcloud Parameters**』の「Bare Metal」を参照してください。

4.1.1. テンプレートの例

テンプレートファイルの例を以下に示します。このファイルは、お使いの環境の要件を満たさない可能性があります。このサンプルを使用する前には、お使いの環境内の既存の設定に干渉しないことを確認してください。この例では、以下の設定を行います。

- **AggregateInstanceExtraSpecsFilter** は、ハイブリッドデプロイメント向けに、仮想インスタンスとベアメタルインスタンスの両方を許可します。
- 初回のデプロイメントまたは再デプロイメントの前に実行されるディスククリーニングでは、パーティションテーブル (metadata) のみが消去されます。

~/templates/ironic.yaml

```
parameter_defaults:

  NovaSchedulerDefaultFilters:
    - AggregateInstanceExtraSpecsFilter
    - AvailabilityZoneFilter
    - ComputeFilter
    - ComputeCapabilitiesFilter
    - ImagePropertiesFilter

  IronicCleaningDiskErase: metadata
```

4.2. IPV6 を使用してベアメタルをプロビジョニングするためのアンダークラウド設定

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

IPv6 ノードおよびインフラストラクチャーがある場合には、IPv4 ではなく IPv6 を使用するようにアンダークラウドおよびプロビジョニングネットワークを設定することができます。これにより、director は IPv6 ノードに Red Hat OpenStack Platform をプロビジョニングおよびデプロイすることができます。ただし、いくつかの考慮事項があります。

- デュアルスタック IPv4/6 は利用できません。
- tempest 検証が正しく動作しない可能性があります。
- アップグレード時に IPv4 から IPv6 に移行することはできません。

undercloud.conf ファイルを変更して、Red Hat OpenStack Platform で IPv6 プロビジョニングを有効にします。

前提条件

- アンダークラウドの IPv6 アドレス。詳しい情報は、『**IPv6 Networking for the Overcloud**』の「Configuring an IPv6 address on the undercloud」を参照してください。

手順

1. サンプルの **undercloud.conf** ファイルをコピーするか、既存の **undercloud.conf** ファイルを変更します。
2. **undercloud.conf** ファイルで以下のパラメーター値を設定します。
   a. NIC が Red Hat OpenStack Platform でステートフル DHCPv6 をサポートする場合は、**ipv6_address_mode** を **dhcpv6-stateless** または **dhcpv6-stateful** に設定します。
   b. アンダークラウドでプロビジョニングネットワークにルーターを作成する必要がない場合は、**enable_routed_networks** を **true** に設定します。この場合、データセンタールーターはルーター広告を提供する必要があります。それ以外の場合は、この値を **false** に設定します。
   c. **local_ip** をアンダークラウドの IPv6 アドレスに設定します。
   d. アンダークラウドインターフェースのパラメーター **undercloud_public_host** と **undercloud_admin_host** に IPv6 アドレス設定を使用します。
   e. ステートフルアドレスモデル、ファームウェア、チェーンローダー、およびオペレーティングシステムは、異なるアルゴリズムを使用して DHCP サーバーが追跡する ID を生成する可能性があります。DHCPv6 は、MAC によってアドレスを追跡しません。要求元の ID の値が変更されても、MAC アドレスが同じままであれば、同じアドレスが返されません。ステートフル DHCPv6 を使用する場合は、**ironic_enabled_network_interfaces** パラメーターを使用して neutron インターフェースを指定します。**ironic_default_network_interface** パラメーターを使用して、neutron インターフェースをベアメタルノードのデフォルトネットワークインターフェースとして設定することもできます。
      - **ironic_enabled_network_interfaces = neutron,flat**
      - **ironic_default_network_interface = neutron**
   f. **[ctlplane-subnet]** セクションで、以下のパラメーターに IPv6 アドレス設定を使用します。
      - **cidr**
      - **dhcp_start**
      - **dhcp_end**
      - **gateway**
      - **inspection_iprange**
   g. **[ctlplane-subnet]** セクションで、**dns_nameservers** パラメーターにサブネットの IPv6 ネームサーバーを設定します。

```
[DEFAULT]
ipv6_address_mode = dhcpv6-stateless
enable_routed_networks: false
local_ip = <ipv6-address>
ironic_enabled_network_interfaces = neutron,flat
ironic_default_network_interface = neutron
undercloud_admin_host = <ipv6-address>
undercloud_public_host = <ipv6-address>

[ctlplane-subnet]
cidr = <ipv6-address>::<ipv6-mask>
dhcp_start = <ipv6-address>
dhcp_end = <ipv6-address>
dns_nameservers = <ipv6-dns>
gateway = <ipv6-address>
inspection_iprange = <ipv6-address>,<ipv6-address>
```

4.3. ネットワーク設定

デフォルトのフラットベアメタルネットワークを使用する場合には、Bare Metal Provisioning サービス (ironic) が使用するブリッジ **br-baremetal** を作成する必要があります。このブリッジは、追加のテンプレートで指定することができます。

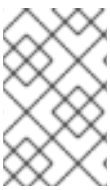
~/templates/network-environment.yaml

```
parameter_defaults:
  NeutronBridgeMappings: datacentre:br-ex,baremetal:br-baremetal
  NeutronFlatNetworks: datacentre,baremetal
```

コントローラーのプロビジョニングネットワーク (コントローラープレーン) をベアメタルネットワークとして再利用できるように、このネットワークにブリッジを設定するか、専用のネットワークを追加してブリッジを設定することができます。設定の要件は同じですが、ベアメタルネットワークはプロビジョニングに使用するので VLAN タグ付けはできません。

~/templates/nic-configs/controller.yaml

```
network_config:
    -
      type: ovs_bridge
        name: br-baremetal
        use_dhcp: false
        members:
          -
            type: interface
            name: eth1
```



注記

ベアメタルノードは、Red Hat OpenStack Platform (RHOSP) インストール環境のコントロールプレーンネットワークに直接アクセスできるため、オーバークラウドの Bare Metal Provisioning サービスは、信頼済みのテナント環境向けに設計されています。

4.3.1. カスタムの IPv6 プロビジョニングネットワークの設定

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

カスタムの IPv6 プロビジョニングネットワークを作成し、IPv6 を使用してオーバークラウドのプロビジョニングとデプロイを行います。

手順

1. Identity に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/stackrc
```

2. **network_data.yaml** ファイルをコピーします。

```
$ cp /usr/share/openstack-tripleo-heat-templates/network_data.yaml .
```

3. 新しい **network_data.yaml** ファイルを編集し、オーバークラウドプロビジョニング用の新規ネットワークを追加します。

```
# custom network for IPv6 overcloud provisioning
- name: OcProvisioningIPv6
vip: true
name_lower: oc_provisioning_ipv6
vlan: 10
ipv6: true
ipv6_subnet: '$IPV6_SUBNET_ADDRESS/$IPV6_MASK'
ipv6_allocation_pools: [{'start': '$IPV6_START_ADDRESS', 'end': '$IPV6_END_ADDRESS'}]
gateway_ipv6: '$IPV6_GW_ADDRESS'
```

- **$IPV6_ADDRESS** は、実際の IPv6 サブネットの IPv6 アドレスに置き換えます。
- **$IPV6_MASK** は、実際の IPv6 サブネット用の IPv6 ネットマスクに置き換えます。
- **$IPV6_START_ADDRESS** と **$IPV6_END_ADDRESS** は、アドレス割り当てに使用する IPv6 範囲に置き換えます。
- **$IPV6_GW_ADDRESS** は、実際のゲートウェイの IPv6 アドレスに置き換えます。

4. 新しいファイル **network-environment.yaml** を作成し、プロビジョニングネットワークの IPv6 設定を定義します。

```
$ touch /home/stack/network-environment.yaml`
```

a. 新しい IPv6 プロビジョニングネットワークを使用するように、ironic ネットワークを再マッピングします。

```
ServiceNetMap:
  IronicApiNetwork: oc_provisioning_ipv6
  IronicNetwork: oc_provisioning_ipv6
```

b. **IronicIpVersion** パラメーターを **6** に設定します。

```
parameter_defaults:
  IronicIpVersion: 6
```

c. **RabbitIPv6**、**MysqlIPv6**、および **RedisIPv6** の各パラメーターを、それぞれ **True** に設定します。

```
parameter_defaults:
  RabbitIPv6: True
  MysqlIPv6: True
  RedisIPv6: True
```

5. **nic-configs/controller.yaml** ファイルに、インターフェースおよび必要なパラメーターを追加します。

```
$network_config:
    - type: vlan
      vlan_id:
        get_param: OcProvisioningIPv6NetworkVlanID
      addresses:
      - ip_netmask:
          get_param: OcProvisioningIPv6IpSubnet
```

6. **roles_data.yaml** ファイルをコピーします。

```
(undercloud) [stack@host01 ~]$ cp /usr/share/openstack-tripleo-heat-
templates/roles_data.yaml .
```

7. 新しい **roles_data.yaml** を編集し、コントローラー用の新規ネットワークを追加します。

```
networks:
 ...
  - OcProvisioningIPv6
```

オーバークラウドをデプロイする際に、**-n** および **-r** オプションを指定して、デプロイメントコマンドに新しい **network_data.yaml** および **roles_data.yaml** ファイルを追加します。また、**-e** オプションを指定して、**network-environment.yaml** ファイルを追加します。

```
$ sudo openstack overcloud deploy --templates \
...
-n /home/stack/network_data.yaml \
-r /home/stack/roles_data.yaml \
-e /home/stack/network-environment.yaml
...
```

IPv6 ネットワーク設定についての詳しい情報は、**『IPv6 Networking for the Overcloud』**の「Configuring the network」を参照してください。

4.4. オーバークラウドでのベアメタルイントロスペクションの有効化

ベアメタルのイントロスペクションを有効にするには、以下のファイルの両方をデプロイメントコマンドに追加します。

OVN を使用するデプロイメントの場合:

- **ironic-overcloud.yaml**
- **ironic-inspector.yaml**

注記

スパイン/リーフ型ルーティング対応デプロイメントでは、ToR ルーター上の DHCP リレーや、各サブネットに DHCP エージェントが必要な場合があります。メタデータサービスには、メタデータサーバーへの静的ルートが必要です。OVN は、デフォルトではベアメタルノードでこのルートを提供しません。

OVS を使用するデプロイメントの場合:

- **ironic.yaml**
- **ironic-inspector.yaml**

これらのファイルは、**/usr/share/openstack-tripleo-heat-templates/environments/services** ディレクトリーにあります。以下の例を使用して、実際の環境に対応する ironic インスペクターの設定の詳細情報を追加します。

```
parameter_defaults:
  IronicInspectorSubnets:
    - ip_range: <ip_range>
  IPAImageURLs: '["http://<ip_address>:<port>/agent.kernel", "http://<ip_address>:
<port>/agent.ramdisk"]'
  IronicInspectorInterface: 'br-baremetal'
```

IronicInspectorSubnets

このパラメーターには複数の IP 範囲を含めることができ、スパインおよびリーフの両方に使用することができます。

IPAImageURLs

このパラメーターには、IPA カーネルおよび ramdisk に関する詳細が含まれます。多くの場合、アンダークラウドで使用するイメージと同じものを使用することができます。このパラメーターを省略する場合には、各コントローラーに代わりの URL を追加する必要があります。

IronicInspectorInterface

このパラメーターを使用して、ベアメタルのネットワークインターフェースを指定します。

注記

コンポーザブル Ironic ロールまたは IronicConductor ロールを使用する場合には、ロールファイルの Ironic ロールに **IronicInspector** サービスを含める必要があります。

```
ServicesDefault:
  OS::TripleO::Services::IronicInspector
```

4.5. オーバークラウドのデプロイ

Bare Metal Provisioning サービスを有効にするには、オーバークラウドの初回または再デプロイメントの時に、**-e** オプションを使用して ironic の環境ファイルを残りのオーバークラウド設定と共に追加します。以下の例を目安にしてください。

```
$ openstack overcloud deploy \
  --templates \
  -e ~/templates/node-info.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e ~/templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic-overcloud.yaml \
  -e ~/templates/ironic.yaml \
```

関連資料

- オーバークラウドのデプロイについての詳しい情報は、『**director のインストール** と使用方法』の「デプロイメントコマンドオプション」および「オーバークラウド作成時の環境ファイル の追加」を参照してください。
- IPv6 を使用したオーバークラウドのデプロイに関する詳しい情報は、**『IPv6 Networking for the Overcloud』**の「Setting up your environment」および「Creating the overcloud」を参照してください。

4.6. BARE METAL PROVISIONING サービスのテスト

OpenStack Integration Test Suite を使用して、Red Hat OpenStack デプロイメントを検証することができます。詳しい情報は、『OpenStack Integration Test Suite Guide』を参照してください。

Bare Metal Provisioning サービスの追加検証方法:

1. Identity に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. **nova-compute** サービスがコントローラーノードで実行中であることを確認します。

```
$ openstack compute service list -c Binary -c Host -c Status
```

3. デフォルトの ironic ドライバーを変更した場合には、必要なドライバーを必ず有効にしてください。

```
$ openstack baremetal driver list
```

4. ironic のエンドポイントがリストされていることを確認します。

```
$ openstack catalog list
```

第5章 デプロイ後の BARE METAL PROVISIONING サービスの設定

Bare Metal Provisioning サービス (ironic) を有効にしてオーバークラウドをデプロイしたら、ベアメタルの負荷用に環境を準備するために追加の設定を完了する必要がある場合があります。

- ネットワークの設定
- ノードのクリーニングの設定
- ベアメタルノード用のベアメタルフレーバーおよびイメージの作成
- デプロイインターフェースの設定
- 仮想メディアブートの設定
- 仮想マシンと物理マシンのプロビジョニングの分離

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

5.1. OPENSTACK のネットワーク設定

DHCP、PXE ブート、およびその他の必要な場合に OpenStack Networking が Bare Metal Provisioning サービスと通信するように設定します。ベアメタルネットワークを設定するには、2 とおりの方法があります。

- Ironic Conductor サービス用にフラットなベアメタルネットワークを使用する。このネットワークは、コントロールプレーンネットワーク上の Ironic サービスにルーティングする必要があります。
- カスタムのコンポーザブルネットワークを使用して、オーバークラウドに Ironic サービスを実装する。

本項の手順に従って、ベアメタルマシンのプロビジョニングに使用する単一のフラットなネットワーク向けに OpenStack Networking を設定するか、あるいは未使用の分離ネットワークまたはフラットネットワークに依存しない新たなコンポーザブルネットワークを設定します。この設定では、ML2 プラグインと Open vSwitch エージェントを使用します。

5.1.1. OpenStack Networking がフラットなベアメタルネットワーク上の Bare Metal Provisioning サービスと通信するための設定

OpenStack Networking サービスをホストするサーバーにおいて、以下の手順に記載するすべてのステップを **root** ユーザーとして実行します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6*

オーバークラウドのデプロイ」を参照してください。

手順

1. Identity に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. ベアメタルインスタンスをプロビジョニングするためのフラットなネットワークを作成します。

```
$ openstack network create \
  --provider-network-type flat \
  --provider-physical-network baremetal \
  --share NETWORK_NAME
```

NETWORK_NAME は、このネットワークの名前に置き換えます。仮想ネットワークの実装先となる物理ネットワークの名前 (この場合は **baremetal**) は、以前の手順で **~/templates/network-environment.yaml** ファイルの **NeutronBridgeMappings** パラメーターで設定されています。

3. フラットネットワーク上にサブネットを作成します。

```
$ openstack subnet create \
  --network NETWORK_NAME \
  --subnet-range NETWORK_CIDR \
  --ip-version 4 \
  --gateway GATEWAY_IP \
  --allocation-pool start=START_IP,end=END_IP \
  --dhcp SUBNET_NAME
```

以下の値を置き換えてください。

- SUBNET_NAME は、サブネットの名前に置き換えます。
- NETWORK_NAME は、以前のステップで作成済みのプロビジョニングネットワークの名前に置き換えます。
- NETWORK_CIDR は、サブネットが示す IP アドレスブロックの Classless Inter-Domain Routing (CIDR) 表記に置き換えます。START_IP で始まり END_IP で終る範囲で指定する IP アドレスブロックは、NETWORK_CIDR で指定されている IP アドレスブロックの範囲内になければなりません。
- GATEWAY_IP は、新しいサブネットのゲートウェイとして機能するルーターインターフェースの IP アドレスまたはホスト名に置き換えます。このアドレスは、NETWORK_CIDR で指定されている IP アドレスブロック内で、かつ START_IP で始まり END_IP で終わる範囲で指定されている IP アドレスブロック外でなければなりません。
- START_IP は、Floating IP アドレスを確保する新規サブネット内の IP アドレス範囲の開始アドレスを示す IP アドレスに置き換えます。
- END_IP は、Floating IP アドレスを確保する新規サブネット内の IP アドレス範囲の終了アドレスを示す IP アドレスに置き換えます。

4. ネットワークとサブネット用のルーターを作成して、OpenStack Networking サービスがメタデータ要求に応答するようにします。

```
$ openstack router create ROUTER_NAME
```

ROUTER_NAME は、ルーターの名前に置き換えます。

5. サブネットを新しいルーターに接続します。

```
$ openstack router add subnet ROUTER_NAME BAREMETAL_SUBNET
```

ROUTER_NAME をルーターの名前に、**BAREMETAL_SUBNET** を以前のステップで作成したサブネットの ID または名前に、それぞれ置き換えます。これにより、**cloud-init** からのメタデータ要求に対応すると共に、ノードを設定することができます。

5.1.2. OpenStack Networking がカスタムコンポーザブルベアメタルネットワーク上の Bare Metal Provisioning サービスと通信するための設定

OpenStack Networking サービスをホストするサーバーにおいて、以下の手順に記載するすべてのステップを **root** ユーザーとして実行します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. デプロイメント中に作成する **OcProvisioning** ネットワークと一致する VLAN ID で、VLAN ネットワークを作成します。クリーニングネットワークのデフォルト名と一致するように、新しいネットワークの名前を **provisioning** と設定します。

```
(overcloud) [stack@host01 ~]$ openstack network create \
  --share \
  --provider-network-type vlan \
  --provider-physical-network datacentre \
  --provider-segment 205 provisioning
```

オーバークラウドネットワークの名前が **provisioning** ではない場合には、**IronicProvisioningNetwork** パラメーターを **provisioning** に設定し、オーバークラウドを再デプロイします。

~/templates/ironic.yaml

```
parameter_defaults:
  IronicProvisioningNetwork:
    default: provisioning
    description: Name or UUID of the overcloud network used for provisioning bare metal
nodes, if IronicDefaultNetworkInterface is set to "neutron". The default value can be left
during the initial deployment and should be changed to an actual UUID in a post-deployment
stack update.
    type: string
```

5.2. ノードのクリーニングの設定

デフォルトでは、Bare Metal Provisioning サービスは、ノードのクリーニングに **provisioning** という名前のネットワークを使用します。ただし、OpenStack Networking ではネットワーク名は一意ではないので、テナントが同じ名前を使用してネットワークを作成して Bare Metal Provisioning サービスとの競合が発生する可能性があります。競合を回避するには、代わりにネットワークの UUID を使用します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

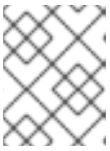
1. ノードのクリーニングを設定するには、Bare Metal Provisioning サービスをホストするコントローラー上のプロバイダーネットワークの UUID を指定します。
~/templates/ironic.yaml

```
parameter_defaults:
  IronicCleaningNetwork: <UUID>
```

<UUID> は、以前のステップで作成したベアメタルネットワークの UUID に置き換えます。

UUID は、**openstack network show** コマンドで確認することができます。

```
openstack network show NETWORK_NAME -f value -c id
```



注記

ネットワークの UUID は、オーバークラウドの初回のデプロイメントが完了するまで利用できないので、この設定はデプロイ後に実行する必要があります。

2. 変更を適用するには、**openstack overcloud deploy** でオーバークラウドを再デプロイします。デプロイメントコマンドについての詳しい情報は、「オーバークラウドのデプロイ」を参照してください。

5.2.1. ノードの手動によるクリーニング

ノードのクリーニングを手動で開始するには、そのノードが **manageable** の状態でなければなりません。

ノードのクリーニングには 2 つのモードがあります。

メタデータのみのクリーニング: 対象のノード上の全ディスクからパーティションを削除します。この方法は、より高速なクリーニングサイクルですが、パーティションテーブルのみが削除されるので、セキュリティーレベルはより低くなります。このモードは、信頼済みのテナント環境でのみ使用してください。

完全なクリーニング: ATA のセキュア消去を使用するか、細断処理を行って、全ディスクから全データを削除します。処理の完了まで数時間かかる場合があります。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

metadata のクリーニングを開始するには、以下のコマンドを実行します。

```
$ openstack baremetal node clean _UUID_ \
    --clean-steps '[{"interface": "deploy", "step": "erase_devices_metadata"}]'
```

full クリーニングを開始するには、以下のコマンドを実行します。

```
$ openstack baremetal node clean _UUID_ \
    --clean-steps '[{"interface": "deploy", "step": "erase_devices"}]'
```

UUID は、クリーニングするノードの UUID に置き換えます。

クリーニングが正常に完了すると、ノードの状態は **manageable** に戻ります。状態が **clean failed** の場合には、**last_error** のフィールドで失敗の原因を調査してください。

5.3. ベアメタルフレーバーおよびリソースクラスの作成

特定のワークロード用にベアメタルノードのタグ付けに使用するフレーバーおよびリソースクラスを作成する必要があります。

手順

1. source コマンドでオーバークラウドの認証情報ファイルを読み込みます。

```
$ source ~/overcloudrc
```

2. ベアメタルノード用に新規インスタンスフレーバーを作成します。

```
(overcloud)$ openstack flavor create --id auto \
 --ram <ram_size_mb> --disk <disk_size_gb> \
 --vcpus <no_vcpus> baremetal
```

   - **<ram_size_mb>** をベアメタルノードの RAM (MB 単位) に置き換えます。
   - **<disk_size_gb>** をベアメタルノード上のディスク容量 (GB 単位) に置き換えます。
   - **<no_vcpus>** をベアメタルノードの CPU 数に置き換えます。

注記

これらの属性は、インスタンスのスケジューリングには使用されません。ただし Compute スケジューラーは、ディスク容量を使用してルートパーティションのサイズを決定します。

3. ノード一覧を取得して UUID を把握します。

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

4. 各ベアメタルノードにカスタムのベアメタルリソースクラスをタグ付けします。

```
(overcloud)$ openstack baremetal node set \
 --resource-class baremetal.<CUSTOM> <node>
```

- &lt **;CUSTOM&gt**; を、リソースクラスの目的を特定する文字列に置き換えます。たとえば、**GPU** に設定して、GPU 負荷用に指定するベアメタルノードにタグ付けするために使用できるカスタム GPU リソースクラスを作成します。
- **<node>** をベアメタルノードの ID に置き換えてください。

5. ベアメタルノードの新規インスタンスフレーバーをカスタムリソースクラスに関連付けます。

```
(overcloud)$ openstack flavor set \
 --property resources:CUSTOM_BAREMETAL_<CUSTOM>=1 \
 baremetal
```

Bare Metal サービスノードのリソースクラスに対応するカスタムリソースクラスの名前を指定するには、リソースクラスを大文字に変換し、それぞれの句読点をアンダースコアに置き換え、**CUSTOM_** のプレフィックスを追加します。

注記

フレーバーが要求できるのは、ベアメタルリソースクラスの 1 つのインスタンスだけです。

6. 以下のフレーバー属性を設定して、Compute スケジューラーがインスタンスのスケジューリングにベアメタルフレーバー属性を使用するのを防ぎます。

```
(overcloud)$ openstack flavor set \
 --property resources:VCPU=0 \
 --property resources:MEMORY_MB=0 \
 --property resources:DISK_GB=0 baremetal
```

7. 新規フレーバーの値が正しいことを確認します。

```
(overcloud)$ openstack flavor list
```

5.4. ベアメタルイメージの作成

Bare Metal Provisioning サービス (ironic) が含まれるオーバークラウドには、2 つのイメージセットが必要です。デプロイメント時に、Bare Metal Provisioning サービスはデプロイイメージからベアメタルノードをブートし、ユーザーイメージをノードにコピーします。

デプロイイメージ

Bare Metal Provisioning サービスはデプロイイメージを使用して、ベアメタルノードをブートしてユーザーイメージをベアメタルノードにコピーします。デプロイイメージは、**カーネル** イメージと **ramdisk** イメージで構成されます。

ユーザーイメージ

ユーザーイメージは、ベアメタルノードにデプロイするイメージです。ユーザーイメージにも **カーネル** イメージと **ramdisk** イメージが含まれますが、追加で **メイン** イメージも含まれます。メインイメージは、ルートパーティションイメージまたは完全なディスクイメージのいずれかです。

- **完全なディスクイメージ** は、パーティションテーブルとブートローダーを含むイメージです。完全なディスクイメージを使用してデプロイされたノードはローカルブートをサポートするので、Bare Metal Provisioning サービスはデプロイ後のノードのリブートは制御しません。
- **ルートパーティションイメージ** には、オペレーティングシステムのルートパーティションのみが含まれています。ルートパーティションを使用する場合には、デプロイイメージが Image サービスに読み込まれた後に、ノードのプロパティーにデプロイイメージをノードのブートイメージとして設定することができます。デプロイ後のノードのリブートでは、netboot を使用してユーザーイメージがプルダウンされます。

本項に記載する例では、ルートパーティションイメージを使用してベアメタルノードをプロビジョニングします。

5.4.1. デプロイイメージの準備

デプロイイメージを作成する必要はありません。アンダークラウドによるオーバークラウドのデプロイ時に、すでにデプロイイメージが作成されているためです。デプロイイメージは、以下に示したように、カーネルイメージと ramdisk イメージの 2 つのイメージで構成されます。

```
/tftpboot/agent.kernel
/tftpboot/agent.ramdisk
```

これらのイメージは、削除したり他の場所でアンパックしたりしていない限りは、多くの場合、ホームディレクトリーにあります。ホームディレクトリーにない場合でも、**rhosp-director-images-ipa** パッケージがインストールされているので、これらのイメージは **/usr/share/rhosp-director-images/ironic-python-agent*.tar** ファイル内にあります。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

イメージを抽出して Image サービスにアップロードします。

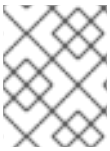
```
$ openstack image create \
  --container-format aki \
  --disk-format aki \
  --public \
  --file ./tftpboot/agent.kernel bm-deploy-kernel
$ openstack image create \
  --container-format ari \
  --disk-format ari \
  --public \
  --file ./tftpboot/agent.ramdisk bm-deploy-ramdisk
```

5.4.2. ユーザーイメージの準備

最後に必要となるイメージは、ベアメタルノードにデプロイするユーザーイメージです。ユーザーイメージには、カーネルイメージと ramdisk イメージに加えて、メインイメージが含まれます。これらのパッケージをダウンロードしてインストールするには、まずご自分の要件に合わせて完全なディスクイメージの環境変数を設定する必要があります。

5.4.2.1. ディスクイメージの環境変数

ディスクイメージのビルドプロセスとして、director にはベースイメージと、新規オーバークラウドイメージのパッケージを取得するための登録情報が必要です。これらの属性は、以下に示す Linux の環境変数を使用して定義します。



注記

イメージのビルドプロセスにより、イメージは一時的に Red Hat サブスクリプションに登録され、イメージのビルドプロセスが完了するとシステムの登録が解除されます。

ディスクイメージをビルドするには、Linux の環境変数をお使いの環境と要件に応じて設定します。

DIB_LOCAL_IMAGE
: 完全なディスクイメージのベースに使用するローカルイメージを設定します。

REG_ACTIVATION_KEY
: 登録プロセスにおいて、ログイン情報の代わりにアクティベーションキーを使用します。

REG_AUTO_ATTACH
: 最も互換性のあるサブスクリプションを自動的にアタッチするかどうかを定義します。

REG_BASE_URL
: イメージのパッケージが含まれるコンテンツ配信サーバーのベース URL。カスタマーポータル Subscription Management のデフォルトプロセスでは **https://cdn.redhat.com** を使用します。Red Hat Satellite 6 サーバーを使用している場合は、このパラメーターをお使いの Satellite サーバーのベース URL に設定します。

REG_ENVIRONMENT
: 組織内の環境に登録します。

REG_METHOD
: 登録の方法を設定します。Red Hat カスタマーポータルに登録するには **portal** を使用します。Red Hat Satellite 6 で登録するには、**satellite** を使用します。

REG_ORG
: イメージを登録する組織

REG_POOL_ID
: 製品のサブスクリプション情報のプール ID

REG_PASSWORD
: イメージを登録するユーザーアカウントのパスワードを設定します。

REG_RELEASE
: Red Hat Enterprise Linux のマイナーリリースバージョンを設定します。**REG_AUTO_ATTACH** または **REG_POOL_ID** 環境変数でこれを使用する必要があります。

REG_REPOS
: リポジトリー名のコンマ区切り文字列。この文字列の各リポジトリーは **subscription-manager** で有効にされます。

REG_SAT_URL

オーバークラウドノードを登録する Satellite サーバーのベース URL。このパラメーターには、HTTPS URL ではなく、Satellite の HTTP URL を使用します。たとえば、https://satellite.example.com ではなく http://satellite.example.com を使用します。

REG_SERVER_URL

使用するサブスクリプションサービスのホスト名を設定します。Red Hat カスタマーポータルの場合、デフォルトホスト名は **subscription.rhn.redhat.com** です。Red Hat Satellite 6 サーバーを使用している場合は、このパラメーターをお使いの Satellite サーバーのホスト名に設定します。

REG_USER

イメージを登録するアカウントのユーザー名を設定します。

5.4.3. ユーザーイメージのインストール

ユーザーイメージを設定してから、Image サービス (glance) にイメージをアップロードします。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. カスタマーポータル から Red Hat Enterprise Linux KVM ゲストイメージをダウンロードします。
2. **DIB_LOCAL_IMAGE** をダウンロードしたイメージとして定義します。

```
$ export DIB_LOCAL_IMAGE=rhel-8.0-x86_64-kvm.qcow2
```

3. 登録情報を設定します。Red Hat カスタマーポータルを使用する場合には、以下の情報を設定する必要があります。

```
$ export REG_USER='USER_NAME'
$ export REG_PASSWORD='PASSWORD'
$ export REG_AUTO_ATTACH=true
$ export REG_METHOD=portal
$ export https_proxy='IP_address:port' (if applicable)
$ export http_proxy='IP_address:port' (if applicable)
```

Red Hat Satellite を使用する場合には、以下の情報を設定する必要があります。

```
$ export REG_USER='USER_NAME'
$ export REG_PASSWORD='PASSWORD'
$ export REG_SAT_URL='<SATELLITE URL>'
$ export REG_ORG='<SATELLITE ORG>'
$ export REG_ENV='<SATELLITE ENV>'
$ export REG_METHOD=<METHOD>
```

オフラインのリポジトリーがある場合には、DIB_YUM_REPO_CONF をローカルリポジトリーの設定として定義することができます。

-

```
$ export DIB_YUM_REPO_CONF=<path-to-local-repository-config-file>
```

4. **diskimage-builder** ツールを使用してユーザーイメージを作成します。

```
$ export DIB_RELEASE=8
$ disk-image-create rhel baremetal -o rhel-image
```

このコマンドにより、カーネルは **rhel-image.vmlinuz** として、初期 ramdisk は **rhel-image.initrd** として、それぞれ抽出されます。

5. イメージを Image サービスにアップロードします。

```
$ KERNEL_ID=$(openstack image create \
  --file rhel-image.vmlinuz --public \
  --container-format aki --disk-format aki \
  -f value -c id rhel-image.vmlinuz)
$ RAMDISK_ID=$(openstack image create \
  --file rhel-image.initrd --public \
  --container-format ari --disk-format ari \
  -f value -c id rhel-image.initrd)
$ openstack image create \
  --file rhel-image.qcow2   --public \
  --container-format bare \
  --disk-format qcow2 \
  --property kernel_id=$KERNEL_ID \
  --property ramdisk_id=$RAMDISK_ID \
  rhel-image
```

5.5. デプロイインターフェースの設定

ベアメタルノードをプロビジョニングする場合、オーバークラウド上の Bare Metal Provisioning サービス (ironic) は、ベアメタルノード上のディスクにベースオペレーティングシステムのイメージを書き込みます。デフォルトでは、デプロイインターフェースは iSCSI マウントにイメージをマウントし、そのイメージを各ノードのディスクにコピーします。あるいは、「直接デプロイ」を使用して、ディスクイメージを HTTP の保管場所から直接ベアメタルノード上のディスクに書き込むこともできます。

プロビジョニングプロセスでは、デプロイインターフェースが重要な役割を果たします。デプロイインターフェースはデプロイメントをオーケストレーションし、イメージをターゲットディスクに転送するメカニズムを定義します。

前提条件

- **ironic-conductor** を実行する Bare Metal サービスノードに設定された依存関係パッケージ。
- OpenStack Compute (nova) が Bare Metal サービスのエンドポイントを使用するように設定されている。
- 利用可能なハードウェア用にフレーバーが作成され、nova が正しいフレーバーから新規ノードを起動する。
- Image サービス (glance) でイメージが利用可能であること。
  - bm-deploy-kernel
  - bm-deploy-ramdisk

- user-image
- user-image-vmlinuz
- user-image-initrd

- Ironic API サービスに登録するためのハードウェア

ワークフロー
以下に示すワークフローの例を使用して、標準的なデプロイプロセスについて説明します。使用する ironic ドライバーインターフェースによって、一部の手順が異なる場合があります。

1. Nova スケジューラーが Nova API からインスタンスのブート要求を受け取る。
2. Nova スケジューラーが該当するハイパーバイザーを識別し、ターゲットの物理ノードを識別する。
3. Nova Compute マネージャーが選択したハイパーバイザーのリソースを要求する。
4. Nova のブート要求が指定するネットワークインターフェースに基づき、Nova Compute マネージャーがバインド前のテナント仮想インターフェース (VIF) を Networking サービスに作成する。
5. Nova Compute が Nova Compute の仮想レイヤーから **driver.spawn** を呼び出し、必要なすべての情報が含まれる子タスクを作成する。子タスク作成プロセス中に、仮想ドライバーは以下の処理を完了します。
    a. デプロイイメージ、インスタンスの UUID、要求された機能、およびフレーバー属性に関する情報で、ターゲットの ironic ノードを更新する。
    b. ironic API をコールして、ターゲットノードの電源およびデプロイインターフェースを検証する。
    c. VIF をノードに接続する。それぞれの neutron ポートは、任意の ironic ポートまたはグループにアタッチすることができます。ポートグループはポートに優先します。
    d. コンフィグドライブを生成する。
6. Nova ironic 仮想ドライバーが、Ironic API を使用してベアメタルノードに対応する Ironic Conductor にデプロイ要求を発行する。
7. 仮想インターフェースが接続され、PXE/TFTP オプションを設定するために Neutron API が DHCP を更新する。
8. ironic ノードのブートインターフェースが (i)PXE 設定を準備し、デプロイカーネルおよび ramdisk をキャッシュする。
9. ironic ノードの管理インターフェースがコマンドを発行し、ノードのネットワークブートを有効にする。
10. 必要に応じて、ironic ノードのデプロイインターフェースがインスタンスイメージ、カーネル、および ramdisk をキャッシュする。
11. ironic ノードの電源インターフェースがノードに電源投入を指示する。
12. ノードがデプロイ ramdisk を起動する。
13. iSCSI デプロイメントの場合には、Conductor が iSCSI 経由でイメージを物理ノードにコピー

する。直接デプロイメントの場合には、デプロイ ramdisk が一時 URL からイメージをダウンロードする。この URL は、Swift API と互換性のあるオブジェクトストアまたは HTTP の URL でなければなりません。

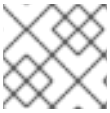
14. ノードのブートインターフェースがインスタンスイメージを参照するように PXE 設定を切り替え、ramdisk エージェントにノードのソフトパワーオフを指示する。ソフトパワーオフに失敗した場合には、ベアメタルノードの電源は IPMI/BMC により切断されます。

15. デプロイインターフェースがネットワークインターフェースにすべてのプロビジョニングポートの削除を指示し、テナントポートをノードにバインドし、ノードの電源を投入する。

これで、新規ベアメタルノードのプロビジョニングの状態が **active** になります。

5.5.1. オーバークラウドにおける直接デプロイインターフェースの設定

iSCSI デプロイインターフェースがデフォルトのデプロイインターフェースです。ただし、直接デプロイインターフェースを有効にして、イメージを HTTP の保管場所からターゲットディスクにダウンロードすることができます。



注記

オーバークラウドノードのメモリー **tmpfs** には、少なくとも 8 GB の RAM が必要です。

手順

1. カスタム環境ファイル **/home/stack/templates/direct_deploy.yaml** を作成または変更し、**IronicEnabledDeployInterfaces** パラメーターおよび **IronicDefaultDeployInterface** パラメーターを指定します。

```
parameter_defaults:
  IronicEnabledDeployInterfaces: direct
  IronicDefaultDeployInterface: direct
```

   iscsi を使用するようにノードを登録する場合には、**IronicEnabledDeployInterfaces** パラメーターに **iscsi** の値を含めます。

```
parameter_defaults:
  IronicEnabledDeployInterfaces: direct,iscsi
  IronicDefaultDeployInterface: direct
```

2. デフォルトでは、各ノードの Bare Metal Provisioning サービス (ironic) エージェントは、HTTP リンクを通じて Object Storage サービス (swift) に保管されているイメージを取得します。あるいは、ironic は、**ironic-conductor** HTTP サーバーを通じて、このイメージを直接ノードにストリーミングすることもできます。イメージを提供するサービスを変更するには、**/home/stack/templates/direct_deploy.yaml** ファイルの **IronicImageDownloadSource** を **http** に設定します。

```
parameter_defaults:
  IronicEnabledDeployInterfaces: direct
  IronicDefaultDeployInterface: direct
  IronicImageDownloadSource: http
```

3. オーバークラウドのデプロイメントにカスタム環境ファイルを追加します。

```
$ openstack overcloud deploy \
  --templates \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/ironic.yaml \
  -e /home/stack/templates/direct_deploy.yaml \
  ...
```

デプロイメントが完了するまで待ちます。

注記

IronicDefaultDeployInterface を指定しない、または別のデプロイインターフェースを使用する場合には、ノードを作成または更新する際にデプロイインターフェースを指定します。

```
$ openstack baremetal node create --driver ipmi --deploy-interface direct
$ openstack baremetal node set <NODE> --deploy-interface direct
```

5.6. ベアメタルノードとしての物理マシンの追加

ベアメタルノードの登録には 2 つの方法があります。

1. ノードの詳細情報を記載したインベントリーファイルを作成し、そのファイルを Bare Metal Provisioning サービスにインポートしてノードを利用できるようにします。
2. 物理ノードをベアメタルノードとして登録してから、手動でハードウェア情報を追加し、各イーサネットの MAC アドレス用にポートを作成します。これらの手順は、**overcloudrc** ファイルがある任意のノードで実行できます。

物理マシンの登録後、新規リソースは Compute に直ぐには通知されません。これは、Compute のリソーストラッカーが定期的にしか同期していないためです。次の定期タスクの実行後に変更が反映されます。定期的なタスクの頻度は、**/etc/nova/nova.conf** ファイルの **scheduler_driver_task_period** で更新できます。デフォルトの間隔は 60 秒です。

5.6.1. インベントリーファイルを使用したベアメタルノードの登録

ノードの詳細情報を記載したインベントリーファイルを作成し、そのファイルを Bare Metal Provisioning サービスにインポートしてノードを利用できるようにします。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. ノードの詳細情報を記載したファイル **overcloud-nodes.yaml** を作成します。1 つのファイルで複数のノードを登録することが可能です。

```
nodes:
    - name: node0
```

```
    driver: ipmi
    driver_info:
      ipmi_address: <IPMI_IP>
      ipmi_username: <USER>
      ipmi_password: <PASSWORD>
    properties:
      cpus: <CPU_COUNT>
      cpu_arch: <CPU_ARCHITECTURE>
      memory_mb: <MEMORY>
      local_gb: <ROOT_DISK>
      root_device:
          serial: <SERIAL>
    ports:
      - address: <PXE_NIC_MAC>
```

以下の値を置き換えます。

- **<IPMI_IP>** は、Bare Metal コントローラーのアドレスに置き換えます。
- **<USER>** は、ユーザー名に置き換えます。
- **<PASSWORD>** は、パスワードに置き換えます。
- **<CPU_COUNT>** は、CPU の数に置き換えます。
- **<CPU_ARCHITECTURE>** は、CPU のアーキテクチャー種別に置き換えます。
- **<MEMORY>** は、メモリー容量 (MiB 単位) に置き換えます。
- **<ROOT_DISK>** は、ルートディスクの容量 (GiB 単位) に置き換えます。
- **<MAC_ADDRESS>** は、PXE ブートで使用する NIC の MAC アドレスに置き換えます。マシンに複数のディスクがある場合に限り、**root_device** を含める必要があります。**<SERIAL>** は、デプロイメントに使用するディスクのシリアル番号に置き換えます。

2. Identity を管理ユーザーとして使用するためのシェルを設定します。

```
$ source ~/overcloudrc
```

3. インベントリーファイルを ironic にインポートします。

```
$ openstack baremetal create overcloud-nodes.yaml
```

これで、ノードは **enroll** の状態となります。

4. 各ノードでデプロイカーネルとデプロイ ramdisk を指定します。

```
$ openstack baremetal node set NODE_UUID \
  --driver-info deploy_kernel=KERNEL_UUID \
  --driver-info deploy_ramdisk=INITRAMFS_UUID
```

以下の値を置き換えてください。

- NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。

- **KERNEL_UUID** は、Image サービスにアップロードしたカーネルデプロイイメージの一意識別子に置き換えます。この値は以下のコマンドで確認します。

```
$ openstack image show bm-deploy-kernel -f value -c id
```

- **INITRAMFS_UUID** は、Image サービスにアップロードした ramdisk イメージの一意識別子に置き換えます。この値は以下のコマンドで確認します。

```
$ openstack image show bm-deploy-ramdisk -f value -c id
```

5. ノードのプロビジョニング状態を **available** に設定します。

```
$ openstack baremetal node manage _NODE_UUID_
$ openstack baremetal node provide _NODE_UUID_
```

ノードのクリーニングを有効にしている場合には、Bare Metal Provisioning サービスがノードをクリーニングします。

6. ノードにローカルブートオプションを設定します。

```
$ openstack baremetal node set _NODE_UUID_ --property capabilities="boot_option:local"
```

7. ノードが正常に登録されたことを確認します。

```
$ openstack baremetal node list
```

ノードを登録した後にその状態が表示されるまで時間がかかる場合があります。

5.6.2. ベアメタルノードの手動登録

物理ノードをベアメタルノードとして登録してから、手動でハードウェア情報を追加し、各イーサネットの MAC アドレス用にポートを作成します。これらの手順は、**overcloudrc** ファイルがある任意のノードで実行できます。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. Identity を管理ユーザーとして使用するためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. 新しいノードを追加します。

```
$ openstack baremetal node create --driver ipmi --name NAME
```

ノードを作成するには、ドライバー名を指定する必要があります。この例では **ipmi** を使用しています。異なるドライバーを使用するには、**IronicEnabledDrivers** パラメーターを設定してそ

のドライバーを有効にする必要があります。サポートされているドライバーについての詳しい情報は、「10章*Bare Metal のドライバー*」を参照してください。

重要

ノードの一意識別子を書き留めておきます。

3. ノードのドライバーの情報を更新して、Bare Metal Provisioning サービスがノードを管理できるようにします。

```
$ openstack baremetal node set NODE_UUID \
  --driver_info PROPERTY=VALUE \
  --driver_info PROPERTY=VALUE
```

以下の値を置き換えてください。

- NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。
- PROPERTY は、**openstack baremetal driver property list <driver> コマンドが返す必須のプロパティー** に置き換えます。
- VALUE は、プロパティーの有効な値に置き換えます。

4. ノードドライバーのデプロイカーネルとデプロイ ramdisk を指定します。

```
$ openstack baremetal node set NODE_UUID \
  --driver-info deploy_kernel=KERNEL_UUID \
  --driver-info deploy_ramdisk=INITRAMFS_UUID
```

以下の値を置き換えてください。

- NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。
- KERNEL_UUID は、Image サービスにアップロードした **.kernel** イメージの一意識別子に置き換えます。
- INITRAMFS_UUID は、Image サービスにアップロードされた **.initramfs** イメージの一意識別子に置き換えます。

5. ノードの属性を更新して、ノード上のハードウェアの仕様と一致するようにします。

```
$ openstack baremetal node set NODE_UUID \
  --property cpus=CPU \
  --property memory_mb=RAM_MB \
  --property local_gb=DISK_GB \
  --property cpu_arch=ARCH
```

以下の値を置き換えてください。

- NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。
- CPU は、CPU の数に置き換えます。

- **RAM_MB** は、メモリー (MB 単位) に置き換えます。
- **DISK_GB** は、ディスク容量 (GB 単位) に置き換えます。
- **ARCH** は、アーキテクチャー種別に置き換えます。

6. UEFI ノードの場合は、ブートモードを **uefi** に設定します。

```
$ openstack baremetal node set --property capabilities="boot_mode:uefi" <node>
```

- &lt **;node&** gt; をノードの名前に置き換えます。

7. オプション： **ironic-conductor** から PXE を使用する代わりに、ノードのディスクにインストールされたローカルのブートローダーから初回のデプロイメントの後にリブートするようにノードを設定します。ノードのプロビジョニングに使用するフレーバーでも、ローカルブートの機能を設定する必要があります。ローカルブートを有効にするには、ノードのデプロイに使用したイメージに **grub2** が含まれる必要があります。以下のコマンドを実行してローカルブートを設定します。

```
$ openstack baremetal node set NODE_UUID \
  --property capabilities="boot_option:local"
```

NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。

8. プロビジョニングネットワーク上の NIC の MAC アドレスを使用してポートを作成することにより、Bare Metal Provisioning サービスにノードのネットワークカードを通知します。

```
$ openstack baremetal port create --node NODE_UUID MAC_ADDRESS
```

NODE_UUID は、ノードの一意識別子に置き換えます。MAC_ADDRESS は、PXE ブートに使用する NIC の MAC アドレスに置き換えます。

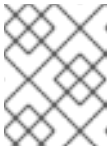
9. 複数のディスクを持つ BIOS ノードの場合は、ルートデバイスのヒントを設定します。これにより、デプロイメントに使用するディスクがデプロイ ramdisk に通知されます。

```
$ openstack baremetal node set NODE_UUID \
  --property root_device={"PROPERTY": "VALUE"}
```

以下の値を置き換えてください。

- NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。
- PROPERTY と VALUE は、デプロイメントに使用するディスクの情報に置き換えます (例: **root_device='{"size": 128}'**)。
  以下のプロパティーがサポートされています。
  - **model** (文字列): デバイスの ID
  - **vendor** (文字列): デバイスのベンダー
  - **serial** (文字列): ディスクのシリアル番号
  - **hctl** (文字列): SCSI のホスト、チャンネル、ターゲット、Lun

- **size** (整数):デバイスのサイズ (GB 単位)
- **wwn** (文字列): 一意のストレージ ID
- **wwn_with_extension** (文字列): ベンダー拡張子を追加した一意のストレージ ID
- **wwn_vendor_extension** (文字列): 一意のベンダーストレージ ID
- **rotational** (ブール値): 回転式デバイス (HDD) には true、そうでない場合 (SSD) には false
- **name** (文字列): デバイス名 (例: /dev/sdb1)。このプロパティーは、永続デバイス名が付いたデバイスにのみ使用してください。



注記

複数のプロパティーを指定する場合には、デバイスはそれらの全プロパティーと一致する必要があります。

10. ノードの設定を検証します。

```
$ openstack baremetal node validate NODE_UUID
+------------+--------+--------------------------------------------+
| Interface  | Result | Reason                                     |
+------------+--------+--------------------------------------------+
| boot       | False  | Cannot validate image information for node  |
|            |        | a02178db-1550-4244-a2b7-d7035c743a9b       |
|            |        | because one or more parameters are missing  |
|            |        | from its instance_info. Missing are:       |
|            |        | ['ramdisk', 'kernel', 'image_source']      |
| console    | None   | not supported                              |
| deploy     | False  | Cannot validate image information for node  |
|            |        | a02178db-1550-4244-a2b7-d7035c743a9b       |
|            |        | because one or more parameters are missing  |
|            |        | from its instance_info. Missing are:       |
|            |        | ['ramdisk', 'kernel', 'image_source']      |
| inspect    | None   | not supported                              |
| management | True   |                                            |
| network    | True   |                                            |
| power      | True   |                                            |
| raid       | True   |                                            |
| storage    | True   |                                            |
+------------+--------+--------------------------------------------+
```

NODE_UUID は、ノードの一意識別子に置き換えます。もしくは、ノードの論理名を使用します。**openstack baremetal node validate** コマンドの出力には、各インターフェースが **True** または **None** のいずれかと報告されます。**None** とマークされたインターフェースは、設定していないか、ドライバーがサポートしていないインターフェースです。

注記

「ramdisk」、「kernel」、および「image_source」のパラメーターが指定されていないと、インターフェースの検証に失敗する場合があります。Compute サービスは、デプロイメントプロセスの最初に未指定のパラメーターを設定するためです。

5.7. REDFISH 仮想メディアブートの設定

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

Redfish 仮想メディアブートを使用して、ノードの Baseboard Management Controller (BMC) にブートイメージを提供することができます。これにより、BMC はイメージを仮想ドライブのいずれかに挿入することができます。その後、ノードは仮想ドライブからイメージに存在するオペレーティングシステムにブートすることができます。

Redfish ハードウェア種別は、仮想メディアを通じたデプロイ、レスキュー、およびユーザーの各イメージのブートに対応しています。Bare Metal Provisioning サービス (ironic) は、ノードのデプロイメント時に、ノードに関連付けられたカーネルイメージおよび ramdisk イメージを使用して、UEFI または BIOS ブートモード用のブート可能 ISO イメージをビルドします。仮想メディアブートの主な利点は、PXE の TFTP イメージ転送フェーズを排除し、HTTP GET 等の方法を使用することができる点です。

5.7.1. Redfish 仮想メディアブートを使用するベアメタルサーバーのデプロイ

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

仮想メディアを通じて **redfish** ハードウェア種別のノードをブートするには、ブートインターフェースを **redfish-virtual-media** に設定し、UEFI ノードの場合は EFI システムパーティション (ESP) イメージを定義します。続いて、登録したノードが Redfish 仮想メディアブートを使用するように設定します。

前提条件

- **undercloud.conf** ファイルの **enabled_hardware_types** パラメーターで、Redfish ドライバーが有効化されている。
- ベアメタルノードが登録されている。
- Image サービス (glance) に IPA およびインスタンスイメージがある。
- UEFI ノードの場合、EFI システムパーティション (ESP) イメージも Image サービス (glance) で利用可能でなければなりません。
- ベアメタルフレーバー
- クリーニングおよびプロビジョニング用ネットワーク
- Sushy ライブラリーがインストールされている。

```
$ sudo yum install sushy
```

手順

1. Bare Metal サービス (ironic) のブートインターフェースを **redfish-virtual-media** に設定します。

```
$ openstack baremetal node set --boot-interface redfish-virtual-media $NODE_NAME
```

$NODE_NAME はノード名に置き換えてください。

2. UEFI ノードの場合は、ブートモードを **uefi** に設定します。

```
$ openstack baremetal node set --property capabilities="boot_mode:uefi" $NODE_NAME
```

$NODE_NAME はノード名に置き換えてください。

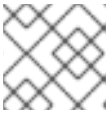
注記

BIOS ノードの場合は、このステップを実施しないでください。

3. UEFI ノードの場合は、EFI システムパーティション (ESP) イメージを定義します。

```
$ openstack baremetal node set --driver-info bootloader=$ESP $NODE_NAME
```

$ESP は glance イメージの UUID または ESP イメージの URL に、 **$NODE_NAME** はノードの名前に、それぞれ置き換えてください。



注記

BIOS ノードの場合は、このステップを実施しないでください。

4. ベアメタルノードにポートを作成し、そのポートをベアメタルノード上の NIC の MAC アドレスに関連付けます。

```
$ openstack baremetal port create --pxe-enabled True --node $UUID $MAC_ADDRESS
```

$UUID はベアメタルノードの UUID に、**$MAC_ADDRESS** はベアメタルノード上の NIC の MAC アドレスに、それぞれ置き換えてください。

5. 新しいベアメタルサーバーを作成します。

```
$ openstack server create \
    --flavor baremetal \
    --image $IMAGE \
    --network $NETWORK \
    test_instance
```

$IMAGE および **$NETWORK** は、使用するイメージおよびネットワークの名前に置き換えます。

5.8. ホストアグリゲートを使用した物理マシンと仮想マシンのプロビジョニングの分離

OpenStack Compute は、ホストアグリゲートを使用してアベイラビリティーゾーンをパーティション分割し、特定の共有属性が指定されたノードをグループ化します。インスタンスがプロビジョニングされると、Compute のスケジューラーがフレーバーのプロパティーをホストアグリゲートに割り当てられたプロパティーと比較して、インスタンスが正しいアグリゲート内の正しいホストに (物理マシン上または仮想マシンとして) プロビジョニングされたことを確認します。

本項の手順を実施して、以下の操作を行います。

- **baremetal** プロパティーをフレーバーに追加して、**true** または **false** に設定する。
- 一致する **baremetal** プロパティーを設定して、ベアメタルホスト用とコンピュートノード用のホストアグリゲートを別々に作成する。1 つのアグリゲートでグループ化されたノードは、このプロパティーを継承します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. ベアメタル用のフレーバーで **baremetal** プロパティーを **true** に設定します。

```
$ openstack flavor set baremetal --property baremetal=true
```

2. 仮想インスタンスに使用するフレーバーで **baremetal** プロパティーを **false** に設定します。

```
$ openstack flavor set FLAVOR_NAME --property baremetal=false
```

3. **baremetal-hosts** という名前のホストアグリゲートを作成します。

```
$ openstack aggregate create --property baremetal=true baremetal-hosts
```

4. 各コントローラーノードを **baremetal-hosts** アグリゲートに追加します。

```
$ openstack aggregate add host baremetal-hosts HOSTNAME
```

注記

NovaIronic サービスでコンポーザブルロールを作成していた場合には、このサービスがあるノードをすべて **baremetal-hosts** アグリゲートに追加します。デフォルトでは、**NovaIronic** サービスがあるのはコントローラーノードのみです。

5. **virtual-hosts** という名前のホストアグリゲートを作成します。

```
$ openstack aggregate create --property baremetal=false virtual-hosts
```

6. 各コンピュートノードを **virtual-hosts** アグリゲートに追加します。

```
$ openstack aggregate add host virtual-hosts HOSTNAME
```

7. オーバークラウドのデプロイ時に以下の Compute フィルタースケジューラーを追加していなかった場合には、この時点で _/**etc**/**nova**/**nova.conf**_ の **scheduler_default_filters** セクションの既存リストに追加します。

```
AggregateInstanceExtraSpecsFilter
```

第6章 ベアメタルノードの管理

Bare Metal Provisioning サービス (ironic) が含まれるオーバークラウドをデプロイしたら、登録済みのベアメタルノードに物理マシンをプロビジョニングして、オーバークラウドでベアメタルインスタンスを起動することができます。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

6.1. ベアメタルインスタンスの起動

コマンドラインまたは OpenStack Dashboard のいずれかで、インスタンスを起動することができます。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

6.1.1. コマンドラインインターフェースを使用したインスタンスの起動

openstack コマンドラインインターフェースを使用してベアメタルインスタンスをデプロイします。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. Identity サービス (keystone) に管理ユーザーとしてアクセスするためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. インスタンスをデプロイします。

```
$ openstack server create \
  --nic net-id=NETWORK_UUID \
  --flavor baremetal \
  --image IMAGE_UUID \
  INSTANCE_NAME
```

以下の値を置き換えてください。

- **NETWORK_UUID** は、Bare Metal Provisioning サービスで使用するために作成したネットワークの一意識別子に置き換えます。
- **IMAGE_UUID** は、Image サービスにアップロードされたディスクイメージの一意識別子に置き換えます。
- **INSTANCE_NAME** は、ベアメタルインスタンスの名前に置き換えます。

セキュリティーグループにインスタンスを割り当てるには、**--security-group SECURITY_GROUP** オプションを指定します。SECURITY_GROUP は、そのセキュリティーグループの名前に置き換えてください。インスタンスを複数のグループに追加するには、このオプションを繰り返します。セキュリティーグループの管理についての詳しい情報は、『Users and Identity Management Guide』を参照してください。

3. インスタンスのステータスを確認します。

```
$ openstack server list --name INSTANCE_NAME
```

6.1.2. Dashboard を使用したインスタンスの起動

Dashboard のグラフィカルユーザーインターフェースを使用してベアメタルインスタンスをデプロイします。

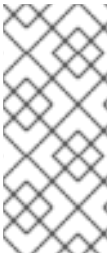
前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. **http[s]://DASHBOARD_IP/dashboard** で Dashboard にログインします。
2. **プロジェクト** > **コンピュート** > **インスタンス**の順にクリックします。
3. **インスタンスの起動** をクリックします。
   - **詳細** タブで **インスタンス名** を指定して、**インスタンス数** に **1** を選択します。
   - **ソース** タブで **ブートソースを選択してください** のドロップダウンメニューから **イメージ** を選択し、続いて ↑ (上向き矢印) の記号をクリックしてオペレーティングシステムのディスクイメージを選択します。選択したイメージが **割り当て済み** に移動します。
   - **フレーバー** タブで **baremetal** を選択します。
   - **ネットワーク** タブで、↑ (上向き矢印) および ↓ (下向き矢印) ボタンを使用して必要なネットワークを **利用可能** から **割り当て済み** に移動します。ここでは、必ず Bare Metal Provisioning サービス用に作成した共有ネットワークを選択してください。
   - インスタンスをセキュリティーグループに割り当てるには、**セキュリティーグループ** タブで矢印を使用してそのグループを **割り当て済み** に移動します。
4. **インスタンスの起動** をクリックします。

6.2. BARE METAL PROVISIONING サービスでのポートグループの設定



注記

ベアメタルノード向けのポートグループ機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的にのみご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

ポートグループ (ボンディング) の機能により、複数のネットワークインターフェースを単一の「ボンディングされた」インターフェースに統合することができます。ポートグループの設定は常に、個別のポート設定に優先します。

ポートグループに物理ネットワークがある場合には、そのポートグループ内の全ポートに同じ物理ネットワークを使用すべきです。Bare Metal Provisioning サービスは、**configdrive** を使用してインスタンスでのポートグループの設定をサポートしています。

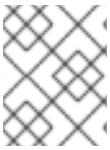
注記

Bare Metal Provisioning サービス API バージョン 1.26 は、ポートグループの設定をサポートしています。前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

6.2.1. 手動によるスイッチ上のポートグループの設定

ベアメタルのデプロイメントでポートグループを設定するには、スイッチ上でポートグループを手動設定する必要があります。スイッチによって名前が異なる場合があるため、スイッチ上のモードとプロパティーが、ベアメタル側のモードとプロパティーに対応している状態にする必要があります。



注記

iPXE を使用してデプロイメントを起動する必要がある場合、プロビジョニングとクリーニングにはポートグループを使用できません。

ポートグループのフォールバック機能により、接続でエラーが発生した際に、1 つのポートグループ内の全ポートを個々のスイッチポートにフォールバックさせることができます。スイッチがポートグループのフォールバックをサポートしているかどうかに応じて、「**--support-standalone-ports**」と「**--unsupport-standalone-ports**」のオプションを使用することができます。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

6.2.2. Bare Metal Provisioning サービスでのポートグループの設定

複数のネットワークインターフェースを単一の **ボンディングインターフェース** に統合するポートグループを作成します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

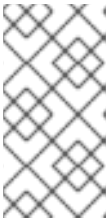
1. ポートグループが属する先のノード、その名前、アドレス、モード、プロパティー、スタンドアロンポートへのフォールバックをサポートするかどうかを指定して、ポートグループを作成します。

```
# openstack baremetal port group create --node NODE_UUID --name NAME --address
MAC_ADDRESS --mode MODE  --property miimon=100 --property
xmit_hash_policy="layer2+3" --support-standalone-ports
```

   また、**openstack baremetal port group set** コマンドを使用してポートグループを更新することもできます。

   アドレスを指定しない場合には、デプロイされるインスタンスのポートグループアドレスは OpenStack Networking のポートと同じになります。neutron ポートを接続しないと、ポートグループの設定は失敗します。

   インターフェースの接続中には、ポートグループの優先度はポートよりも高くなるので、最初に使用されます。現在、インターフェースの接続要求で、ポートグループとポートのどちらを優先するかを指定することはできません。ポートのないポートグループは無視されます。



注記

ポートグループは、手動でスタンドアロンモードに設定する必要があります。そのためには、イメージ内で設定するか、**configdrive** を生成してノードの **instance_info** に追加します。ポートグループの設定が機能するには、**cloud-init** バージョン 0.7.7 以降を使用している必要があります。

2. ポートをポートグループに関連付けます。

   - ポートの作成中

```
# openstack baremetal port create --node NODE_UUID --address MAC_ADDRESS --
port-group test
```

   - ポートの更新中

```
# openstack baremetal port set PORT_UUID --port-group PORT_GROUP_UUID
```

3. **cloud-init** 対応のイメージまたはボンディングをサポートしているイメージを提供することにより、インスタンスを起動します。
   ポートグループが適切に設定されているかを確認するには、以下のコマンドを実行します。

```
# cat /proc/net/bonding/bondX
```

-

X は、**cloud-init** が設定済みの各ポートグループに対して自動生成する番号です。**0** で始まり、ポートグループを設定するたびに 1 つずつ増えます。

6.3. ホストから IP アドレスへのマッピングの確認

各 IP アドレスが割り当てられているホストおよびベアメタルノードを確認するには、以下のコマンドを使用します。これらのコマンドにより、ホストに直接アクセスせずに、ホストから IP へのマッピングをアンダークラウドで確認することが可能です。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. 以下のコマンドを実行して、各ホストの IP アドレスを表示します。

```
(undercloud) [stack@host01 ~]$ openstack stack output show overcloud HostsEntry --max-
width 80

+--------------+------------------------------------------------------------+
| Field        | Value                                                      |
+--------------+------------------------------------------------------------+
| description  | The content that should be appended to your /etc/hosts if you |
|              | want to get                                                |
|              | hostname-based access to the deployed nodes (useful for       |
|              | testing without                                            |
|              | setting up a DNS).                                         |
|              |                                                            |
| output_key   | HostsEntry                                                 |
| output_value | 172.17.0.10 overcloud-controller-0.localdomain overcloud-     |
|              | controller-0                                               |
|              | 10.8.145.18 overcloud-controller-0.external.localdomain       |
|              | overcloud-controller-0.external                            |
|              | 172.17.0.10 overcloud-controller-0.internalapi.localdomain     |
|              | overcloud-controller-0.internalapi                         |
|              | 172.18.0.15 overcloud-controller-0.storage.localdomain        |
|              | overcloud-controller-0.storage                             |
|              | 172.21.2.12 overcloud-controller-0.storagemgmt.localdomain     |
|              | overcloud-controller-0.storagemgmt                         |
|              | 172.16.0.15 overcloud-controller-0.tenant.localdomain         |
|              | overcloud-controller-0.tenant                              |
|              | 10.8.146.13 overcloud-controller-0.management.localdomain      |
|              | overcloud-controller-0.management                          |
|              | 10.8.146.13 overcloud-controller-0.ctlplane.localdomain       |
|              | overcloud-controller-0.ctlplane                            |
|              |                                                            |
|              | 172.17.0.21 overcloud-compute-0.localdomain overcloud-         |
|              | compute-0                                                  |
|              | 10.8.146.12 overcloud-compute-0.external.localdomain          |
|              | overcloud-compute-0.external                               |
```

```
|             | 172.17.0.21 overcloud-compute-0.internalapi.localdomain       |
|             | overcloud-compute-0.internalapi                               |
|             | 172.18.0.20 overcloud-compute-0.storage.localdomain           |
|             | overcloud-compute-0.storage                                   |
|             | 10.8.146.12 overcloud-compute-0.storagemgmt.localdomain       |
|             | overcloud-compute-0.storagemgmt                               |
|             | 172.16.0.16 overcloud-compute-0.tenant.localdomain overcloud- |
|             | compute-0.tenant                                              |
|             | 10.8.146.12 overcloud-compute-0.management.localdomain        |
|             | overcloud-compute-0.management                                |
|             | 10.8.146.12 overcloud-compute-0.ctlplane.localdomain          |
|             | overcloud-compute-0.ctlplane                                  |
|             |                                                               |
|             |                                                               |
|             |                                                               |
|             |                                                               |
|             | 10.8.145.16  overcloud.localdomain                            |
|             | 10.8.146.7  overcloud.ctlplane.localdomain                    |
|             | 172.17.0.19  overcloud.internalapi.localdomain                |
|             | 172.18.0.19  overcloud.storage.localdomain                    |
|             | 172.21.2.16  overcloud.storagemgmt.localdomain                |
+-------------+---------------------------------------------------------------+
```

2. 特定のホストに絞り込むには、以下のコマンドを実行します。

```
(undercloud) [stack@host01 ~]$ openstack stack output show overcloud HostsEntry -c
output_value -f value | grep overcloud-controller-0

172.17.0.12 overcloud-controller-0.localdomain overcloud-controller-0
10.8.145.18 overcloud-controller-0.external.localdomain overcloud-controller-0.external
172.17.0.12 overcloud-controller-0.internalapi.localdomain overcloud-controller-0.internalapi
172.18.0.12 overcloud-controller-0.storage.localdomain overcloud-controller-0.storage
172.21.2.13 overcloud-controller-0.storagemgmt.localdomain overcloud-controller-
0.storagemgmt
172.16.0.19 overcloud-controller-0.tenant.localdomain overcloud-controller-0.tenant
10.8.146.13 overcloud-controller-0.management.localdomain overcloud-controller-
0.management
10.8.146.13 overcloud-controller-0.ctlplane.localdomain overcloud-controller-0.ctlplane
```

3. ホストをベアメタルノードにマッピングするには、以下のコマンドを実行します。

```
(undercloud) [stack@host01 ~]$ openstack baremetal node list --fields uuid name
instance_info -f json
[
  {
    "UUID": "c0d2568e-1825-4d34-96ec-f08bbf0ba7ae",
    "Instance Info": {
      "root_gb": "40",
      "display_name": "overcloud-compute-0",
      "image_source": "24a33990-e65a-4235-9620-9243bcff67a2",
      "capabilities": "{\"boot_option\": \"local\"}",
      "memory_mb": "4096",
      "vcpus": "1",
      "local_gb": "557",
      "configdrive": "******",
```

```
      "swap_mb": "0",
      "nova_host_id": "host01.lab.local"
    },
    "Name": "host2"
  },
  {
    "UUID": "8c3faec8-bc05-401c-8956-99c40cdea97d",
    "Instance Info": {
      "root_gb": "40",
      "display_name": "overcloud-controller-0",
      "image_source": "24a33990-e65a-4235-9620-9243bcff67a2",
      "capabilities": "{\"boot_option\": \"local\"}",
      "memory_mb": "4096",
      "vcpus": "1",
      "local_gb": "557",
      "configdrive": "*******",
      "swap_mb": "0",
      "nova_host_id": "host01.lab.local"
    },
    "Name": "host3"
  }
]
```


6.4. 仮想ネットワークインターフェースの接続と切断

Bare Metal Provisioning サービスには、仮想ネットワークインターフェース間のマッピングを管理するための API があります。たとえば、OpenStack Networking サービスのインターフェースと実際の物理インターフェース (NIC) などです。これらのインターフェースを各 Bare Metal Provisioning ノードに設定して、仮想ネットワークインターフェース (VIF) から物理ネットワークインターフェース (PIF) へのマッピングロジックを設定することができます。インターフェースを設定するには、**openstack baremetal node vif*** コマンドを使用します。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

1. ベアメタルノードに現在接続されている VIF の ID を一覧表示します。

```
$ openstack baremetal node vif list baremetal-0
+--------------------------------------+
| ID                                   |
+--------------------------------------+
| 4475bc5a-6f6e-466d-bcb6-6c2dce0fba16 |
+--------------------------------------+
```

2. VIF がアタッチされた後に、Bare Metal Provisioning サービスは OpenStack Networking サービス内の仮想ポートを実際の物理ポートの MAC アドレスで更新します。このポートアドレスを確認します。

```
$ openstack port show 4475bc5a-6f6e-466d-bcb6-6c2dce0fba16 -c mac_address -c fixed_ips
+-------------+-------------------------------------------------------------------------+
| Field       | Value                                                                   |
+-------------+-------------------------------------------------------------------------+
| fixed_ips   | ip_address='192.168.24.9', subnet_id='1d11c677-5946-4733-87c3-
23a9e06077aa' |
| mac_address | 00:2d:28:2f:8d:95                                                       |
+-------------+-------------------------------------------------------------------------+
```

3. **baremetal-0** ノードを作成したネットワーク上に新規ポートを作成します。

```
$ openstack port create --network baremetal --fixed-ip ip-address=192.168.24.24 baremetal-
0-extra
```

4. インスタンスからポートを削除します。

```
$ openstack server remove port overcloud-baremetal-0 4475bc5a-6f6e-466d-bcb6-
6c2dce0fba16
```

5. その IP アドレスがリストには存在しなくなったことを確認します。

```
$ openstack server list
```

6. そのノードに接続されている VIF があるかどうかを確認します。

```
$ openstack baremetal node vif list baremetal-0
$ openstack port list
```

7. 新規作成されたポートを追加します。

```
$ openstack server add port overcloud-baremetal-0 baremetal-0-extra
```

8. 新しい IP アドレスに新しいポートが表示されることを確認します。

```
$ openstack server list
+--------------------------------------+------------------------+--------+-----------------------+----------
------+---------+
| ID                                   | Name                   | Status | Networks              | Image          |
Flavor  |
+--------------------------------------+------------------------+--------+-----------------------+----------
------+---------+
| 53095a64-1646-4dd1-bbf3-b51cbcc38789 | overcloud-controller-2  | ACTIVE |
ctlplane=192.168.24.7  | overcloud-full | control |
| 3a1bc89c-5d0d-44c7-a569-f2a3b4c73d65 | overcloud-controller-0  | ACTIVE |
ctlplane=192.168.24.8  | overcloud-full | control |
| 6b01531a-f55d-40e9-b3a2-6d02be0b915b | overcloud-controller-1  | ACTIVE |
ctlplane=192.168.24.16 | overcloud-full | control |
| c61cc52b-cc48-4903-a971-073c60f53091 | overcloud-novacompute-0overcloud-baremetal-
0 | ACTIVE | ctlplane=192.168.24.24 | overcloud-full | compute |
+--------------------------------------+------------------------+--------+-----------------------+----------
------+---------+
```

9. VIF ID が新規ポートの UUID であるかどうかを確認します。

```
$ openstack baremetal node vif list baremetal-0
+--------------------------------------+
| ID                                   |
+--------------------------------------+
| 6181c089-7e33-4f1c-b8fe-2523ff431ffc |
+--------------------------------------+
```

10. OpenStack Networking ポートの MAC アドレスが更新され、Bare Metal Provisioning サービスポートの中の 1 つと一致しているかどうかを確認します。

```
$ openstack port show 6181c089-7e33-4f1c-b8fe-2523ff431ffc -c mac_address -c fixed_ips
+-------------+-------------------------------------------------------------------------------+
| Field       | Value                                                                         |
+-------------+-------------------------------------------------------------------------------+
| fixed_ips   | ip_address='192.168.24.24', subnet_id='1d11c677-5946-4733-87c3-
23a9e06077aa' |
| mac_address | 00:2d:28:2f:8d:95                                                             |
+-------------+-------------------------------------------------------------------------------+
```

11. 新規 IP アドレスを認識するように、ベアメタルノードを再起動します。

```
$ openstack server reboot overcloud-baremetal-0
```

インターフェースを接続または切断した後には、ベアメタルの OS は変更されたネットワークインターフェースを削除/追加/変更します。ポートを置き換える場合、DHCP 要求が新規 IP アドレスを取得しますが、古い DHCP リースがまだ有効なので、多少時間がかかる場合があります。変更を即時に適用するには、ベアメタルホストをリブートします。

6.5. BARE METAL PROVISIONING サービスの通知の設定

Bare Metal Provisioning サービス (ironic) を設定して、サービス内で発生するさまざまなイベントの通知を表示することができます。外部サービスは、請求目的、データストアの監視、およびその他の目的でこれらの通知を使用することができます。Bare Metal Provisioning サービスの通知を有効にするには、**ironic.conf** 設定ファイルで以下のオプションを設定する必要があります。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

- **[DEFAULT]** セクションの **notification_level** オプションは、通知送信の最小の優先度を決定します。このオプションの値は、**debug**、**info**、**warning**、**error**、**critical** のいずれかに設定することができます。オプションが **warning** に設定されると、優先度が **warning**、**error**、または **critical** のいずれかである通知はすべて送信されますが、優先度が **debug** または **info** の通知は送信させません。このオプションが設定されていない場合には、通知は一切送信されません。利用可能な各通知の優先度は、以下に記載しています。

- **[oslo_messaging_notifications]** セクションの **transport_url** のオプションは、通知の送信に使用されるメッセージバスを決定します。このオプションが設定されていない場合には、RPC に使われるデフォルトのトランスポートが使用されます。

通知はすべて、メッセージバス内の **ironic_versioned_notifications** トピックで発行されます。通常は、メッセージバスを通過する各種別のメッセージは、メッセージの内容を説明しているトピックに関連付けられます。

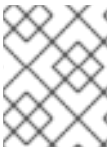
6.6. 電源異常からの自動復帰の設定

Bare Metal Provisioning サービス (ironic) には、ノードの電源、クリーニング、およびレスキューアボートの失敗を記録する文字列フィールド **fault** があります。

表6.1 Ironic ノードの異常

異常	説明
power failure	電源の同期に失敗したため (リトライ回数の最大値の超過)、ノードはメンテナンスモードに移行しています。
clean failure	クリーニング操作に失敗したため、ノードはメンテナンスモードに移行しています。
rescue abort failure	レスキューアボート時のクリーニング操作に失敗したため、ノードはメンテナンスモードに移行しています。
none	異常は発生していません。

Conductor は、このフィールドの値を定期的に確認します。Conductor が **power failure** の状態を検出し、ノードの電源の復旧に成功すると、ノードはメンテナンスモードから抜け出し動作状態に戻ります。



注記

オペレーターが手動でノードをメンテナンスモードに切り替えた場合には、Conductor が自動的にノードをメンテナンスモードから移行させることはありません。

デフォルトの間隔は 300 秒ですが、hieradata を使用して director からこの間隔を設定することができます。

前提条件

- Bare Metal Provisioning サービスが含まれるオーバークラウドのデプロイメントが正常に完了していること。詳細な情報は、「*3章Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「*4章Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

手順

- 以下の hieradata を追加して、カスタムの復帰間隔を設定します。

```
ironic::conductor::power_failure_recovery_interval
```

  電源異常からの自動復帰を無効にするには、値を **0** に設定します。

6.7. オーバークラウドノードのイントロスペクション

オーバークラウドノードのイントロスペクションを実施して、director でノードの仕様を特定して保存します。

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. source コマンドで **overcloudrc** 認証情報ファイルを読み込みます。

```
$ source ~/overcloudrc
```

3. イントロスペクションコマンドを実行します。

```
$ openstack baremetal introspection start [--wait] <NODENAME>
```

<NODENAME> を、検査するノードの名前または UUID に置き換えます。

4. イントロスペクションのステータスを確認します。

```
$ openstack baremetal introspection status <NODENAME>
```

<NODENAME> は、ノードの名前または UUID に置き換えます。

次のステップ

- イントロスペクションデータの抽出：

```
$ openstack baremetal introspection data save <NODE-UUID>
```

<NODENAME> は、ノードの名前または UUID に置き換えます。

第7章 CINDER ボリュームからのブート

Block Storage サービス (cinder) にボリュームを作成し、これらのボリュームを Bare Metal Provisioning サービス (ironic) で作成するベアメタルインスタンスに接続することができます。

7.1. ベアメタルノード向けの CINDER ボリュームブート

OpenStack Block Storage (cinder) に保管されるブロックストレージデバイスからベアメタルノードをブートすることができます。OpenStack Bare Metal (ironic) は、iSCSI インターフェースを介してベアメタルノードをボリュームに接続します。

ironic は、オーバークラウドのデプロイメント時にこの機能を有効にします。ただし、オーバークラウドをデプロイする前に、以下の条件を考慮してください。

- オーバークラウドでは、cinder iSCSI バックエンドを有効にする必要があります。オーバークラウドのデプロイメント時に **CinderEnableIscsiBackend** heat パラメーターを **true** に設定します。
- Red Hat Ceph Storage バックエンドでは、cinder ボリュームブート機能を使用することはできません。
- ブートディスクで **rd.iscsi.firmware=1** カーネルパラメーターを設定する必要があります。

7.2. CINDER ボリュームブート用ノードの設定

cinder ボリュームから正常に起動するには、各ベアメタルノードで特定のオプションを設定する必要があります。

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. source コマンドでオーバークラウドの認証情報を読み込みます。

```
$ source ~/overcloudrc
```

3. **iscsi_boot** 機能を **true** に、**storage-interface** を選択したノードの **cinder** に、それぞれ設定します。

```
$ openstack baremetal node set --property capabilities=iscsi_boot:true --storage-interface cinder <NODEID>
```

<NODEID> を選択したノードの ID に置き換えてください。

4. ノードの iSCSI コネクターを作成します。

```
$ openstack baremetal volume connector create --node <NODEID> --type iqn --connector-id iqn.2010-10.org.openstack.node<NUM>
```

各ノードのコネクター ID は一意でなければなりません。この例では、コネクターは **iqn.2010-10.org.openstack.node<NUM>** です。ここで、**<NUM>** は各ノードの通し番号です。

7.3. ブートディスクでの ISCSI カーネルパラメーターの設定

イメージ上のカーネルで iSCSI ブートを有効にする必要があります。そのためには、QCOW2 イメージをマウントし、イメージ上で iSCSI コンポーネントを有効にします。

前提条件

1. Red Hat Enterprise Linux QCOW2 イメージをダウンロードして、アンダークラウドの **/home/stack/** ディレクトリーにコピーします。以下のページから、QCOW2 形式で Red Hat Enterprise Linux KVM イメージをダウンロードすることができます。
   - Red Hat Enterprise Linux 7
   - Red Hat Enterprise Linux 8

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. QCOW2 イメージをマウントし、**root** ユーザーとしてアクセスします。
   a. **nbd** カーネルモジュールを読み込みます。

   ```
   $ sudo modprobe nbd
   ```

   b. QCOW イメージを **/dev/nbd0** として接続します。

   ```
   $ sudo qemu-nbd --connect=/dev/nbd0 <IMAGE>
   ```

   c. NBD 上のパーティションを確認します。

   ```
   $ sudo fdisk /dev/nbd0 -l
   ```

   新しい Red Hat Enterprise Linux QCOW2 イメージには、パーティションが 1 つだけ含まれます。通常、そのパーティションは NBD の **/dev/nbd0p1** という名前です。

   d. イメージのマウントポイントを作成します。

   ```
   mkdir /tmp/mountpoint
   ```

   e. イメージをマウントします。

   ```
   sudo mount /dev/nbd0p1 /tmp/mountpoint/
   ```

   f. イメージがホストのデバイス情報にアクセスできるように、**dev** ディレクトリーをマウントします。

   ```
   sudo mount -o bind /dev /tmp/mountpoint/dev
   ```

   g. ルートディレクトリーをマウントポイントに変更します。

   ```
   sudo chroot /tmp/mountpoint /bin/bash
   ```

3. イメージ上で iSCSI を設定します。

注記

このステップの一部のコマンドにより、以下のエラーが返される場合があります。

```
lscpu: cannot open /proc/cpuinfo: No such file or directory
```

このエラーは重要ではないので、エラーを無視して構いません。

a. **resolv.conf** ファイルを一時的な場所に移動します。

```
# mv /etc/resolv.conf /etc/resolv.conf.bak
```

b. Red Hat コンテンツ配信ネットワークの DNS 要求を解決するために、一時的な **resolv.conf** ファイルを作成します。以下の例では、ネームサーバーに **8.8.8.8** を使用しています。

```
# echo "nameserver 8.8.8.8" > /etc/resolv.conf
```

c. マウントしたイメージを Red Hat コンテンツ配信ネットワークに登録します。

```
# subscription-manager register
```

コマンドにより要求されたら、ユーザー名およびパスワードを入力します。

d. Red Hat Enterprise Linux が含まれるサブスクリプションをアタッチします。

```
# subscription-manager list --all --available
# subscription-manager attach --pool <POOLID>
```

<POOLID> をサブスクリプションのプール ID に置き換えます。

e. デフォルトのリポジトリーを無効にします。

```
# subscription-manager repos --disable "*"
```

f. Red Hat Enterprise Linux リポジトリーを有効にします。

- Red Hat Enterprise Linux 7:

```
# subscription-manager repos --enable "rhel-7-server-rpms"
```

- Red Hat Enterprise Linux 8:

```
# subscription-manager repos --enable "rhel-8-for-x86_64-baseos-eus-rpms"
```

g. **iscsi-initiator-utils** パッケージをインストールします。

```
# yum install -y iscsi-initiator-utils
```

h. マウントしたイメージの登録を解除します。

```
# subscription-manager unregister
```

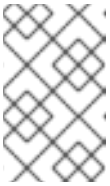
i. 元の **resolv.conf** ファイルを復元します。

```
# mv /etc/resolv.conf.bak /etc/resolv.conf
```

j. マウントされたイメージのカーネルバージョンを確認します。

```
# rpm -qa kernel
```

たとえば、出力が **kernel-3.10.0-1062.el7.x86_64** の場合、カーネルバージョンは **3.10.0-1062.el7.x86_64** になります。次のステップのために、このカーネルバージョンを書き留めておきます。



注記

新しい Red Hat Enterprise Linux QCOW2 イメージには、1 つのカーネルバージョンしかインストールされません。複数のカーネルバージョンがインストールされている場合は、最新のものを使用してください。

k. initramfs イメージに **network** および **iscsi** dracut モジュールを追加します。

```
# dracut --force --add "network iscsi" /boot/initramfs-<KERNELVERSION>.img
<KERNELVERSION>
```

<KERNELVERSION> を **rpm -qa kernel** から取得したバージョン番号に置き換えます。以下の例では、カーネルバージョンに **3.10.0-1062.el7.x86_64** を使用しています。

```
# dracut --force --add "network iscsi" /boot/initramfs-3.10.0-1062.el7.x86_64.img 3.10.0-
1062.el7.x86_64
```

l. マウントされたイメージからホストオペレーティングシステムに戻ります。

```
# exit
```

4. イメージをアンマウントします。

a. 一時的なマウントポイントから **dev** ディレクトリーをアンマウントします。

```
$ sudo umount /tmp/mountpoint/dev
```

b. マウントポイントからイメージをアンマウントします。

```
$ sudo umount /tmp/mountpoint
```

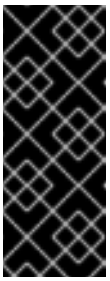
c. QCOW2 イメージを **/dev/nbd0/** から切断します。

```
$ sudo qemu-nbd --disconnect /dev/nbd0
```

5. イメージ上で **grub** メニュー設定を再ビルドします。

a. **libguestfs-tools** パッケージをインストールします。

```
$ sudo yum -y install libguestfs-tools
```



重要

アンダークラウドに **libguestfs-tools** パッケージをインストールする場合は、アンダークラウドの **tripleo_iscsid** サービスとのポートの競合を避けるために **iscsid.socket** を無効にします。

```
$ sudo systemctl disable --now iscsid.socket
```

b. QEMU を直接使用するように **libguestfs** バックエンドを設定します。

```
$ export LIBGUESTFS_BACKEND=direct
```

c. イメージ上の grub 設定を更新します。

```
$ guestfish -a /tmp/images/{{ dib_image }} -m /dev/sda3 sh "mount /dev/sda2 /boot/efi &&
rm /boot/grub2/grubenv && /sbin/grub2-mkconfig -o /boot/grub2/grub.cfg && cp
/boot/grub2/grub.cfg /boot/efi/EFI/redhat/grub.cfg && grubby --update-kernel=ALL --
args=\"rd.iscsi.firmware=1\" && cp /boot/grub2/grubenv /boot/efi/EFI/redhat/grubenv &&
echo Success"
```

7.4. CINDER でのブートボリュームの作成および使用

iSCSI 対応イメージを OpenStack Image Storage (glance) にアップロードして、OpenStack Block Storage (cinder) にブートボリュームを作成する必要があります。

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. iSCSI 対応イメージを glance にアップロードします。

```
$ openstack image create --disk-format qcow2 --container-format bare --file rhel-server-7.7-
x86_64-kvm.qcow2 rhel-server-7.7-iscsi
```

3. イメージからボリュームを作成します。

```
$ openstack volume create --size 10 --image rhel-server-7.7-iscsi --bootable rhel-test-volume
```

4. cinder のブートボリュームを使用するベアメタルインスタンスを作成します。

```
$ openstack server create --flavor baremetal --volume rhel-test-volume --key default rhel-test
```

第8章 ML2 NETWORKING-ANSIBLE

Networking サービス (neutron) を使用してオーバークラウド上で **networking-ansible** ML2 ドライバーを有効にして設定し、それを Bare Metal Provisioning サービス (ironic) と統合することができます。

8.1. MODULAR LAYER 2 (ML2) NETWORKING-ANSIBLE

OpenStack Networking (neutron) に含まれる **networking-ansible** は、Ansible Engine Networking を使用してネットワークスイッチを管理する ML2 ドライバーです。また、このドライバーは、OpenStack Bare Metal (ironic) と統合して、ベアメタルゲスト用にスイッチポート上の VLAN 設定も行います。つまり、VLAN neutron ネットワークを使用するベアメタルゲストによって、このドライバーは Ansible Engine Networking を使用して物理スイッチを設定します。

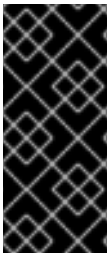
現在の **networking-ansible** ドライバーには、以下の機能が備わっています。

- Red Hat OpenStack Platform (RHOSP) でネットワークを作成する際に、スイッチに VLAN を定義する
- RHOSP でポートを作成または更新する際に、スイッチ上のアクセスポートに VLAN を割り当てる
- RHOSP でポートを削除する際に、スイッチ上のアクセスポートから VLAN を削除する

8.2. NETWORKING-ANSIBLE のネットワーク要件

networking-ansible 機能を有効にするには、お使いの環境に以下のネットワーク設定が含まれている必要があります。

- Ansible Network Automation 対応のネットワークスイッチ。
  - Juniper Networks (**junos**)
  - Arista Extensible Operating System (**eos**)



重要

Arista Extensible Operating System (**eos**) のサポートは、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

- ネットワークスイッチには、Ansible Network Automation がデバイスと対話できるようにするため、SSH ユーザーが必要です。このユーザーには、スイッチに対する以下の権限が必要です。
  - アクセスモード
  - VLAN のポートへの割り当て
  - VLAN の作成

  セキュリティー上の理由から、SSH ユーザーにはスイッチへの管理者アクセス権限を付与しないでください。

- スイッチが使用する VLAN を準備します。VLAN を準備するには、スイッチに各 VLAN を作成してから各 VLAN を削除します。
- ベアメタルゲスト用に予約済みのネットワークスイッチポートは、初めに、イントロスペクション専用のネットワークに接続するよう設定する必要があります。これ以外では、これらのポートに追加設定は必要ありません。

8.3. NETWORKING-ANSIBLE 用の OPENSTACK BARE METAL (IRONIC) の要件

networking-ansible ドライバーは、Openstack Bare Metal (ironic) サービスと統合します。正常に統合するためには、以下の推奨事項に従ってオーバークラウドに Bare Metal Provisioning サービス (ironic) をデプロイします。

- オーバークラウドには、プロビジョニングネットワークが必要です。以下のいずれかのオプションを使用します。
  - ironic サービス用のブリッジネットワーク
  - ironic サービス用のカスタムコンポーザブルネットワーク

  プロビジョニングネットワークの設定の詳細については、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」または「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。
- オーバークラウドには、プロビジョニングプロセスの後に使用するベアメタルシステム用のテナントネットワークが必要です。本ガイドの例では、**br-baremetal** という名前のブリッジにマッピングされた、デフォルトの **baremetal** ネットワークを使用します。このネットワークには、VLAN ID の範囲も必要です。以下の heat パラメーターセットは、本ガイドの例に合わせてこれらの値を設定します。

```
parameter_defaults:
  NeutronNetworkVLANRanges: baremetal:1200:1299
  NeutronFlatNetworks: datacentre,baremetal
  NeutronBridgeMappings: datacentre:br-ex,baremetal:br-baremetal
```

- オーバークラウドはイントロスペクションサービスを使用して、特定のハードウェア情報を自動的に識別し、他のサービスで使用できるようマッピングします。マッピングしたインターフェースとポート間の詳細情報を **networking-ansible** が使用できるように、ironic イントロスペクションサービスを有効にすることを推奨します。このタスクは手動で行うこともできます。

Bare Metal Provisioning サービス (ironic) のデプロイについての詳しい情報は、「3章*Bare Metal Provisioning サービスを有効にした IPv4 オーバークラウドのデプロイ*」および「4章*Bare Metal Provisioning サービスを有効にした IPv6 オーバークラウドのデプロイ*」を参照してください。

8.4. NETWORKING-ANSIBLE ML2 機能の有効化

オーバークラウドで **networking-ansible** ML2 ドライバーを有効にするには、デプロイメントに 2 つの環境ファイルを追加する必要があります。

/usr/share/openstack-tripleo-heat-templates/environments/neutron-ml2-ansible.yaml

このファイルは、**networking-ansible** ドライバーを有効にし、ネットワーク種別を **vlan** に設定します。このファイルは、コア heat テンプレートコレクションにすでに存在します。

/home/stack/templates/ml2-ansible-hosts.yaml

スイッチの詳細情報が含まれるファイルです。このファイルは手動で作成します。

手順

1. **/home/stack/templates/ml2-ansible-hosts.yaml** を作成し、以下の初期コンテンツを追加します。

```
parameter_defaults:
  ML2HostConfigs:
```

2. **ML2HostConfigs** パラメーターには、スイッチの詳細情報が含まれる **dict** 値が必要です。**dict** の各初期キーは、スイッチの名前です。この値によって、OpenStack Networking (neutron) ML2 設定に特定の **ansible:[switchname]** セクションが定義されます。各スイッチ名のキーには、実際のスイッチの詳細情報が含まれる個別の **dict** が必要です。たとえば、スイッチを 3 つ設定する場合、スイッチキーを 3 つ追加します。

```
parameter_defaults:
  ML2HostConfigs:
    switch1:
      [SWITCH DETAILS]
    switch2:
      [SWITCH DETAILS]
    switch3:
      [SWITCH DETAILS]
```

3. 各スイッチには、**dict** 内に特定のキー値ペアが必要です。

ansible_network_os

(必須) スイッチのオペレーティングシステム。選択肢は、**junos** と **eos** です。

重要

Arista Extensible Operating System (**eos**) のサポートは、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

ansible_host

(必須) スイッチの IP またはホスト名。

ansible_user

(必須) Ansible がスイッチにアクセスする際に使用するユーザー。

ansible_ssh_pass

(必須) Ansible がスイッチにアクセスするために使用する SSH パスワード。

mac

ネットワークデバイスのシャーシ MAC ID。これを使用して、Link Layer Discovery Protocol (LLDP) MAC アドレス値を、**ML2HostConfigs** 設定で定義されたスイッチ名にマッピングします。この値は、イントロスペクションを使用してポート自動設定を実行する際に必要です。

manage_vlans

OpenStack Networking (neutron) が物理デバイス上の VLAN の作成と削除を制御するかど

うかを定義するブール型変数。この機能によって、スイッチは各 Neutron ネットワークに対応する ID を持つ VLAN を作成および削除します。スイッチにこれらの VLAN が事前定義されていて、Neutron でスイッチに VLAN を作成したり削除したりする必要がない場合は、このパラメーターを **false** に設定します。デフォルト値は **true** です。

4. 以下の例は、全 **ML2HostConfigs** パラメーターで、これらの値を対応するキーにマッピングする方法を示しています。

```
parameter_defaults:
  ML2HostConfigs:
    switch1:
      ansible_network_os: juno
      ansible_host: 10.0.0.1
      ansible_user: ansible
      ansible_ssh_pass: "p@55w0rd!"
      mac: 01:23:45:67:89:AB
      manage_vlans: false
```

5. **/home/stack/templates/ml2-ansible-hosts.yaml** ファイルを保存します。

6. オーバークラウドのデプロイメントコマンドの実行時に、**-e** オプション指定して **/usr/share/openstack-tripleo-heat-templates/environments/neutron-ml2-ansible.yaml** と **/home/stack/templates/ml2-ansible-hosts.yaml** ファイルを追加します。以下の例で、これらのファイルの追加方法を説明します。

```
$ openstack overcloud deploy --templates \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments/neutron-ml2-ansible.yaml \
  -e /home/stack/templates/ml2-ansible-hosts.yaml \
  ...
```

director は、**neutron_api** コンテナーの OpenStack Networking (neutron) API の一部としてドライバーを有効にします。

8.5. NETWORKING-ANSIBLE 用ネットワーク設定

Bare Metal Provisioning および **networking-ansible** ドライバーを有効にしてオーバークラウドをデプロイしたら、ベアメタルノード用にプロビジョニングネットワークおよびテナントネットワークを作成する必要があります。要件に応じて、アクセスモードかトランクモードのいずれかで、ベアメタルノード用のポートも設定する必要があります。

アクセスモード

アクセスモードでは、スイッチポートは 1 つの VLAN のトラフィックしか伝送せず、単一のブロードキャストドメインで機能します。アクセスポートに到達するすべてのトラフィックは、ポートに割り当てられた VLAN に属します。

トランクモード

トランクモードでは、スイッチポートは複数の VLAN に属することができます。トランクモードでスイッチポートを使用すると、複数 VLAN のトラフィックを伝送することができます。つまり、複数の VLAN が設定された複数スイッチ間のトラフィックを交換することができます。

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

Bare Metal サービス (ironic) は **networking-ansible** を使用してベアメタルゲストのスイッチポートを ironic プロビジョニングネットワークに割り当てます。これにより、プロビジョニングプロセスは正常に完了することができます。プロビジョニングが完了すると、ironic はベアメタルゲストのスイッチポートを、Networking サービス (neutron) によってベアメタルゲストのテナントネットワークに割り当てられる VLAN に割り当てます。

8.5.1. アクセスモードでの networking-ansible 用ネットワーク設定

Bare Metal Provisioning および **networking-ansible** ドライバーを有効にしてオーバークラウドをデプロイしたら、ベアメタルノード用に以下のネットワークを作成します。

プロビジョニングネットワーク

ベアメタルシステムは、このネットワークを初期作成に使用します。

テナントネットワーク

ベアメタルシステムは、プロビジョニング後にこのネットワークに切り替え、このネットワークを内部通信に使用します。

手順

1. プロビジョニングネットワークおよびサブネットを作成します。この操作は、使用しているプロビジョニングネットワークの種別により異なります。プロビジョニングネットワークの設定の詳細については、「*5章 デプロイ後の Bare Metal Provisioning サービスの設定*」を参照してください。

2. テナントネットワークおよびサブネットを作成します。

```
$ openstack network create --provider-network-type vlan --provider-physical-network
baremetal tenant-net
$ openstack subnet create --network tenant-net --subnet-range 192.168.3.0/24 --allocation-
pool start=192.168.3.10,end=192.168.3.20 tenant-subnet
```

networking-ansible が機能するように、必ず **--provider-network-type** オプションを **vlan** に設定してください。

8.5.2. アクセスモードでのベアメタルゲスト用ポート設定

ベアメタルゲストには、スイッチに接続するためのポート情報が必要です。この操作には、2 つの方式があります。

- **自動:** ノードのイントロスペクション。自動方式では、各スイッチの **mac** 値を **ML2HostConfigs** パラメーターの一部として設定する必要があります。

- **手動:** OpenStack Networking (neutron) ポート設定の定義。お使いのオーバークラウドにベアメタルイントロスペクション機能がない場合には、この手法を使用します。

手順

- **自動:**

  a. イントロスペクションコマンドを実行します。

  ```
  $ openstack baremetal introspection start [--wait] <NODENAME>
  ```

  イントロスペクション中に、ベアメタルノードはスイッチの MAC アドレスを取得します。**networking-ansible** ML2 ドライバーはこの MAC アドレスを使用して、各スイッチの **ML2HostConfigs** パラメーターの **mac** パラメーターに定義されたものと同じ MAC アドレスにマッピングします。

  b. イントロスペクションが完了するまで待ちます。

- **手動:**

  1. ベアメタルノードのポートを作成します。以下のコマンド例を、ポート作成のベースとして使用します。

  ```
  $ openstack baremetal port create [NODE NIC MAC] --node [NODE UUID] \
    --local-link-connection port_id=[SWITCH PORT ID] \
    --local-link-connection switch_info=[SWITCH NAME] \
    --local-link-connection switch_id=[SWITCH MAC]
  ```

  以下の大かっこ内の値は、実際の環境の情報に置き換えてください。

  [NODE NIC MAC]

  スイッチに接続された NIC の MAC アドレス

  --node [NODE UUID]

  新しいポートを使用するノードの UUID。

  --local-link-connection port_id=[SWITCH PORT ID]

  ベアメタルノードに接続するスイッチ上のポート ID

  --local-link-connection switch_info=[SWITCH NAME]

  ベアメタルノードに接続するスイッチの名前。スイッチ名は、**ML2HostConfigs** パラメーターで定義する各スイッチ名と一致していなければなりません。

  --local-link-connection switch_id=[SWITCH MAC]

  スイッチの MAC アドレス。この値は、**ML2HostConfigs** パラメーターのスイッチ設定の各 **mac** 値と一致していなければなりません。これは、**switch_info** の使用に対する代替オプションです。

8.5.3. トランクモードでの networking-ansible 用ネットワーク設定

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

Bare Metal Provisioning および **networking-ansible** ドライバーを有効にしてオーバークラウドをデプロイしたら、ベアメタルノード用に以下のネットワークを作成します。

プロビジョニングネットワーク

ベアメタルシステムは、このネットワークを初期作成に使用します。

テナントネットワーク

ベアメタルシステムは、プロビジョニング後にこのネットワークに切り替え、このネットワークを内部通信に使用します。

手順

1. プロビジョニングネットワークおよびサブネットを作成します。この操作は、使用しているプロビジョニングネットワークの種別により異なります。プロビジョニングネットワークの設定の詳細については、「5章 *デプロイ後の Bare Metal Provisioning サービスの設定*」を参照してください。

2. プライマリーテナント VLAN ネットワーク、セカンダリーテナントネットワーク、ならびにゲストが接続された物理ネットワークを使用する各ネットワークのサブネットを作成します。

```
$ openstack network create --provider-network-type vlan --provider-physical-network
baremetal primary-tenant-net
$ openstack network create --provider-network-type vlan --provider-physical-network
baremetal secondary-tenant-net
$ openstack subnet create --network primary-tenant-net --subnet-range 192.168.3.0/24 --
allocation-pool start=192.168.3.10,end=192.168.3.20 primary-tenant-subnet
$ openstack subnet create --network secondary-tenant-net --subnet-range 192.168.7.0/24 --
allocation-pool start=192.168.7.10,end=192.168.7.20 secondary-tenant-subnet
```

networking-ansible が機能するように、必ず **--provider-network-type** オプションを **vlan** に設定してください。

8.5.4. トランクモードでのベアメタルゲスト用ポート設定

重要

この機能は、本リリースでは **テクノロジープレビュー** として提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

ベアメタルゲストには、スイッチに接続するためのポート情報が必要です。これにより、Bare Metal Provisioning サービス (ironic) を使用して、単一のスイッチポートを使用して複数のネットワークにデプロイすることができます。このスイッチポートはトランクモードで設定され、指定されたネットワークから Networking サービス (neutron) が割り当てる VLAN が使用されます。

ベアメタルゲスト用にトランクポートを設定するには、以下の手順を行います。

手順

1. ポートとトランクを作成し、ポートを親ポートとしてトランクに割り当てます。

```
$ port create --network primary-tenant-net primary-port
$ network trunk create --parent-port primary-port my-trunk
```

2. セカンダリーネットワークのポートを作成し、新しいポートをトランクにサブポートとして追加します。

```
$ port create --network secondary-tenant-net secondary-port
$ network trunk set --subport port=secondary-port,segmentation-type=vlan,segmentation-id=1234 my-trunk
```

8.6. NETWORKING-ANSIBLE ML2 機能のテスト

ベアメタルノードの **networking-ansible** 設定が完了したら、ベアメタル用の負荷を作成し、設定が正しいことを確認します。

前提条件

- OpenStack Baremetal (ironic) サービスが設定されたオーバークラウド。
- 有効な **networking-ansible** ML2 ドライバー。
- スイッチのアクセス情報が含まれる **ML2HostConfigs** パラメーター。
- 登録済みのベアメタルノード。
- スイッチ上のノード接続に使用する各ベアメタルポートの設定。このポートは、アクセスポートまたはトランクポートのいずれかです。
- 初期プロビジョニング用に OpenStack Networking (neutron) で定義された VLAN ベースのプロビジョニングネットワーク。
- 内部通信用に OpenStack Networking (neutron) で定義された VLAN ベースのテナントネットワーク。
- オーバークラウドで利用可能なディスクイメージとキーペア。

手順

1. ベアメタルシステムを作成します。
   - アクセスポートを使用するベアメタルシステムを作成するには、以下のコマンドを実行します。

     ```
     openstack server create --flavor baremetal --image overcloud-full --key default --network tenant-net test1
     ```

   - トランクポートを使用するベアメタルシステムを作成するには、以下のコマンドを実行します。

     ```
     openstack server create --flavor baremetal --image overcloud-full --port {primary-port-uuid} --key default test1
     ```

オーバークラウドは、まずプロビジョニングネットワークにベアメタルシステムを作成します。作成が完了すると、**networking-ansible** ドライバーによってスイッチ上のポート設定が変更され、ベアメタルシステムがテナントネットワークを使用するようになります。

第9章 BARE METAL PROVISIONING サービスのトラブルシューティング

Bare Metal Provisioning サービス (ironic) が含まれる環境内の問題を診断します。

9.1. PXE ブートエラー

PXE ブートで直面する問題を評価し、修正するには、以下のトラブルシューティング手順を使用します。

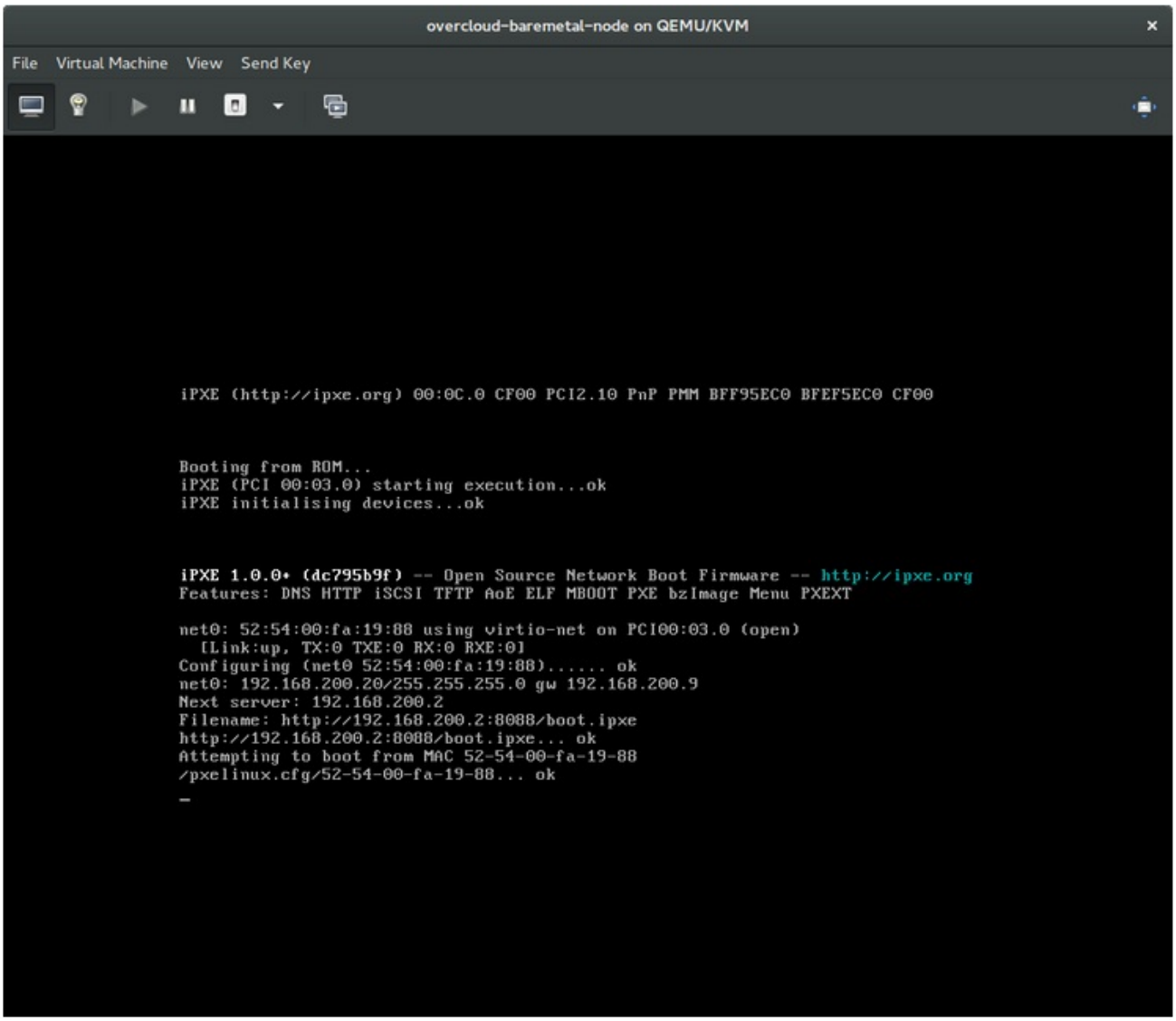
Permission Denied エラー

ベアメタルノードのコンソールで **Permission Denied** エラーが表示された場合には、以下に示すように、必ず適切な SELinux コンテキストを **/httpboot** および **/tftpboot** ディレクトリーに適用してください。

```
# semanage fcontext -a -t httpd_sys_content_t "/httpboot(/.*)?"
# restorecon -r -v /httpboot
# semanage fcontext -a -t tftpdir_t "/tftpboot(/.*)?"
# restorecon -r -v /tftpboot
```

/pxelinux.cfg/XX-XX-XX-XX-XX-XX でのブートプロセスのフリーズ

ノードのコンソールで、IP アドレスは取得しているがプロセスが停止しているように表示されている場合は、**ironic.conf** ファイルで誤った PXE ブートテンプレートを使用している可能性があります。



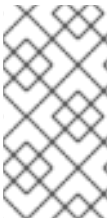

```
$ grep ^pxe_config_template ironic.conf
pxe_config_template=$pybasedir/drivers/modules/ipxe_config.template
```

デフォルトのテンプレートは **pxe_config.template** です。

9.2. ベアメタルノードブート後のログインエラー

設定時に指定した root パスワードを使用してノードにログインできない場合、デプロイされたイメージにブートしていないことを示しています。**deploy-kernel/deploy-ramdisk** イメージにログインしており、システムが正しい イメージを読み込んでいない可能性があります。

この問題を修正するには、Compute または Bare Metal Provisioning サービスノードの **/httpboot/pxelinux.cfg/MAC_ADDRESS** にある PXE ブートの設定ファイルをチェックして、このファイルにリストされている全 IP アドレスがベアメタルネットワークの IP アドレスに対応していることを確認してください。



注記

Bare Metal Provisioning サービスノードが使用する唯一のネットワークは、ベアメタルネットワークです。エンドポイントの 1 つがこのネットワーク上にない場合には、そのエンドポイントはブートプロセスの一環として Bare Metal Provisioning サービスノードに到達することはできません。

たとえば、ファイルの kernel の行は以下のようになります。

```
kernel http://192.168.200.2:8088/5a6cdbe3-2c90-4a90-b3c6-85b449b30512/deploy_kernel selinux=0
disk=cciss/c0d0,sda,hda,vda iscsi_target_iqn=iqn.2008-10.org.openstack:5a6cdbe3-2c90-4a90-b3c6-
85b449b30512 deployment_id=5a6cdbe3-2c90-4a90-b3c6-85b449b30512
deployment_key=VWDYDVVEFCQJNOSTO9R67HKUXUGP77CK
ironic_api_url=http://192.168.200.2:6385 troubleshoot=0 text nofb nomodeset vga=normal
boot_option=netboot ip=${ip}:${next-server}:${gateway}:${netmask} BOOTIF=${mac}  ipa-api-
url=http://192.168.200.2:6385 ipa-driver-name=ipmi boot_mode=bios initrd=deploy_ramdisk
coreos.configdrive=0 || goto deploy
```

上記の例の kernel 行の値	対応する情報
http://192.168.200.2:8088	**/etc/ironic/ironic.conf** ファイルのパラメーター **http_url**。この IP アドレスはベアメタルネットワーク上にある必要があります。
5a6cdbe3-2c90-4a90-b3c6-85b449b30512	**ironic node-list** 内のベアメタルノードの UUID。
deploy_kernel	これは、**/httpboot/<NODE_UUID>/deploy_kernel** としてコピーされた Image サービス内のデプロイカーネルイメージです。
http://192.168.200.2:6385	**/etc/ironic/ironic.conf** ファイル内のパラメーター **api_url**。この IP アドレスはベアメタルネットワーク上にある必要があります。
ipmi	このノードの Bare Metal Provisioning サービスが使用している IPMI ドライバー
deploy_ramdisk	これは、**/httpboot/<NODE_UUID>/deploy_ramdisk** としてコピーされた Image サービス内のデプロイ ramdisk イメージです。

/httpboot/pxelinux.cfg/MAC_ADDRESS と **ironic.conf** ファイルの間で値が一致していない場合:

1. **ironic.conf** ファイル内の値を更新します。
2. Bare Metal Provisioning サービスを再起動します。
3. ベアメタルインスタンスを再デプロイします。

9.3. デプロイされたノードでの BOOT-TO-DISK エラー

特定のハードウェアでは、デプロイされたノードで問題が発生し、デプロイメントの一環としての以降のブート操作中にノードがディスクからブートできない可能性があります。これは通常、director がノードで要求する永続的なブート設定を BMC が反映しないために発生します。代わりに、ノードは PXE ターゲットからブートします。

この場合、ノードの BIOS でブート順序を更新する必要があります。HDD を最初のブートデバイスに設定し、次に PXE を非デフォルトのオプションに設定します。これにより、デフォルトではノードはディスクからブートし、必要に応じてイントロスペクションまたはデプロイメント時にネットワークからブートすることができます。

注記

このエラーは、LegacyBIOS ファームウェアを使用するほとんどのノードに該当します。

9.4. BARE METAL PROVISIONING サービスが正しいホスト名を受信しない

Bare Metal Provisioning サービスが正しいホスト名を受信しない場合は、**cloud-init** でエラーが発生していることを意味します。この問題を修正するには、ベアメタルのサブネットを OpenStack Networking サービス内のルーターに接続します。この設定により、要求が meta-data エージェントに正しくルーティングされます。

9.5. BARE METAL PROVISIONING サービスのコマンド実行時に OPENSTACK IDENTITY サービスの認証情報が無効

Identity サービスへの認証で問題がある場合には、**ironic.conf** ファイルの **identity_uri** パラメーターをチェックして、**keystone** AdminURL から **/v2.0** が削除されていることを確認してください。たとえば、**identity_uri** を **http://IP:PORT** に設定します。

9.6. ハードウェアの登録

ノードの登録情報が間違っていると、登録したハードウェアで問題が発生する可能性があります。属性名と値を正しく入力してください。属性名を誤って入力すると、システムは属性をノードの詳細に追加しますが、無視されます。

openstack baremetal node set コマンドを使用して、ノードの情報を更新します。以下の例では、ノードの登録されたメモリー使用量を 2 GB に更新します。

```
$ openstack baremetal node set --property memory_mb=2048 NODE_UUID
```

9.7. IDRAC に関する問題のトラブルシューティング

Redfish 管理インターフェースがブートデバイスの設定に失敗する

特定の iDRAC ファームウェアバージョンの **idrac-redfish** 管理インターフェースを使用する場合、UEFI ブートのベアメタルサーバーでブートデバイスの設定を試みると、iDRAC は以下のエラーを返します。

```
Unable to Process the request because the value entered for the
parameter Continuous is not supported by the implementation.
```

この問題が発生した場合には、ノードの **driver-info** の **force_persistent_boot_device** パラメーターを **Never** に設定します。

```
openstack baremetal node set --driver-info force_persistent_boot_device=Never ${node_uuid}
```

電源オフ時のタイムアウト

一部のサーバーで、電源のオフに長時間を要し、タイムアウトする場合があります。デフォルトのリトライ回数は **6** で、その結果 30 秒でタイムアウトになります。タイムアウトの時間を 90 秒に増やすには、アンダークラウドの hieradata オーバーライドファイルで

ironic::agent::rpc_response_timeout の値を **18** に設定して、**openstack undercloud install** コマンドを再実行します。

```
ironic::agent::rpc_response_timeout: 18
```

ベンダーパススルーのタイムアウト

ベンダーパススルーコマンドを実行するのに iDRAC が利用できない場合、これらのコマンドの実行に非常に長い時間がかかり、タイムアウトします。

```
openstack baremetal node passthru call --http-method GET \
aed58dca-1b25-409a-a32f-3a817d59e1e0 list_unfinished_jobs
Timed out waiting for a reply to message ID 547ce7995342418c99ef1ea4a0054572 (HTTP 500)
```

メッセージングのタイムアウト時間を増やすには、アンダークラウドの hieradata オーバーライドファイルで **ironic::default::rpc_response_timeout** パラメーターの値を増やし、**openstack undercloud install** コマンドを再実行します。

```
ironic::default::rpc_response_timeout: 600
```

9.8. サーバーコンソールの設定

オーバークラウドノードからのコンソール出力は、常にサーバーコンソールに送信される訳ではありません。サーバーコンソールでこの出力を表示するには、ハードウェアの正しいコンソールを使用するようにオーバークラウドを設定する必要があります。この設定を行うには、以下のいずれかの方法を使用します。

- オーバークラウドロールごとに **KernelArgs** heat パラメータを変更する
- オーバークラウドノードをプロビジョニングするのに director が使用する **overcloud-full.qcow2** イメージをカスタマイズする

前提条件

- アンダークラウドの正常なインストール。詳しくは、『Director Installation and Usage』を参照してください。
- デプロイ可能なオーバークラウドノード

デプロイメント時の heat を使用した KernelArgs の変更

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. source コマンドで **stackrc** 認証情報ファイルを読み込みます。

```
$ source stackrc
```

3. 環境ファイル **overcloud-console.yaml** を作成して、以下の内容を記載します。

```
parameter_defaults:
  <role>Parameters:
    KernelArgs: "console=<console-name>"
```

<role> を設定するオーバークラウドロールの名前に置き換え、**<console-name>** を使用するコンソールの ID に置き換えます。たとえば、デフォルトロールのすべてのオーバークラウドノードが **tty0** を使用するように設定するには、以下のスニペットを使用します。

```
parameter_defaults:
  ControllerParameters:
    KernelArgs: "console=tty0"
  ComputeParameters:
    KernelArgs: "console=tty0"
  BlockStorageParameters:
    KernelArgs: "console=tty0"
  ObjectStorageParameters:
    KernelArgs: "console=tty0"
  CephStorageParameters:
    KernelArgs: "console=tty0"
```

4. **-e** オプションを使用して、**overcloud-console-tty0.yaml** ファイルをデプロイメントコマンドに追加します。

overcloud-full.qcow2 イメージの変更

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. source コマンドで **stackrc** 認証情報ファイルを読み込みます。

```
$ source stackrc
```

3. **overcloud-full.qcow2** イメージのカーネル引数を変更して、ハードウェアの正しいコンソールを設定します。たとえば、コンソールを **tty0** に設定します。

```
$ virt-customize --selinux-relabel -a overcloud-full.qcow2 --run-command 'grubby --update-kernel=ALL --args="console=tty0"'
```

4. イメージを director にインポートします。

```
$ openstack overcloud image upload --image-path overcloud-full.qcow2
```

5. オーバークラウドをデプロイします。

検証

1. アンダークラウドからオーバークラウドノードにログインします。

```
$ ssh heat-admin@<IP-address>
```

<IP-address> をオーバークラウドノードの IP アドレスに置き換えます。

2. **/proc/cmdline** ファイルの内容を調べ、**console=** パラメーターが使用するコンソールの値に設定されていることを確認します。

```
[heat-admin@controller-0 ~]$ cat /proc/cmdline
BOOT_IMAGE=(hd0,msdos2)/boot/vmlinuz-4.18.0-193.29.1.el8_2.x86_64
root=UUID=0ec3dea5-f293-4729-b676-5d38a611ce81 ro console=tty0
console=ttyS0,115200n81 no_timer_check crashkernel=auto rhgb quiet
```

-

第10章 BARE METAL のドライバー

ベアメタルノードが Bare Metal Provisioning サービスで有効にしたドライバーの 1 つを使用するように設定することができます。各ドライバーには、プロビジョニング方法と電源管理の種別が含まれます。ドライバーによっては追加の設定が必要な場合があります。このセクションに記述された各ドライバーはプロビジョニングに PXE を使用します。ドライバーは電源管理タイプ別にリストされます。

ironic.yaml ファイルに **IronicEnabledHardwareTypes** パラメーターを設定して、ドライバーを追加することができます。デフォルトでは、ipmi および **redfish** は有効になっています。

サポートされているプラグインとドライバーの全一覧は、「Component, Plug-In, and Driver Support in Red Hat OpenStack Platform」のアーティクルを参照してください。

10.1. INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)

IPMI は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェースです。この電源管理種別を使用するには、全 Bare Metal Provisioning サービスノードで IPMI が共有ベアメタルネットワークに接続されている必要があります。**ipmi** ドライバーを有効にし、ノードの **driver_info** に以下の情報を設定します。

- **ipmi_address**: IPMI NIC の IP アドレス
- **ipmi_username**: IPMI のユーザー名
- **ipmi_password**: IPMI のパスワード

10.2. REDFISH

Distributed Management Task Force (DMTF) の開発した、IT インフラストラクチャー向け標準 RESTful API。

- **redfish_username**: Redfish のユーザー名
- **redfish_password**: Redfish のパスワード
- **redfish_address**: Redfish コントローラーの IP アドレス
- **redfish_system_id**: システムリソースへの正規のパス。このパスには、そのシステムの root サービス、バージョン、パス/一意 ID を含める必要があります (例: **/redfish/v1/Systems/CX34R87**)。
- **redfish_verify_ca**: ブール値、または CA_BUNDLE ファイルもしくは信頼済み CA の証明書が含まれるディレクトリーへのパス。この値を **True** に設定すると、ドライバーはホストの証明書を検証します。この値を **False** に設定すると、ドライバーは SSL 証明書の検証を無視します。この値をパスに設定すると、ドライバーは指定された証明書またはディレクトリー内の証明書の 1 つを使用します。デフォルトは **True** です。

10.3. DELL REMOTE ACCESS CONTROLLER (DRAC)

DRAC は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェースです。この電源管理種別を使用するには、全 Bare Metal Provisioning サービスノードで DRAC が共有ベアメタルネットワークに接続されている必要があります。**idrac** ドライバーを有効にし、ノードの **driver_info** に以下の情報を設定します。

- **drac_address**: DRAC NIC の IP アドレス

- **drac_username**: DRAC のユーザー名
- **drac_password**: DRAC のパスワード

10.4. INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)

富士通の iRMC は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェースです。Bare Metal Provisioning サービスノードでこの電源管理種別を使用するには、このノードで iRMC インターフェースが共有ベアメタルネットワークに接続されている必要があります。**irmc** ドライバーを有効にし、ノードの **driver_info** に以下の情報を設定します。

- **irmc_address**: iRMC インターフェースの NIC の IP アドレス
- **irmc_username**: iRMC のユーザー名
- **irmc_password**: iRMC のパスワード

IPMI を使用してブートモードを設定する場合、または SCCI を使用してセンサーデータを取得する場合には、追加で以下のステップを完了する必要があります。

1. **ironic.conf** でセンサーメソッドを有効にします。

```
$ openstack-config --set /etc/ironic/ironic.conf \
  irmc sensor_method METHOD
```

   METHOD は **scci** または **ipmitool** に置き換えます。

2. SCCI を有効にした場合は、**python-scciclient** パッケージをインストールします。

```
# dnf install python-scciclient
```

3. Bare Metal Conductor サービスを再起動します。

```
# systemctl restart openstack-ironic-conductor.service
```

注記

iRMC ドライバーを使用するには、iRMC S4 以降が必要です。

10.5. INTEGRATED LIGHTS-OUT (ILO)

Hewlett-Packard の iLO は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェースです。この電源管理タイプを使用するには、全ベアメタルノードで iLO インターフェースが共有ベアメタルネットワークに接続されている必要があります。**ilo** ドライバーを有効にし、ノードの **driver_info** に以下の情報を設定します。

- **ilo_address**: iLO インターフェースの NIC の IP アドレス
- **ilo_username**: iLO のユーザー名
- **ilo_password**: iLO のパスワード

python-proliantutils パッケージもインストールして、Bare Metal Conductor サービスを再起動する必要があります。

```
# dnf install python-proliantutils
# systemctl restart openstack-ironic-conductor.service
```

10.6. 次世代電源管理ドライバーへの移行

Red Hat OpenStack Platform では **ハードウェアタイプ** とも呼ばれる次世代ドライバーが使用され、従来のドライバーがこれに置き換えられています。

従来のドライバーとそれと等価な次世代ハードウェアタイプの対比を、以下の表に示します。

従来のドライバー	新しいハードウェアタイプ
pxe_ipmitool	**ipmi**
pxe_drac	**idrac**
pxe_ilo	**ilo**
pxe_irmc	**irmc**
fake_pxe	**fake-hardware**

Red Hat OpenStack Platform (RHOSP) 15 では、これらの従来ドライバーは廃止され、使用できなくなっています。RHOSP 15 にアップグレードする前に、新しいハードウェアタイプに変更する必要があります。

手順

1. 有効なハードウェアタイプの最新の一覧を確認します。

```
$ source ~/overcloud
$ openstack baremetal driver list --type dynamic
```

2. 有効ではないハードウェアタイプのドライバーを使用する場合には、環境ファイルの **IronicEnabledHardwareTypes** パラメーターを使用してそのドライバーを有効にします。

```
parameter_defaults:
  IronicEnabledHardwareTypes: ipmi,redfish,idrac
```

3. ファイルを保存し、オーバークラウドのデプロイコマンドを実行します。

```
$ openstack overcloud deploy -e [ENVIRONMENT_FILE] -r [ROLES_DATA] -n
[NETWORK_DATA]
```

ご自分のオーバークラウドに関連する環境ファイルおよびデータファイルをすべて追加するようにしてください。

4. 以下のコマンドを実行します。**OLDDRIVER** と **NEWDRIVER** 変数を実際の電源管理タイプに置き換えます。

```
$ source ~/overcloud
$ OLDDRIVER="pxe_ipmitool"
$ NEWDRIVER="ipmi"
$ for NODE in $(openstack baremetal node list --driver $OLDDRIVER -c UUID -f value) ; do
openstack baremetal node set $NODE --driver $NEWDRIVER; done
```