



# Red Hat OpenStack Platform 13

## 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



# Red Hat OpenStack Platform 13 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

OpenStack Team  
rhos-docs@redhat.com

## 法律上の通知

Copyright © 2021 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

本ガイドには、Red Hat OpenStack Platform 環境のオーバークラウドに Bare Metal サービスをインストール、設定、および使用するための手順を記載しています。

## 目次

前書き .....	4
第1章 BARE METAL サービスについて .....	5
第2章 ベアメタルプロビジョニングのプランニング .....	7
2.1. インストールの前提条件	7
2.2. ハードウェア要件	7
2.3. ネットワーク要件	7
2.3.1. デフォルトのベアメタルネットワーク	8
2.3.2. カスタムコンポーザブルネットワーク	9
第3章 BARE METAL サービスを有効にしたオーバークラウドのデプロイ .....	11
3.1. IRONIC のテンプレートの作成	11
3.2. ネットワーク設定	11
3.2.1. カスタムプロビジョニングネットワークの設定	12
3.3. テンプレートの例	13
3.4. オーバークラウドのデプロイ	14
3.5. BARE METAL サービスのテスト	14
第4章 デプロイ後の BARE METAL サービスの設定 .....	16
4.1. OPENSTACK NETWORKING の設定	16
4.1.1. OpenStack Networking がフラットなベアメタルネットワーク上の Bare Metal サービスと通信するための設定	16
4.1.2. 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 サービスと通信するための設定	17
4.2. ノードのクリーニングの設定	18
4.2.1. 手動によるノードのクリーニング	19
4.3. ベアメタルフレーバーの作成	19
4.4. ベアメタルイメージの作成	20
4.4.1. デプロイイメージの準備	20
4.4.2. ユーザーイメージの準備	21
4.4.2.1. ディスクイメージの環境変数	21
4.4.3. ユーザーイメージのインストール	22
4.5. ベアメタルノードとしての物理マシンの追加	23
4.5.1. 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	23
4.5.2. ベアメタルノードの手動登録	25
4.6. 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	28
第5章 ベアメタルノードの管理 .....	30
5.1. 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	30
5.2. DASHBOARD を使用したインスタンスの起動	30
5.3. BARE METAL PROVISIONING サービスでのポートグループの設定	31
5.3.1. スイッチの設定	31
5.3.2. Bare Metal Provisioning サービスでのポートグループの設定	32
5.4. ホストから IP アドレスへのマッピングの確認	32
5.5. 仮想ネットワークインターフェースの接続と切断	35
5.6. BARE METAL サービスの通知の設定	37
第6章 インスタンスとしてのベアメタルノードの使用 .....	38
6.1. 前提条件	38
6.2. イメージの生成	38
6.3. クラスターの作成	38

<b>第7章 ML2 NETWORKING-ANSIBLE</b> .....	<b>40</b>
7.1. MODULAR LAYER 2 (ML2) NETWORKING-ANSIBLE	40
7.2. NETWORKING-ANSIBLE のネットワーク要件	40
7.3. NETWORKING-ANSIBLE 用の OPENSTACK BARE METAL (IRONIC) の要件	40
7.4. NETWORKING-ANSIBLE ML2 機能の有効化	41
7.5. NETWORKING-ANSIBLE 用ネットワーク設定	43
7.6. ベアメタルゲスト用のポート設定	43
7.7. NETWORKING-ANSIBLE ML2 機能のテスト	44
<b>第8章 BARE METAL サービスのトラブルシューティング</b> .....	<b>45</b>
8.1. PXE ブートエラー	45
8.2. ベアメタルノードの起動後のログインエラー	46
8.3. デプロイされたノードでの BOOT-TO-DISK エラー	47
8.4. BARE METAL サービスが正しいホスト名を取得しない	47
8.5. BARE METAL 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	47
8.6. ハードウェアの登録	48
8.7. NO VALID HOST エラー	48
<b>付録A BARE METAL のドライバー</b> .....	<b>49</b>
A.1. INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)	49
A.2. DELL REMOTE ACCESS CONTROLLER (DRAC)	49
A.3. INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)	49
A.4. INTEGRATED LIGHTS-OUT (ILO)	50
A.5. SSH と VIRSH	50
A.6. VIRTUALBMC	51
A.6.1. pxe_ssh から VirtualBMC への移行	51



## 前書き

本ガイドには、オーバークラウドに Bare Metal サービス (ironic) をインストールして設定し、そのサービスを使用して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を行うための手順を記載しています。

Bare Metal サービスのコンポーネントは、Red Hat OpenStack Platform director で OpenStack 環境 (オーバークラウド) を構成するベアメタルノードのプロビジョニングと管理を行うためにアンダークラウドの一部としても使用されます。director による Bare Metal サービスの使用方法については、『[director のインストールと使用方法](#)』を参照してください。

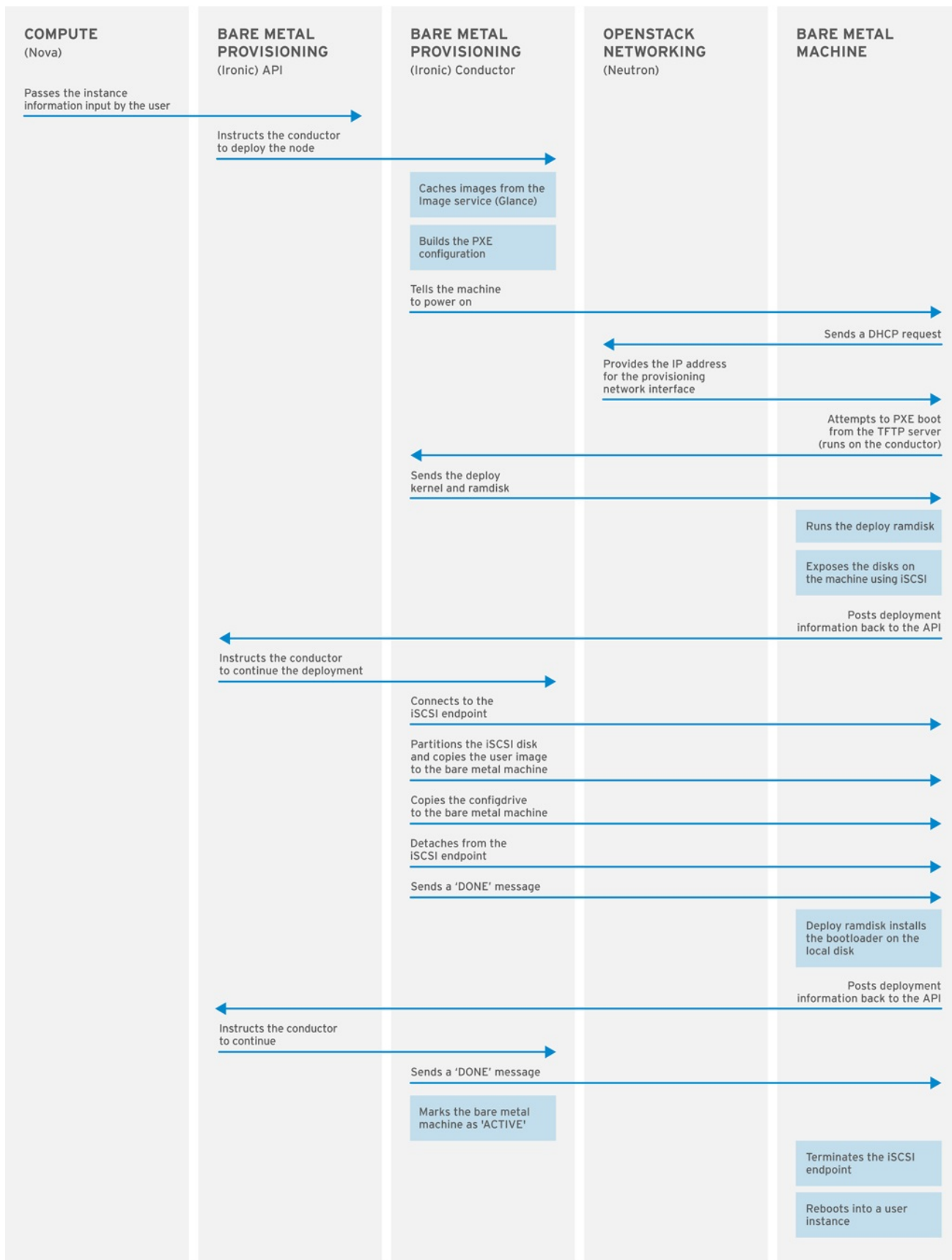


## 第1章 BARE METAL サービスについて

OpenStack Bare Metal 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必要なコンポーネントを提供します。オーバークラウドの Bare Metal 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 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 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、特定のドライバーがイメージの一時的な URL を公開するために使用されます。

Bare Metal 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、ユーザーが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プランニング

本章では、インストールの前提条件、ハードウェア要件、ネットワーク要件など、Bare Metal サービスを設定するための要件について説明します。

### 2.1. インストールの前提条件

本ガイドでは、アンダークラウドに director をインストール済みで、Bare Metal サービスと残りのオーバークラウドをインストールする準備が整っていることを前提とします。director のインストール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「[director のインストール](#)」を参照してください。



#### 注記

ベアメタルノードは、OpenStack 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 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。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場合、ユーザーはコントロールプレーンにアクセスする必要はありません。

### 2.2. ハードウェア要件

#### オーバークラウドの要件

Bare Metal 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 のインストールと使用方法』の「[オーバークラウドの要件](#)」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 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はすべて、以下の要件を満たす必要があります。

- 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** を別の場所で行っている場合以外は、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



#### 注記

OVN を使用する場合、Bare Metal Provisioning サービス (ironic) はオーバークラウドではサポートされません。現在、OVN の組み込み型 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。

## ベアメタルネットワーク:

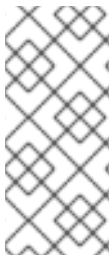
これは、Bare Metal サービスが以下の用途で使用するプライベートネットワークです。

- 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 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 Ironic サービスを実装する。



### 注記

ベアメタルノードは、OpenStack 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 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。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場合、ユーザーはコントロールプレーンにアクセスする必要はありません。

## 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す。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 サービスを有効にした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:

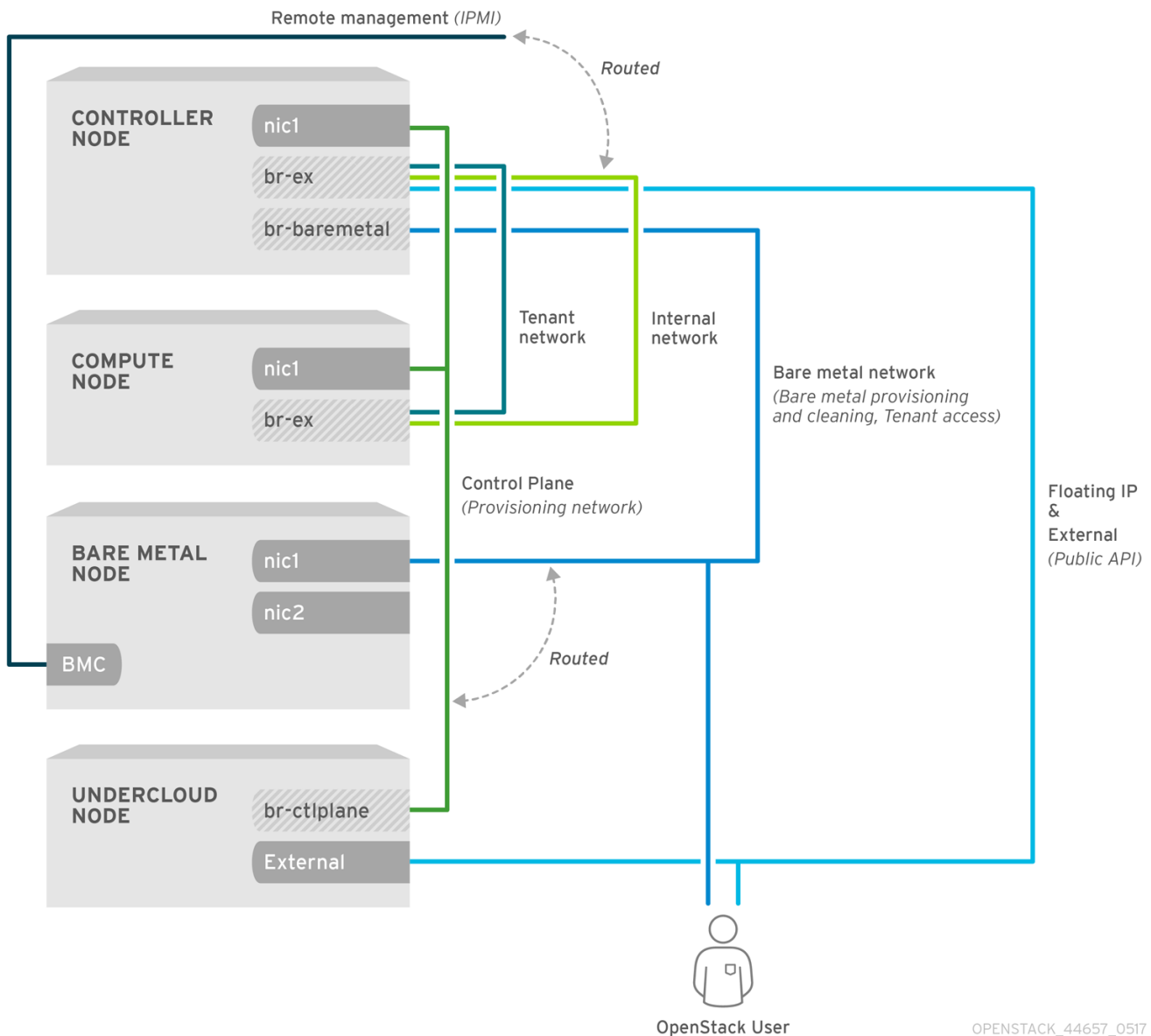
ベアメタルノードの PXE ブートに使用するよう設定されている NIC は、ベアメタルネットワークにアクセス可能である必要があります。

### 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。
- Ironic 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。
- Ironic 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したい場合には、このネットワークの作成が推奨されます。

- 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 サービスを有効にしたオーバークラウドのデプロイ

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 のインストールと使用方法](#)』を参照してください。本章では、ironic 固有のデプロイメント手順のみを説明します。

### 3.1. IRONIC のテンプレートの作成

環境ファイルを使用して、Bare Metal 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。テンプレートは、director ノードの `/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml` にあります。

#### テンプレートへの記入

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。サンプルは、『[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 `puppet/services/ironic-conductor.yaml` によって「full」に設定されます。この設定を「metadata」にすると、パーティションテーブルのみがクリーニングされるので処理速度を大幅に向上させることができますが、複数のテナントがある環境ではデプロイメントのセキュリティレベルが低くなるため、信頼済みのテナント環境でのみ適用すべきです。
- IronicEnabledDrivers** 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、`pxe_ipmitool`、`pxe_drac`、および `pxe_ilo` が有効です。

設定パラメーターの全一覧は、『[オーバークラウドのパラメーター](#)』の『[Bare Metal \(ironic\) パラメーター](#)』を参照してください。

### 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、ironic が使用するブリッジ `br-baremetal` を作成する必要があります。これは、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
```

このブリッジを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) 内に設定して、このネットワークをベアメタルネットワークとして再利用できるようにするか、専用のネットワークを追加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、OpenStack 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 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りますので、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。

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. **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、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、VLAN を追加し 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
```

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. 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を干渉しないことを確認してください。

**~/templates/ironic.yaml**

```
parameter_defaults:

  NovaSchedulerDefaultFilters:
    - RetryFilter
    - AggregateInstanceExtraSpecsFilter
    - AvailabilityZoneFilter
    - RamFilter
    - DiskFilter
    - ComputeFilter
    - ComputeCapabilitiesFilter
```

```
- ImagePropertiesFilter
```

```
IronicCleaningDiskErase: metadata
```

この例では、

- **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。

### 3.4. オーバークラウドのデプロイ

Bare Metal サービスを有効にするには、オーバークラウドの初回のデプロイメントまたは再デプロイメントの時に **-e** を使用して `ironic` の環境ファイルをオーバークラウドの残りの設定と共に追加します。

以下に例を示します。

```
$ 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-docker/ironic.yaml \
-e ~/templates/ironic.yaml \
```

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「[CLI ツールを使用したオーバークラウドの作成](#)」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。

### 3.5. BARE METAL 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 サービスを検証するその他の方法

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 デプロイ後の BARE METAL サービスの設定

本項では、デプロイ後のオーバークラウドの設定に必要な手順について説明します。

### 4.1. OPENSTACK NETWORKING の設定

DHCP、PXE ブート、およびその他の必要な場合に OpenStack Networking が Bare Metal 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。

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

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

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。ノードのクリーニングを実行する際に誤って再構成を行わないようにするには、この値を **provisioning** に設定します。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** に **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入る必要があります。
  - **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ある必要があります。
  - **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. ネットワークを新しいルーターに接続します。

```
$ openstack router add network ROUTER_NAME NETWORK
```

**ROUTER\_NAME** をルーターの名前に、**NETWORK** を以前のステップで作成したネットワークの ID または名前に、それぞれ置き換えます。

6. 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。

#### 4.1.2. 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 サービスと通信するための設定

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、コントローラーにログオンし、以下のコマンドを実行して名前を変更し、ネットワークを再起動します。

```
heat-admin@overcloud-controller-0 ~]$ sudo vi /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf
```

```
heat-admin@overcloud-controller-0 ~]$ sudo docker restart ironic_conductor
```

## 4.2. ノードのクリーニングの設定

デフォルトでは、Bare Metal 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するように設定されます。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 サービスとの競合が発生する可能性があります。このため、ネットワーク名の代わりにネットワークの UUID を使用することを推奨します。

1. Bare Metal サービスを実行しているコントローラー上のプロバイダーネットワークの UUID を指定して、クリーニングを設定します。

```
~/templates/ironic.yaml
```

```
parameter_defaults:
  IronicCleaningNetwork: UUID
```

**UUID** は、以前のステップで作成され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** 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. 「[オーバークラウドのデプロイ](#)」の説明に従って **openstack overcloud deploy** コマンドを実行し、オーバークラウドを再デプロイして変更を適用します。 **openstack overcloud deploy** コマンドでオーバークラウドを再デプロイすると、手動で加えていた変更はすべて元に戻るのので、 **openstack overcloud deploy** を次回使用する前には、(前のステップで説明した) クリーニングの設定を `~/templates/ironic.yaml` に必ず追加してください。
3. `director` を使用せずに、手動で変更を適用することも可能です。クリーニングネットワークを設定します。

```
crudini --set /var/lib/config-data/puppet-generated/ironic/etc/ironic/ironic.conf neutron
cleaning_network
```

4. Bare Metal Provisioning サービスを再起動します。

```
# docker restart ironic_conductor
```

### 4.2.1. 手動によるノードのクリーニング

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** の状態である必要があります。

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、より高速なクリーンサイクルですが、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。

**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 に置き換えてください。

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確認してください。

## 4.3. ベアメタルフレーバーの作成

デプロイメントの一部として使用するフレーバーを作成する必要があります。このフレーバーの仕様(メモリー、CPU、ディスク)はベアメタルノードが提供する仕様以下である必要があります。

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. 既存のフレーバーを一覧表示します。

```
$ openstack flavor list
```

3. Bare Metal サービス向けに新規フレーバーを作成します。

```
$ openstack flavor create \
  --id auto --ram RAM \
  --vcpus VCPU --disk DISK \
  --property baremetal=true \
  --public baremetal
```

**RAM** はメモリー量、**VCPU** は仮想 CPU 数、**DISK** はディスクストレージの値に置き換えま  
す。**baremetal** プロパティーは、ベアメタルを仮想インスタンスと区別するために使用されま  
す。

4. 指定したそれぞれの値を使用して新規フレーバーが作成されたことを確認します。

```
$ openstack flavor list
```

## 4.4. ベアメタルイメージの作成

デプロイメントには 2 セットのイメージが必要です。

- **デプロイイメージ** は、Bare Metal サービスが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するのに使用されます。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** は、ベアメタルノードにデプロイされるイメージです。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 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。

### 4.4.1. 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使用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、kernel 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。

イメージを抽出して Image サービスにアップロードします。

```
$ mkdir images  
$ tar -xf /usr/share/rhosp-director-images/ironic-python-agent-latest.tar -C images/  
$ cd images  
$ 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

```

#### 4.4.2. 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され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。

##### 4.4.2.1. 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。これらは、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、システムがイメージのビルドプロセスを完了すると登録を解除します。

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

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

登録プロセスの一部で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

パッケージをプルするための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、このパラメーターにお使いの Satellite サーバーのベース URL を使用する必要があります。

#### 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

イメージに登録する組織

#### 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指定します。

## REG\_REPOS

コンマ区切りのリポジトリ名の文字列 (空白なし)。この文字列の各リポジトリは **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指定します。Red Hat カスタマーポータルの場合のデフォルトは **subscription.rhn.redhat.com** です。Red Hat Satellite 6 サーバーを使用する場合は、このパラメーターにお使いの Satellite サーバーのホスト名を使用する必要があります。

## REG\_USER

イメージを登録するアカウントのユーザー名を指定します。

### 4.4.3. ユーザーイメージのインストール

1. [カスタマーポータル](#) (ログインが必要) 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-server-7.4-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がある場合には、`DIB_YUM_REPO_CONF` をローカルリポジトリ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。

```
$ disk-image-create rhel7 baremetal -o rhel-image
```

これで kernel は **rhel-image.vmlinuz** として、初期 ramdisk は **rhel-image.initrd** として抽出されます。

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
```

## 4.5. 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 サービスにインポートしてからノードを利用できるようにします。
2. 物理マシン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 ファイルが配置されている任意のノードで実行できます。

本項では、両方の方法について詳しく説明します。

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は同期していないためです。次の定期タスクが実行されると、変更が認識されるようになります。この値 **scheduler\_driver\_task\_period** は、`/etc/nova/nova.conf` で更新することができます。デフォルトの間隔は 60 秒です。

### 4.5.1. 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

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: pxe_ipmitool
    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てください。

- **<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。
- **<PXE\_NIC\_MAC>** 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、含める必要があるのは **root\_device** のみです。**<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
```

4. これで、ノードは **enroll** の状態となります。
5. 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
```

6. 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 サービスがノードをクリーニングします。

7. 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"
```

8. 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。

#### 4.5.2. ベアメタルノードの手動登録

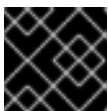
1. Identity を管理ユーザーとして使用するためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. 新しいノードを追加します。

```
$ openstack baremetal node create --driver pxe_ipmitool --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 `pxe_ipmitool` 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化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、「[付録A 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 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、`ironic driver-properties` コマンドで返された必要なプロパティに置き換えます。
  - **VALUE** は、プロパティ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された `.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. ノードのプロパティを更新して、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. オプション: 初回のデプロイの後には、`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するイメージに `grub2` が含まれている必要があります。ローカルブートを以下のように設定します。

```
$ 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。

7. 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 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。

8. 複数のディスクがある場合には、ルートデバイスのヒントを設定してください。これにより、デプロイメントに使用すべき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"}
```

以下の値に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): ディスクを用いるデバイス (HDD) の場合は true、それ以外 (SSD) の場合は false。
  - **name**: デバイス名 (例: /dev/sdb1)。これは、永続デバイス名が付いたデバイスのみを使用してください。



#### 注記

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

9. 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。上記の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るはずです。**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ので、この結果は問題ありません。

## 4.6. 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ゾーンをパーティション分割し、特定の共有プロパティ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をホストアグリゲートに割り当てられたプロパティと比較して、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。

以下の手順は、次の作業の方法を説明します。

- **baremetal** プロパティ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、このプロパティを継承します。

### ホストアグリゲートの作成

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

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



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

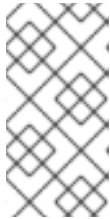
```
$ openstack flavor set FLAVOR_NAME --property baremetal=false
```

- 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
```

- 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

**Novalronic** 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、**Novalronic** サービスがあるのはコントローラーノードのみです。

- 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
```

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

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

- 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5章 ベアメタルノードの管理

本章では、登録済みのベアメタルノードで物理マシンをプロビジョニングする方法について説明します。インスタンスは、コマンドラインまたは OpenStack Dashboard で起動することができます。

### 5.1. 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。

#### コマンドライン上でのインスタンスのデプロイ

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. 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 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、『[ユーザーおよびアイデンティティー管理ガイド](#)』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
```

### 5.2. 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。

#### Dashboard でのインスタンスのデプロイ

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。選択したイメージは 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 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 の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。

## 5.3. 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。

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** を使用する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。

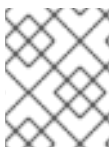


### 注記

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。

### 5.3.1. スイッチの設定

Bare Metal Provisioning デプロイメントでポートグループを設定するには、スイッチ上で手動設定する必要があります。スイッチ上のモードとプロパティは、スイッチによって名前が異なる場合があるため、それらがベアメタル側のモードとプロパティに対応していることを確認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。

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

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、スタンドアロンポートへのフォールバックをサポートしてい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** ポートが接続されていない場合には、ポートグループは設定されません。

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。

## 5.4. ホストから IP アドレスへのマッピングの確認

以下のコマンドを実行すると、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できます。

この機能により、ホストに直接アクセスする必要なく、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。

```
(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 |
+-----+-----+-----+-----+
```

特定のホストをフィルターするには、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
```

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"
}
]
```

## 5.5. 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 (たとえば、OpenStack Networking サービスで使用される仮想ネットワークインターフェース) と物理ネットワークインターフェース (NIC) との間のマッピングを管理するための API があります。これらのインターフェースは各 Bare Metal Provisioning ノードに対して設定可能で、**openstack baremetal node vif\*** コマンドを使用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。

以下の例で、VIF を接続および切断する手順を説明します。

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 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。  
これは、以下のコマンドで確認できます。

```
$ 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 | control          | control    |
ctlplane=192.168.24.7 | overcloud-full |
| 3a1bc89c-5d0d-44c7-a569-f2a3b4c73d65 | overcloud-controller-0 | ACTIVE | control          | control    |
ctlplane=192.168.24.8 | overcloud-full |
| 6b01531a-f55d-40e9-b3a2-6d02be0b915b | overcloud-controller-1 | ACTIVE | control          | control    |
ctlplane=192.168.24.16 | overcloud-full |
| c61cc52b-cc48-4903-a971-073c60f53091 | overcloud-novacompute-0overcloud-baremetal-0 | ACTIVE | compute         | compute    |
ctlplane=192.168.24.24 | overcloud-full |
+-----+-----+-----+-----+-----+
-----+-----+
```

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 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. ベアメタルノードを再起動して、新規 IP アドレスが認識されるようにします。

```
$ 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最も簡単な方法は、ベアメタルホストをリブートすることです。

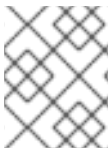
## 5.6. BARE METAL サービスの通知の設定

Bare Metal サービスを設定して、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。このような通知は、課金目的やデータストアの監視などで外部のサービスが使用することができます。本項では、この通知を有効にする方法について説明します。

Bare Metal サービスの通知を有効化するには、**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。

- **[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。



### 注記

通知は失われる可能性があり、通知がメッセージバスを通過してエンドユーザーに届く保証はありません。

## 第6章 インスタンスとしてのベアメタルノードの使用

このユースケースでは、ベアメタルノードを下層のハードウェアとして使用するインスタンスをデプロイすることができます。Sahara は、ビッグデータクラスター作成時に2つの内部タスクを実行します。

1. Heat は、インスタンスの作成 (必須ネットワークを含む) に使用されます。
2. インスタンスの準備が整ったら (**openstack server list** が **ACTIVE** の状態)、sahara は各ノードに接続して、指定されたビッグデータプラグインの設定を適用します。これには、ビッグデータインスタンスの準備が整うまでの追加のソフトウェアのインストール、サービスの起動、およびその他のタスクが含まれます。

### 6.1. 前提条件

- オーバークラウド上で Bare Metal Provisioning (ironic) と Data Processing (sahara) をデプロイする際には、デフォルトのパラメーターを使用します。
- ベアメタルノードはすべて、事前定義済みのフレーバー (以下 **baremetal\_flavor** と呼ぶ) の下でグループ化される必要があります。
- 仮想ノードとベアメタルノードを組み合わせた混合設定はテスト済みではないため、サポートされない場合があります。

一般的には、仮想インスタンスは通常プライベートプロジェクトネットワークに接続されてから、パブリックネットワーク上の Floating IP プールを介してアクセス可能となります)。ただし、ironic によって管理されるベアメタルマシンが単一のネットワークでのみアクセス可能な場合には、問題が発生する場合があります。そのため、sahara のクラスターは Floating IP アドレスプールを使用せずに、そのネットワークのみを使用するように設定すべきです。この問題は、ベアメタルノードに限られず、sahara が仮想マシンのみで使用されている場合にも発生する可能性があります。

### 6.2. イメージの生成

**sahara-image-elements** (追加の **baremetal** スイッチを含む) を使用してベアメタルノード用のイメージを生成する必要がある可能性があります。この作業を行う場合は、カーネルと initrd イメージも生成する必要があります。ただし、通常 **sahara-image-elements** によって生成されるイメージは完全なディスクイメージとして機能するため、ベアメタルイメージを生成する必要が全くない場合もあります。フレーバーには一時ディスクが必要で、それによりさらにパーティションイメージが必要となるため、ベアメタルイメージは、**MapR** プラグインに必要な場合があります。現在、生成スクリプトには、ベアメタルイメージの作成を妨げる既知の問題があります。これは、今後の更新で解決される見込みです。

イメージが生成されたら、それらを glance にアップロードして、sahara で登録します。

### 6.3. クラスターの作成

CDH プラグインを使用したテストシナリオの例を以下に示します。

1. CDH ノードグループテンプレートおよびクラスターテンプレートの標準的なセットを作成します。ただし、このユースケースでは、新規 **baremetal\_flavor** を指定する必要があり、Floating IP アドレスプールは必要ない可能性があります。たとえば、以下のように割り当てます。
  - 1x manager
  - 1x master-core

- 1x master-additional
  - 1x worker-nm-dn
2. **dfs\_replication** を **1** に設定して有効化します。
  3. フレーバーを **baremetal\_flavor** に設定します。
  4. クラスターを作成します。作成されるクラスターは、正常に初期化され、デプロイされるインスタンスはベアメタルノードを使用するはずです。

## 第7章 ML2 NETWORKING-ANSIBLE

本項では、OpenStack Networking (neutron) の **networking-ansible** ML2 ドライバー、OpenStack Bare Metal (ironic) との統合、およびオーバークラウドでこのドライバーの有効化および設定を行う手順について説明します。

### 7.1. 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、オーバークラウドのデプロイメントプロセスがトリガーとなり、**networking-ansible** ドライバーが、VLAN neutron ネットワークを使用するベアメタルゲストおよびそれに対応する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。

- OpenStack 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
- OpenStack 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
- OpenStack 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

### 7.2. NETWORKING-ANSIBLE のネットワーク要件

以下の一覧で、**networking-ansible** 機能を有効にするためのネットワーク要件の概要を説明します。

- Ansible Network Automation 対応のネットワークスイッチ。現在、Red Hat では Juniper Networks (**Junos**) スイッチの使用のみをサポートしています。
- 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 ユーザーも必要です。このユーザーには、スイッチでの特定の権限が必要です。
  - 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。

- VLAN を手動で管理する場合、スイッチが使用する VLAN を準備します。そのためには、それぞれのスイッチポートの VLAN を作成します。この要件は、**networking-ansible** 機能を有効にする際の **manage\_vlans** オプションに基づいています。
- 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。

### 7.3. 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 ironic サービスをデプロイします。

- 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その他の設定例については、「[3章 Bare Metal サービスを有効にした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
```

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

## 7.4. NETWORKING-ANSIBLE ML2 機能の有効化

この手順では、オーバークラウドで **networking-ansible** ML2 ドライバーを有効にする方法を説明します。この設定では、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`

このファイルには、スイッチの詳細情報が含まれます。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. 環境内の各スイッチ用に、**ML2HostConfigs** パラメーターに **dict** 定義を追加します。それぞれの **dict** キーはスイッチの名前で、OpenStack Networking (neutron) ML2 設定の特定の **ansible:[switchname]** セクションを定義します。各スイッチの **dict** 値には、スイッチの詳細情報が含まれます。たとえば、スイッチを3つ設定するには、**ML2HostConfigs** パラメーターにスイッチキーを3つ追加します。

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

```
[SWITCH DETAILS]
switch3:
[SWITCH DETAILS]
```

- それぞれのスイッチの **dict** に、以下のキー/値のペアを含めます。

#### **ansible\_network\_os**

(必須) スイッチのオペレーティングシステム。現在、Red Hat では **Junos** のみをサポートしています。

#### **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。

#### **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。

- 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: junos
      ansible_host: 10.0.0.1
      ansible_user: ansible
      ansible_ssh_pass: "p@55w0rd!"
      mac: 01:23:45:67:89:AB
      manage_vlans: false
```

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

```
$ 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** コンテナの OpenStack Networking (neutron) API の一部としてドライバーを有効にします。

## 7.5. 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については、「[4章 デプロイ後の Bare Metal 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。

## 7.6. ベアメタルゲスト用のポート設定

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。そのためには、OpenStack Networking (neutron) ポート設定を定義します。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

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=[SWICH PORT ID] \
--local-link-connection switch_info=[SWITCH NAME] \
--local-link-connection switch_id=[SWITCH MAC]
```

以下の大カッコ内の値は、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した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。

## 7.7. 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、動作を確かめるために機能をテストします。これを行うには、ベアメタルに関する負荷を作成する必要があります。

### 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witch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
  1. 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
- 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

### 手順

1. ベアメタルシステムを作成します。

```
openstack server create --flavor baremetal --image overcloud-full --key default --network
tenant-net 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8章 BARE METAL サービスのトラブルシューティング

以下の項には、Bare Metal サービスを有効にした環境における問題を診断するのに役立つ可能性のある情報と手順を記載します。

### 8.1. PXE ブートエラー

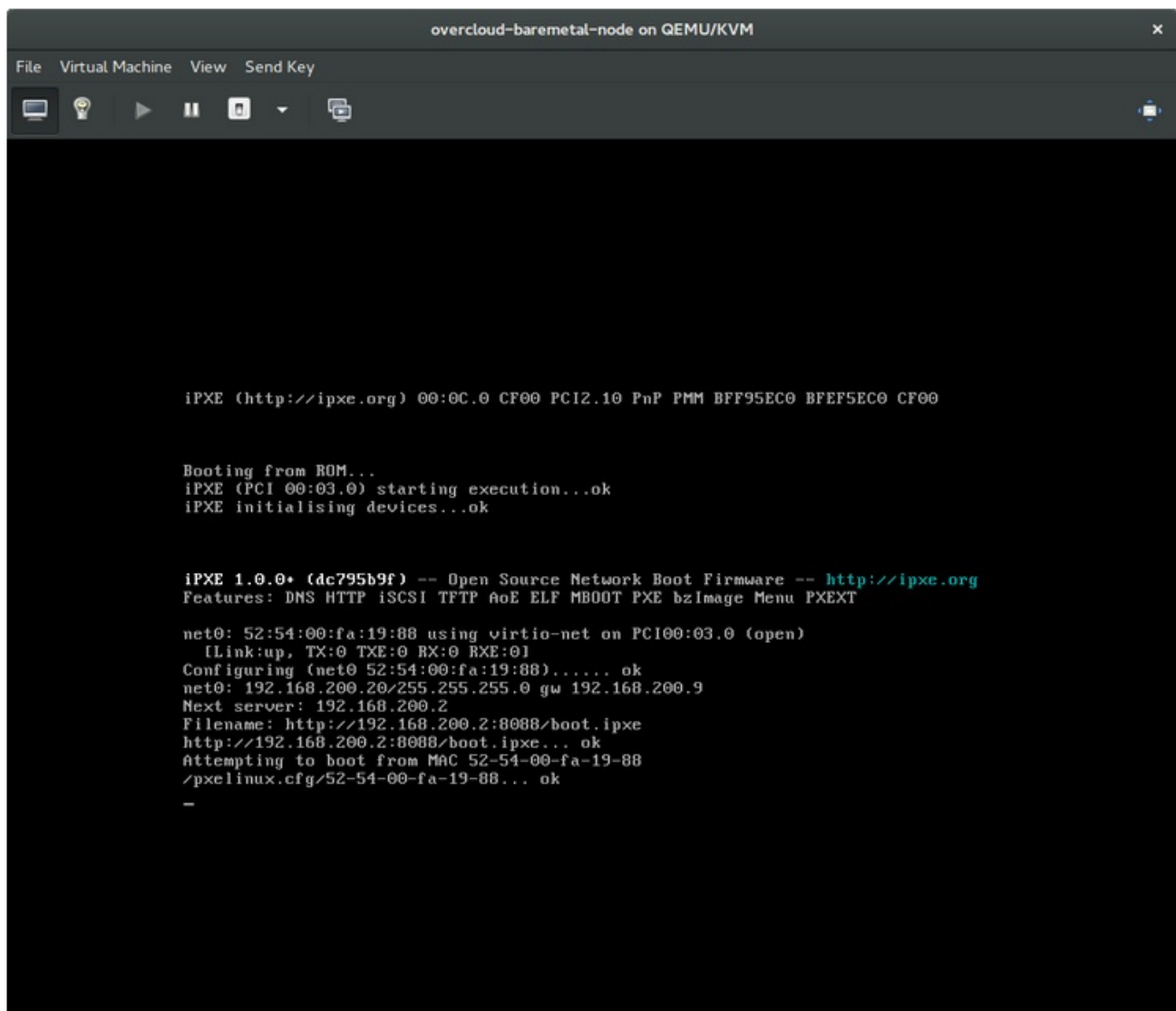
#### Permission Denied エラー

Bare Metal サービス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、以下に示したように、適切な SELinux コンテキストを `/httpboot` と `/tftpboot` のディレクトリーに必ず適用してください。

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

以下の図に示したように、ノードのコンソールで、IP アドレスは取得され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:



```
overcloud-baremetal-node on QEMU/KVM
File Virtual Machine View Send Key

IPXE (http://ipxe.org) 00:0C:0 CF00 PCI2.10 PnP PMM BFF95EC0 BFEF5EC0 CF00

Booting from ROM...
IPXE (PCI 00:03.0) starting execution...ok
IPXE initialising devices...ok

IPXE 1.0.0• (dc795b9f) -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS HTTP iSCSI TFTP AoE ELF MBOOT PXE bzImage Menu PXEXT

net0: 52:54:00:fa:19:88 using virtio-net on PCI00:03.0 (open)
[Link:up, TX:0 TXE:0 RX:0 RXE:0]
Configuring (net0 52:54:00:fa:19:88)..... ok
net0: 192.168.200.20/255.255.255.0 gw 192.168.200.9
Next server: 192.168.200.2
Filename: http://192.168.200.2:8088/boot.ipxe
http://192.168.200.2:8088/boot.ipxe... ok
Attempting to boot from MAC 52-54-00-fa-19-88
/pxelinux.cfg/52-54-00-fa-19-88... ok
-
```

これは、**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ることを示しています。

```
$ 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 **ipxe\_config.template** です。

## 8.2. ベアメタルノードの起動後のログインエラー

ノードのコンソールのログインプロンプトで、設定手順中に設定した **root** パスワードを使用してログインを試みてもログインできない場合には、デプロイしたイメージでブートしていないことを意味します。**deploy-kernel/deploy-ramdisk** イメージにスタックしてしまって、システムが正しいイメージをまだ取得して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 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 サービスノードが認識してい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 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=pxe_ipmitool 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	<b>/etc/ironic/ironic.conf</b> ファイルのパラメーター <b>http_url</b> 。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	<b>ironic node-list</b> 内のベアメタルノードの UUID
deploy_kernel	これは、 <b>/httpboot/&lt;NODE_UUID&gt;/deploy_kernel</b> 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。

上記の例の kernel 行の値	対応する情報
http://192.168.200.2:6385	<code>/etc/ironic/ironic.conf</code> ファイルのパラメーター <code>api_url</code> 。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。
pxe_impitool	このノードの Bare Metal サービスが使用している IPMI ドライバー
deploy_ramdisk	これは、 <code>/httpboot/&lt;NODE_UUID&gt;/deploy_ramdisk</code> 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 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。

### 8.3. 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、Legacy BIOS 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。

### 8.4. BARE METAL サービスが正しいホスト名を取得しない

Bare Metal サービスが正しいホスト名を取得しない場合は、`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。meta-data エージェントへの要求はこれで正しくルーティングされるようになるはずです。

### 8.5. BARE METAL 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` は `http://IP:PORT` に設定する必要があります。

## 8.6. ハードウェアの登録

ハードウェア登録での問題は、ノードの登録情報が誤っていることが原因となっている可能性があります。プロパティ名と値が正しく入力されていることを確認してください。プロパティ名に誤りやタイプミスがあってもノードの情報には正常に追加されますが、そのプロパティ名は無視されます。

ノードの情報を更新します。以下の例では、登録するノードのメモリ使用量を 2 GB に更新します。

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

## 8.7. NO VALID HOST エラー

Compute スケジューラーがインスタンスを起動するのに適切なベアメタルノードを見つけられない場合、**NoValidHost** エラーが `/var/log/nova/nova-conductor.log` に表示されるか、起動に失敗した直後に Dashboard に表示されます。通常これは、Compute が想定するリソースとベアメタルノードが提供するリソースが一致しないことが原因です。

1. 利用可能なハイパーバイザーのリソースを確認します。

```
$ openstack hypervisor stats show
```

このコマンドで返されるリソースは、Bare Metal が提供するリソースと一致する必要があります。

2. Compute がベアメタルノードをハイパーバイザーとして認識していることを確認します。

```
$ openstack hypervisor list
```

ノードは UUID で識別され、一覧に表示されるはずですが。

3. ベアメタルノードの詳細を確認します。

```
$ openstack baremetal node list
$ openstack baremetal node show NODE_UUID
```

ノードの詳細が、Compute によって返された情報と一致することを確認します。

4. 選択したフレーバーがベアメタルノードで利用可能なリソースを超えていないことを確認します。

```
$ openstack flavor show FLAVOR_NAME
```

5. `openstack baremetal node list` の出力をチェックして、ベアメタルノードがメンテナンスモードに入っていないことを確認します。必要な場合には、メンテナンスモードを解除してください。

```
$ openstack baremetal node maintenance unset NODE_UUID
```

6. `openstack baremetal node list` の出力をチェックして、ベアメタルノードが **available** の状態であることを確認します。必要な場合には、ノードを **available** に切り替えます。

```
$ openstack baremetal node provide NODE_UUID
```

## 付録A BARE METAL のドライバー

ベアメタルノードは、Bare Metal 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。各ドライバーは、プロビジョニングメソッドと電源管理のタイプで構成されます。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** ファイルの **IronicEnabledDrivers** 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、**pxe\_ipmitool**、**pxe\_drac**、および **pxe\_ilo** が有効です。

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」](#) のアートを参照してください。

### A.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。この電源管理タイプを使用するには、全 Bare Metal 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。**pxe\_ipmitool** 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

### A.2. 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。この電源管理タイプを使用するには、全 Bare Metal 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。**pxe\_drac** 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

### A.3. 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 サービスノードでこの電源管理タイプを使用するには、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。**pxe\_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。

```
# yum 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。

## A.4. 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。`pxe_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。

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

## A.5. SSH と VIRSH



#### 注記

`pxe_ssh` ドライバーは非推奨となっており、`pxe_ipmitool` ドライバーを使用する VirtualBMC が推奨されています。

Bare Metal サービスは、`libvirt` を実行するホストにアクセスして、仮想マシンをノードとして使用することができます。`virsh` はノードの電源管理機能を制御します。



#### 重要

SSH ドライバーは、テストおよび評価の目的でのみ利用いただけます。Red Hat OpenStack Platform のエンタープライズ環境には推奨していません。

この電源管理タイプを使用するには、Bare Metal サービスは仮想ノードを設定するホスト上の libvirt 環境に完全にアクセス可能なアカウントに SSH でアクセスする必要があります。 **pxe\_ssh** 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。

- **ssh\_virt\_type**: このオプションは **virsh** に設定します。
- **ssh\_address**: virsh ホストの IP アドレス
- **ssh\_username**: SSH ユーザー名
- **ssh\_key\_contents**: Bare Metal コンダクターノード上の SSH 秘密鍵の内容。対応する公開鍵が virsh ホストにコピーされている必要があります。

## A.6. VIRTUALBMC

VirtualBMC とは、仮想 Baseboard Management Controller (BMC) を作成することのできるユーティリティです。仮想 BMC により、Intelligent Platform Management Interface (IPMI) プロトコルを使用して仮想マシンを物理マシンのように制御できます。



### 重要

VirtualBMC は、テストおよび評価の目的でのみ利用いただけます。Red Hat OpenStack Platform のエンタープライズ環境には推奨していません。

1. 仮想マシンをホストしているハイパーバイザーで VirtualBMC をインストールします。

```
# yum install python-virtualbmc
```

2. 各仮想マシン用に BMC を作成します。

```
$ vbmc add DOMAIN --port PORT_NUMBER --username USERNAME --password PASSWORD
```

3. 仮想 BMC を起動します。

```
$ vbmc start DOMAIN
```

仮想 BMC は自動的に起動しないので、ホストマシンをリブートした場合には、仮想 BMC を再起動することを忘れないようにしてください。

### A.6.1. pxe\_ssh から VirtualBMC への移行

**pxe\_ssh** ドライバーを使用する既存の仮想オーバークラウドがある場合には、仮想 BMC を使用して **pxe\_impitool** に移行することができます。

1. 上記のステップに従って、**python-virtualbmc** をインストールし、各仮想マシン用の BMC を作成します。
2. **pxe\_impitool** が有効化されていることを確認してください (デフォルトでは有効化されます)。有効でない場合には、**ironic.yaml** ファイルに追加し、**openstack overcloud deploy** コマンドを再度実行して、変更を適用します。

```
parameter_defaults:
```

```
IronicEnabledDrivers:
```

- pxe\_ipmitool
- pxe\_drac
- pxe\_ilo

3. 各ノードで、ドライバーとドライバーのプロパティを更新します。

```
$ openstack baremetal node set NODE \  
  --driver pxe_ipmitool \  
  --driver-info ipmi_address=IP_ADDRESS \  
  --driver-info ipmi_port=PORT \  
  --driver-info ipmi_username="USERNAME" \  
  --driver-info ipmi_password="PASSWORD"
```

- **NODE** は、ノードの名前または UUID に置き換えます。
  - **IP\_ADDRESS** は仮想ホストの IP アドレスに置き換えます。
  - **PORT** は仮想 BMC ポートに置き換えます。
4. ノードが正しく更新されたことを確認します。

```
$ openstack baremetal node validate NODE | grep power
```

- **NODE** は、ノードの名前または UUID に置き換えます。