



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、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、使用方法

OpenStack Team

rhos-docs@redhat.com

法律上の通知

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本ガイドには、Red Hat OpenStack Platform 環境のオーバークラウドで Bare Metal サービスをインストール、設定、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
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	10
3.1. IRONIC のテンプレートの作成	10
3.2. ネットワーク設定	10
3.3. テンプレートの例	11
3.4. オーバークラウドのデプロイ	11
3.5. BARE METAL サービスのテスト	12
第4章 デプロイ後の BARE METAL サービスの設定	13
4.1. OPENSTACK NETWORKING の設定	13
4.2. ノードのクリーニングの設定	14
4.2.1. 手動によるノードのクリーニング	15
4.3. ベアメタル用のフレーバーの作成	16
4.4. ベアメタルイメージの作成	16
4.4.1. デプロイイメージの準備	17
4.4.2. ユーザーイメージの準備	17
4.5. ベアメタルノードとしての物理マシンの追加	18
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	18
4.5.2. ベアメタルノードの手動登録	20
4.6. ホストアグリゲートを使用した物理/仮想マシンのプロビジョニングの分離	23
第5章 ベアメタルノードの管理	25
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	25
5.2. DASHBOARD を使用したインスタンスの起動	25
5.3. BARE METAL PROVISIONING サービスでのポートグループの設定	26
5.3.1. スイッチの設定	26
5.3.2. Bare Metal Provisioning サービスでのポートグループの設定	27
5.4. ホストから IP アドレスへのマッピングの確認	28
5.5. 仮想ネットワークインターフェースのアタッチとデタッチ	31
5.6. BARE METAL サービスの通知の設定	33
第6章 インスタンスとしてのベアメタルノードの使用	34
6.1. 前提条件	34
6.2. イメージの生成	34
6.3. クラスターの作成	34
第7章 BARE METAL サービスのトラブルシューティング	36
7.1. PXE ブートエラー	36
7.2. ベアメタルノードの起動後のログインエラー	37
7.3. BARE METAL サービスが正しいホスト名を取得しない	38
7.4. 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	38
7.5. ハードウェアの登録	38
7.6. NO VALID HOST エラー	38
付録A BARE METAL のドライバー	40

A.1. INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)	40
A.2. DELL REMOTE ACCESS CONTROLLER (DRAC)	40
A.3. INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)	40
A.4. INTEGRATED LIGHTS-OUT (ILO)	41
A.5. SSH と VIRSH	41
A.6. VIRTUALBMC	42
A.6.1. pxe_ssh から VirtualBMC への移行	42

前書き

本ガイドでは、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手順を記載します。

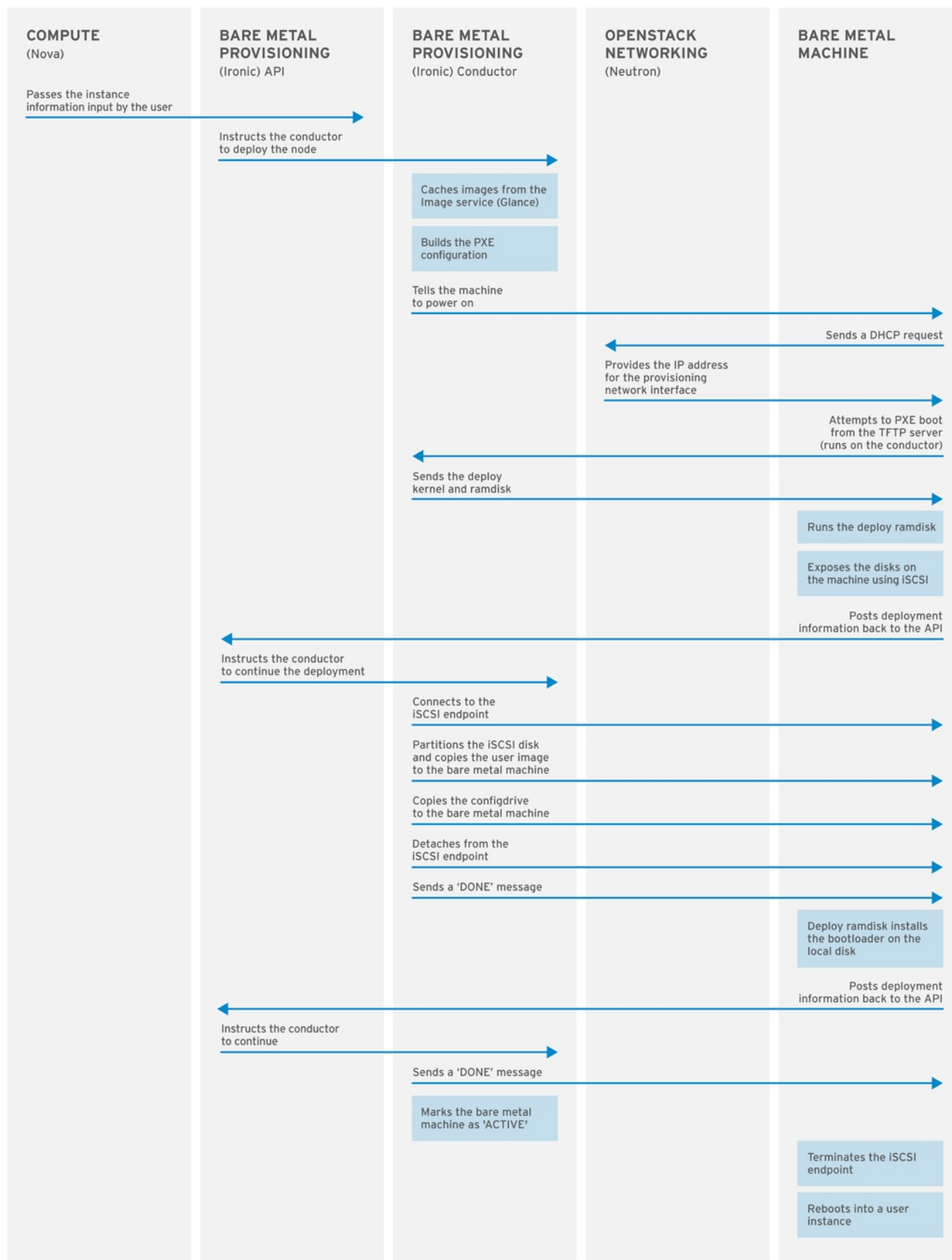
Bare Metal サービスのコンポーネントは、**Red Hat OpenStack Platform director** で **OpenStack 環境** (オーバークラウド) を構成するベアメタルノードのプロビジョニングと管理を行うためにアンダークラウドの一部としても使用されます。**director** による **Bare Metal サービス**の使用方法については、『[director のインストールと使用方法](#)』を参照してください。

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。オーバークラウド内の 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 を公開するのに使用します。

Bare Metal サービスは、PXE 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に、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。



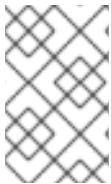
OPENSTACK_377593_1215

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** のインストールに関する詳しい情報は、「[アンダークラウドのインストール](#)」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アクセスが可能であるため、オーバークラウド内の 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。

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』ガイドの「[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)。SSH ドライバーをテスト目的で使用している場合には、これは必要ありません。コンポーザブルロールを使用して **ironic-conductor** を別の場所で実行している場合以外は、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 では 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:

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がコントロールプレーンネットワークに到達していること:

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 ブートできなくなります。



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、オーバークラウド内の **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。

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:

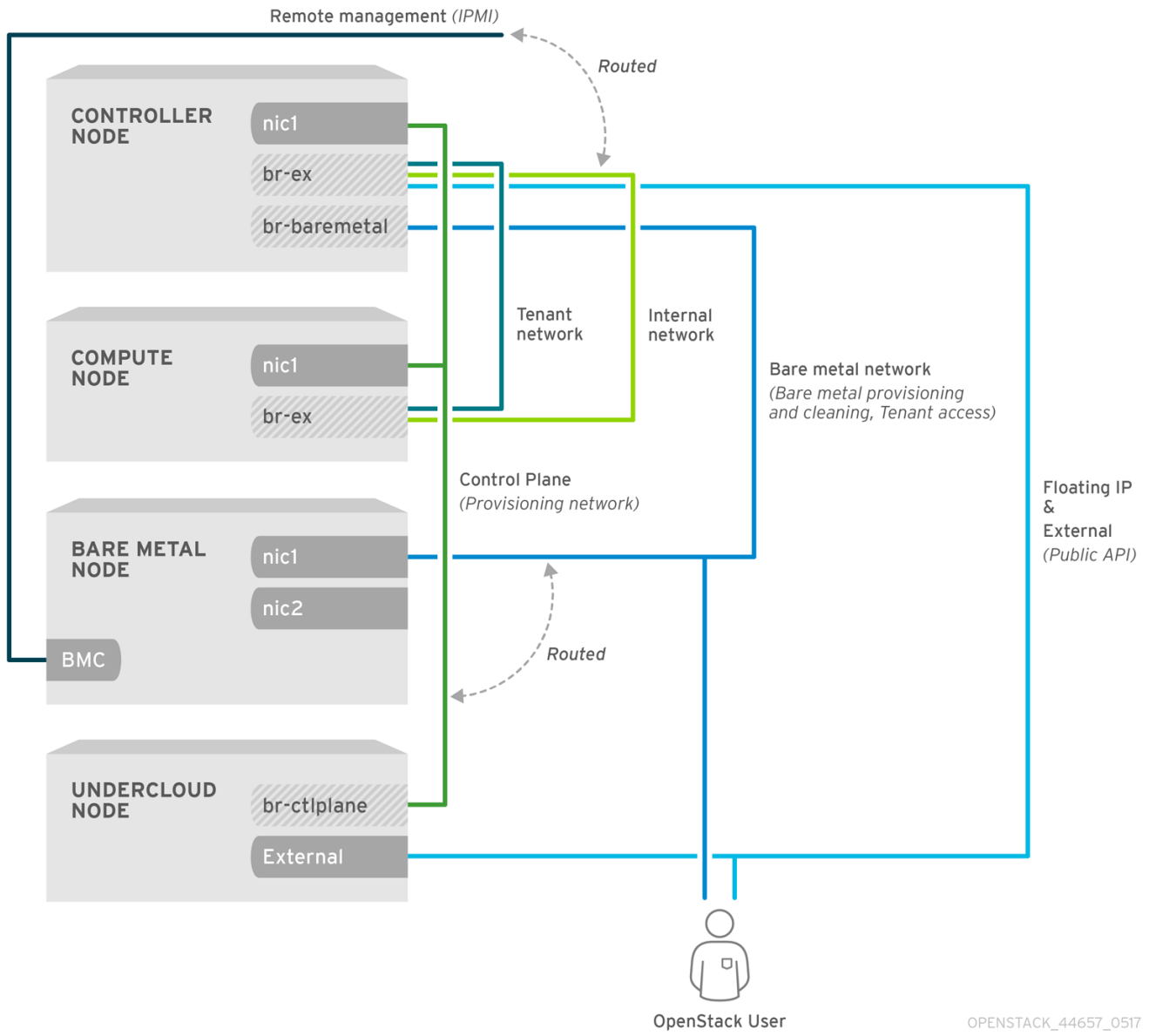
ベアメタルノードの PXE ブートに使用するよう設定されている **NIC** は、ベアメタルネットワークにアクセス可能である必要があります。

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します。

- 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



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/ironic.yaml` にあります。

テンプレートへの情報記入

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`、`pxe_ilo` が有効化されています。

設定パラメーターの全一覧は、『[Overcloud Parameters](#)』の『[Bare Metal \(ironic\) Parameters](#)』の項を参照してください。

3.2. ネットワーク設定

ironic が使用する **br-baremetal** という名前のブリッジを作成します。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
```

3.3. テンプレートの例

テンプレート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。

例:

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

詳しい情報は、「[CLI ツールを使用したオーバークラウドの作成](#)」([『director のインストールと使用方法』](#))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。詳しくは、[『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追加の方法

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。以下の手順では、ベアメタルマシンのを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のユースケース向けに OpenStack Networking 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。**flat** ネットワークのみがサポートされています。

以下の手順では、ベアメタルネットワークのインターフェースを使用してブリッジを作成し、リモートの接続をすべて切断します。

以下の手順に記載するステップはすべて、OpenStack Networking 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する必要があります。

OpenStack Networking が 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。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 アドレスの割り当て元とな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 アドレスの割り当て元とな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をルーターに接続して、メタデータ要求が OpenStack Networking サービスによって応答されるようにします。

```
$ openstack router create ROUTER_NAME
```

ROUTER_NAME はルーターの名前に置き換えます。

5. ベアメタルのサブネットをこのルーターに追加します。

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

ROUTER_NAME は、ルーターの名前に、 **BAREMETAL_SUBNET** は以前に作成した ID またはサブネット名に置き換えます。これにより、 **cloud-init** からのメタデータ要求に応答することができます。

4.2. ノードのクリーニングの設定

デフォルトでは、ベアメタルサービスは、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。



注記

各コントローラーで **ironic.conf** を更新することによって、オーバークラウドの再デプロイを回避することが可能です。ただし、**OpenStack director** のインストール環境における **ironic.conf** ファイルの手動更新はサポートされていません。以下に記載する手順は便宜のためにのみ提供しています。

1. 以下の行をコメント解除して、**<None>** をベアメタルネットワークの UUID に置き換えます。

```
cleaning_network = <None>
```

2. Bare Metal サービスを再起動します。

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

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ため、**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。

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 のセキュアな削除を使用するか、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。処理が完了するには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 workflow execution create manual_cleaning \
  '{"node_uuid": "UUID", "clean_steps": [{"step": \
    "erase_devices_metadata", "interface": "deploy"}]}'
```

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 workflow execution create manual_cleaning \
  '{"node_uuid": "UUID", "clean_steps": [{"step": \
    "erase_devices", "interface": "deploy"}]}'
```

manual_cleaning はワークフロー名です。**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. ベアメタル用のフレーバーの作成

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、**kernel** イメージと **ramdisk** 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 **kernel** イメージと **ramdisk** イメージが含まれますが、追加で **main** 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ディスク全体のイメージのいずれかです。

- **ディスク全体のイメージ**は、パーティションテーブルとブートローダーが含まれたイメージです。ディスク全体の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をダウンロードします。

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

アンダークラウドでオーバークラウドをデプロイする際には、デプロイイメージはすでに使用されているので、作成する必要はありません。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。

```
ironic-python-agent.kernel
ironic-python-agent.initramfs
```

これらのイメージは、削除してしまったり、別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。

```
$ openstack image create \
  --container-format aki \
  --disk-format aki \
  --public \
  --file ./ironic-python-agent.kernel bm-deploy-kernel
$ openstack image create \
  --container-format ari \
  --disk-format ari \
  --public \
  --file ./ironic-python-agent.initramfs 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 **kernel** と **ramdisk** に加えて、**main** イメージが含まれます。

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. **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。

4. 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。これらのステップは、**overcloudrc** ファイルが配置されている任意のノードで実行することができます。

本項では、両メソッドについて詳しく説明します。

物理マシンを登録した後は、新規リソースは **Compute** に直ちに通知されます。これは、**Compute** のリソーストラッカーが定期的に同期しているためです。次の定期タスクが実行されると変更が表示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 **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>** はベアメタル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 のアーキテクチャタイプに置き換えます。
- **<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、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. ノードはこれで **enroll** の状態となります。各ノードで **deploy kernel** と **deploy ramdisk** を指定して、利用可能な状態にします。

```

$ 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** 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

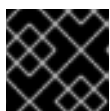
1. Identity を管理ユーザーとして使用するためのシェルを設定します。

```
$ source ~/overcloudrc
```

2. 新規ノードを追加します。

```
$ openstack baremetal node create --driver pxe_impitool --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_impitool** を使用しています。異なるドライバーを使用するには、**IroniEnabledDrivers** 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 RAM ディスクを指定します。

```
$ 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** はディスクのサイズ (GB 単位) に置き換えます。
 - **ARCH** はアーキテクチャタイプに置き換えます。
6. オプション: ノードを設定して、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、その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 **deploy 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}'**)。以下のプロパティーがサポートされています。
 - **model** (文字列): デバイスの識別子
 - **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 向けのホスト:チャンネル:ターゲット:Lun
 - **size** (整数): デバイスのサイズ (GB)
 - **wwn** (文字列): ストレージの一意識別子
 - **wwn_with_extension** (文字列): ベンダー拡張が末尾に付いたストレージの一意識別子
 - **wwn_vendor_extension** (文字列): ベンダーのストレージの一意識別子
 - **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**: 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。

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」、「image_source」のパラメーターが指定されていないことが原因となって、インターフェースでエラーが発生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/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、特定の共通プロパティー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。

以下の手順は、次の作業の方法を説明します。

- **baremetal** プロパティーをフレーバーに追加して、**true** または **false** に設定します。

- ベアメタルホスト用とコンピュートノード用にホストアグリゲートを別々に作成し、**baremetal** プロパティーが一致するようにします。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。

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には **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た場合には、このサービスの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** 下に追加してください。

```
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ている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関する詳しい情報は、『[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
```

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](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 をクリックします。

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** を選択して、**+**(プラス)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、**+**(プラス)および **-**(マイナス) 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、**Bare Metal** サービス用に作成した共有ネットワークを必ず選択してください。
- インスタンスをセキュリティグループに割り当てるには、**セキュリティグループ** 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) の機能は、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方法を提供します。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 を使用するデプロイメントを起動する必要がある場合には、プロビジョニングとクリーニングにはポートグループは使用できません。

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、最初に使われます。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、イメージ内で手動でスタンドアロンモードに設定するか、**configdrive** を生成してノードの **instance_info** に追加して設定する必要があります。ポートグループの設定が機能するには、**cloud-init** バージョンが **0.7.7** 以降であることを確認してください。

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** で開始し、設定済みポートグループごとに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.storage.mgmt.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.storage.mgmt.localdomain overcloud-
controller-0.storage.mgmt
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アタッチとデタッチ

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 サービスで使わ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) の間のマッピングに使用される 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。

以下の例では、VIF を接続/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 | 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 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には、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再起動することです。

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異なるイベントの通知が表示されるようにすることが可能です。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、この通知を有効化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**、**critical** のいずれかの全通知が送信されますが、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、でふお
- 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ます、**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 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。

7.1. 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ように表示されており、プロセスが停止している場合:

```
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 **pxe_config.template** なので、これを **ipxe_config.template** に変更するための **i** を忘れがちです。

7.2. ベアメタルノードの起動後のログインエラー

設定ステップで設定した **root** パスワードを使用して、ノードのコンソールのログインプロンプトで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、ファイルのカーネル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=VWDYDVVEFCQJNOST09R67HKUXUGP77CK
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	/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

上記の例の kernel 行の 値	対応する情報
deploy_kernel	これは、 <code>/httpboot/<NODE_UUID>/deploy_kernel</code> としてコピーされた Image サービス内の deploy kernel イメージです。
<code>http://192.168.200.2:6385</code>	<code>/etc/ironic/ironic.conf</code> ファイル内のパラメーター <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_ipmitool	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/<NODE_UUID>/deploy_ramdisk</code> としてコピーされた Image サービス内の deploy 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. Bare Metal インスタンスを再デプロイします。

7.3. 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。

7.4. 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。

7.5. 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場合があります。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。プロパティ名が誤ったり、タイプエラーがある場合でも、ノードの情報には正常に追加されますが、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ー使用量を 2 GB に更新します。

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

7.6. NO VALID HOST エラー

Compute の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には、**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。これは、通常 **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。

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 サービスで有効化されたドライバーの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 **IroniEnabledDrivers** 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**、**pxe_ilo** が有効化されています。

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](#)」の記事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のモニタリングを含む帯域外 (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** ドライバーを有効化し、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のモニタリングを含む帯域外 (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** ドライバーを有効化し、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のモニタリングを含む帯域外 (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、このノードに、共有ベアメタルネットワークに接続された iRMC インターフェースが1つ必要です。**pxe_irmc** ドライバーを有効化し、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か、SCCI を使用してセンサーデータを取得するには、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 **sccci** 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 コンダクター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)

iLO は、電源管理やサーバーのモニタリングを含む帯域外 (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、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た iLO インターフェースが1つ必要です。**pxe_ilo** ドライバーを有効化し、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 コンダクター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
```

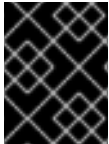
A.5. SSH と VIRSH



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

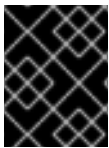
この SSH ドライバーは、検証および評価のみを目的としており、Red Hat OpenStack Platform のエンタープライズ環境には推奨されません。

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て、ノードの **driver_info** で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、[IPMI protocol](#) 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されません。

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。

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

3. 仮想 BMC を起動します。

```
$ vmc start DOMAIN
```

仮想 BMC は自動的に起動しないので、ホストを再起動した場合には、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_ipmitool** 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せん)。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。