



# Red Hat OpenStack Platform 13

## director のインストールと使用方法

Red Hat OpenStack Platform director を使用した OpenStack クラウド作成のエンド  
ツーエンドシナリオ



# Red Hat OpenStack Platform 13 director のインストールと使用方法

---

Red Hat OpenStack Platform director を使用した OpenStack クラウド作成のエンドツーエンドシナリオ

OpenStack Team  
rhos-docs@redhat.com

## 法律上の通知

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 概要

本ガイドでは、エンタープライズ環境で Red Hat OpenStack Platform director を使用して Red Hat OpenStack Platform 13 をインストールする方法について説明します。これには、director のインストール、環境のプランニング、director を使用した OpenStack 環境の構築などが含まれます。

## 目次

第1章 はじめに .....	6
1.1. アンダークラウド	6
1.2. オーバークラウド	7
1.3. 高可用性	9
1.4. CEPH STORAGE	9
第2章 要件 .....	11
2.1. 環境要件	11
2.2. アンダークラウドの要件	11
2.2.1. 仮想化サポート	12
2.3. ネットワーク要件	14
2.4. オーバークラウドの要件	17
2.4.1. Compute ノードの要件	17
2.4.2. コントローラーノードの要件	18
2.4.3. Ceph Storage ノードの要件	18
2.4.4. Object Storage ノードの要件	19
2.5. リポジトリの要件	20
第3章 オーバークラウドのプランニング .....	23
3.1. ノードのデプロイメントロールのプランニング	23
3.2. ネットワークのプランニング	24
3.3. ストレージのプランニング	30
第4章 アンダークラウドのインストール .....	32
4.1. STACK ユーザーの作成	32
4.2. アンダークラウドのホスト名の設定	32
4.3. アンダークラウドの登録と更新	33
4.4. DIRECTOR パッケージのインストール	34
4.5. DIRECTOR の設定	34
4.6. DIRECTOR の設定パラメーター	35
4.7. DIRECTOR のインストール	40
4.8. オーバークラウドノードのイメージの取得	40
4.9. コントロールプレーン用のネームサーバーの設定	42
4.10. 次のステップ	43
第5章 コンテナイメージのソースの設定 .....	44
5.1. レジストリーメソッド	44
5.2. CONTAINER IMAGE PREPARE コマンドの使用方法	44
5.3. 追加のサービス用のコンテナイメージ	46
5.4. RED HAT レジストリーをリモートレジストリーソースとして使用する方法	48
5.5. ローカルレジストリーとしてアンダークラウドを使用する方法	48
5.6. SATELLITE サーバーをレジストリーとして使用する手順	50
5.7. 次のステップ	53
第6章 CLI ツールを使用した基本的なオーバークラウドの設定 .....	54
6.1. オーバークラウドへのノードの登録	55
6.2. ノードのハードウェアの検査	56
6.3. ベアメタルノードの自動検出	63
6.4. プロファイルへのノードのタグ付け	66
6.5. ノードのルートディスクの定義	67
6.6. 環境ファイルを使用したオーバークラウドのカスタマイズ	69
6.7. CLI ツールを使用したオーバークラウドの作成	70
6.8. オーバークラウド作成時の環境ファイルの追加	75

6.9. オーバークラウドプランの管理	78
6.10. オーバークラウドのテンプレートおよびプランの検証	79
6.11. オーバークラウド作成の監視	79
6.12. オーバークラウドへのアクセス	80
6.13. オーバークラウド作成の完了	80
<b>第7章 WEB UI を使用した基本的なオーバークラウドの設定</b> .....	<b>81</b>
7.1. WEB UI へのアクセス	81
7.2. WEB UI のナビゲーション	83
7.3. WEB UI を使用したオーバークラウドプランのインポート	86
7.4. WEB UI を使用したノードの登録	87
7.5. WEB UI を使用したノードのハードウェアのイントロスペクション	89
7.6. WEB UI を使用したプロファイルへのノードのタグ付け	90
7.7. WEB UI を使用したオーバークラウドプランのパラメーターの編集	91
7.8. WEB UI でのロールの追加	93
7.9. WEB UI を使用したロールへのノードの割り当て	93
7.10. WEB UI を使用したロールパラメーターの編集	94
7.11. WEB UI を使用したオーバークラウドの作成開始	95
7.12. オーバークラウド作成の完了	97
<b>第8章 事前にプロビジョニングされたノードを使用した基本的なオーバークラウドの設定</b> .....	<b>98</b>
8.1. ノード設定のためのユーザーの作成	99
8.2. ノードのオペレーティングシステムの登録	99
8.3. ノードへのユーザーエージェントのインストール	100
8.4. DIRECTOR への SSL/TLS アクセスの設定	101
8.5. コントロールプレーンのネットワークの設定	101
8.6. オーバークラウドノードに別のネットワークを使用する方法	103
8.7. 事前にプロビジョニングされたノードでのオーバークラウドの作成	105
8.8. メタデータサーバーのポーリング	106
8.9. オーバークラウド作成の監視	109
8.10. オーバークラウドへのアクセス	109
8.11. 事前にプロビジョニングされたノードのスケーリング	109
8.12. 事前にプロビジョニングされたオーバークラウドの削除	110
8.13. オーバークラウド作成の完了	110
<b>第9章 オーバークラウド作成後のタスクの実行</b> .....	<b>111</b>
9.1. コンテナ化されたサービスの管理	111
9.2. オーバークラウドのテナントネットワークの作成	112
9.3. オーバークラウドの外部ネットワークの作成	113
9.4. 追加の FLOATING IP ネットワークの作成	114
9.5. オーバークラウドのプロバイダーネットワークの作成	114
9.6. オーバークラウドの検証	115
9.7. オーバークラウド環境の変更	116
9.8. 動的インベントリースクリプトの実行	117
9.9. オーバークラウドへの仮想マシンのインポート	118
9.10. コンピュートノードからのインスタンスの移行	119
9.11. オーバークラウドの削除防止	120
9.12. オーバークラウドの削除	120
9.13. トークンのフラッシュ間隔の確認	120
<b>第10章 ANSIBLE を使用したオーバークラウドの設定</b> .....	<b>122</b>
10.1. ANSIBLE ベースのオーバークラウド設定 (CONFIG-DOWNLOAD)	122
10.2. オーバークラウドの設定メソッドを CONFIG-DOWNLOAD に切り替える手順	122
10.3. 事前にプロビジョニング済みのノードでの CONFIG-DOWNLOAD の有効化	124

10.4. CONFIG-DOWNLOAD の作業ディレクトリーへのアクセスの有効化	125
10.5. CONFIG-DOWNLOAD のログと作業ディレクトリーの確認	125
10.6. CONFIG-DOWNLOAD の手動による実行	126
10.7. CONFIG-DOWNLOAD の無効化	127
10.8. 次のステップ	128
<b>第11章 オーバークラウドのスケーリング</b> .....	<b>129</b>
11.1. ノードのさらなる追加	129
11.2. コンピュートノードの削除	131
11.3. コンピュートノードの置き換え	133
11.4. コントローラーノードの置き換え	133
11.4.1. 事前のチェック	133
11.4.2. Ceph monitor デーモンの削除	135
11.4.3. ノードの置き換え	136
11.4.4. 手動での介入	138
11.4.5. オーバークラウドサービスの最終処理	140
11.4.6. L3 エージェントのルーターホスティングの最終処理	140
11.4.7. Compute サービスの最終処理	141
11.4.8. 結果	141
11.5. CEPH STORAGE ノードの置き換え	141
11.6. OBJECT STORAGE ノードの置き換え	141
11.7. ノードのブラックリスト登録	143
<b>第12章 ノードの再起動</b> .....	<b>145</b>
12.1. アンダークラウドノードの再起動	145
12.2. コントローラーノードおよびコンポーザブルノードの再起動	145
12.3. CEPH STORAGE (OSD) クラスターの再起動	146
12.4. コンピュートノードの再起動	146
<b>第13章 DIRECTOR の問題のトラブルシューティング</b> .....	<b>149</b>
13.1. ノード登録のトラブルシューティング	149
13.2. ハードウェアインタロスペクションのトラブルシューティング	149
13.3. WORKFLOW および EXECUTION のトラブルシューティング	151
13.4. オーバークラウドの作成のトラブルシューティング	153
13.4.1. Orchestration	153
13.4.2. Bare Metal Provisioning	153
13.4.3. デプロイメント後の設定	154
13.5. プロビジョニングネットワークでの IP アドレスの競合に対するトラブルシューティング	156
13.6. "NO VALID HOST FOUND" エラーのトラブルシューティング	157
13.7. オーバークラウド作成後のトラブルシューティング	158
13.7.1. オーバークラウドスタックの変更	158
13.7.2. コントローラーサービスのエラー	158
13.7.3. コンテナ化されたサービスのエラー	159
13.7.4. Compute サービスのエラー	161
13.7.5. Ceph Storage サービスのエラー	161
13.8. アンダークラウドの調整	161
13.9. SOS レポートの作成	163
13.10. アンダークラウドとオーバークラウドの重要なログ	163
<b>付録A SSL/TLS 証明書の設定</b> .....	<b>165</b>
A.1. 署名ホストの初期化	165
A.2. 認証局の作成	165
A.3. クライアントへの認証局の追加	165
A.4. SSL/TLS キーの作成	166

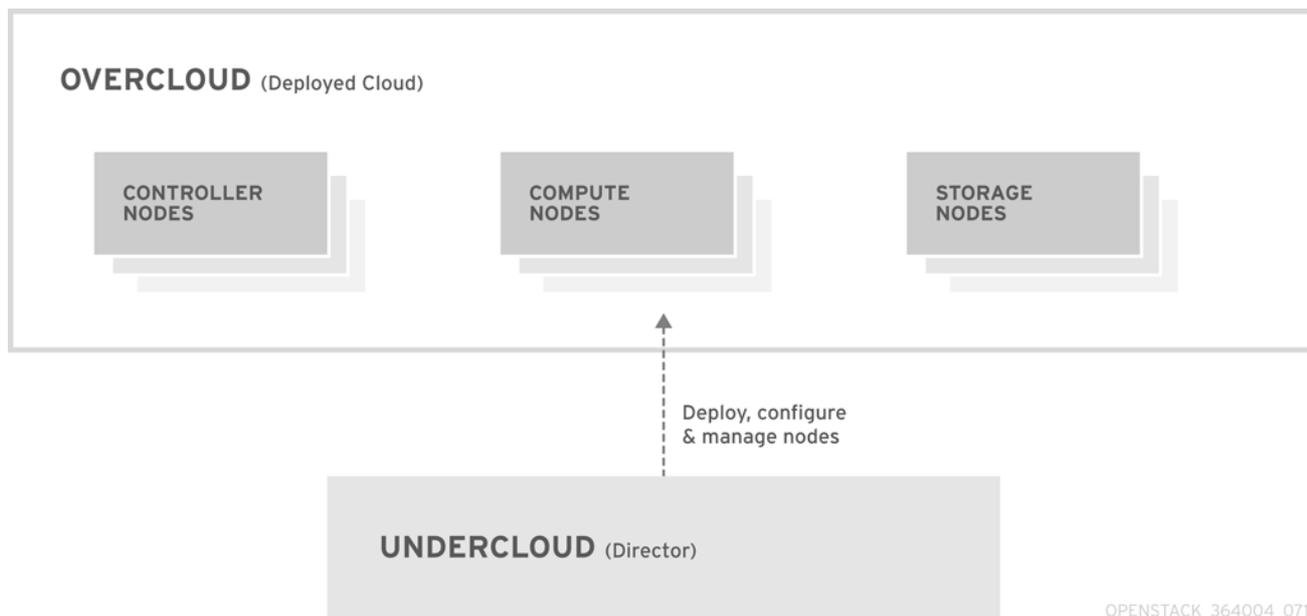
A.5. SSL/TLS 証明書署名要求の作成	166
A.6. SSL/TLS 証明書の作成	167
A.7. アンダークラウドで証明書を使用する場合	167
<b>付録B 電源管理ドライバー</b>	<b>169</b>
B.1. REDFISH	169
B.2. DELL REMOTE ACCESS CONTROLLER (DRAC)	169
B.3. INTEGRATED LIGHTS-OUT (ILO)	169
B.4. CISCO UNIFIED COMPUTING SYSTEM (UCS)	170
B.5. FUJITSU INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)	170
B.6. VIRTUAL BASEBOARD MANAGEMENT CONTROLLER (VBMC)	171
B.7. RED HAT VIRTUALIZATION	173
B.8. フェイクドライバー	174
<b>付録C 完全なディスクイメージ</b>	<b>175</b>
C.1. ベースのクラウドイメージのダウンロード	176
C.2. 環境変数の設定	176
C.3. ディスクレイアウトのカスタマイズ	177
C.3.1. パーティショニングスキーマの変更	178
C.3.2. イメージサイズの変更	179
C.4. セキュリティー強化された完全なディスクイメージの作成	180
C.5. セキュリティー強化された完全なディスクイメージのアップロード	180
<b>付録D 代替ブートモード</b>	<b>182</b>
D.1. 標準の PXE	182
D.2. UEFI ブートモード	182
<b>付録E プロファイルの自動タグ付け</b>	<b>184</b>
E.1. ポリシーファイルの構文	184
E.2. ポリシーファイルの例	186
E.3. ポリシーファイルのインポート	187
E.4. プロファイルの自動タグ付けのプロパティー	188
<b>付録F セキュリティーの強化</b>	<b>189</b>
F.1. HAPROXY の SSL/TLS の暗号およびルールの変更	189
<b>付録G POWER 版 RED HAT OPENSTACK PLATFORM (テクノロジープレビュー)</b>	<b>191</b>



## 第1章 はじめに

Red Hat OpenStack Platform director は、完全な OpenStack 環境のインストールおよび管理を行うためのツールセットです。director は、主に OpenStack プロジェクト TripleO (「OpenStack-On-OpenStack」の略語) をベースとしています。このプロジェクトは、OpenStack のコンポーネントを活用して、完全に機能する OpenStack 環境をインストールします。これには、OpenStack ノードとして使用するベアメタルシステムをプロビジョニングし、制御する新しい OpenStack のコンポーネントが含まれます。

Red Hat OpenStack Platform director は、アンダークラウドとオーバークラウドという 2 つの主要な概念を採用しています。以下の数項では、それぞれの概念について説明します。



### 1.1. アンダークラウド

アンダークラウドは、director の主要ノードで、OpenStack をインストールした単一システムです。このノードには、OpenStack 環境 (オーバークラウド) を構成する OpenStack ノードのプロビジョニング/管理のためのコンポーネントが含まれます。アンダークラウドを形成するコンポーネントは、複数の機能を提供します。

#### 環境のプランニング

アンダークラウドは、ユーザーが特定のノードロールを作成するためのプランニング機能を提供します。アンダークラウドには、コンピューター、コントローラー、さまざまなストレージロールなどのデフォルトのノードセットが含まれるだけでなく、カスタムロールを使用する機能も提供されています。さらに、各ノードロールにどの OpenStack Platform サービスを含めるかを選択でき、新しいノード種別をモデル化するか、独自のホストで特定のコンポーネントを分離する方法を提供します。

#### ベアメタルシステムの制御

アンダークラウドは、各ノードの Intelligent Platform Management Interface (IPMI) などのアウトバウンド管理インターフェースを使用して電源管理機能を制御し、PXE ベースのサービスを使用してハードウェア属性を検出し、各ノードに OpenStack をインストールします。この機能により、ベアメタルシステムを OpenStack ノードとしてプロビジョニングする方法が提供されます。電源管理ドライバの完全一覧については、「[付録B 電源管理ドライバ](#)」を参照します。

#### Orchestration

アンダークラウドでは、環境のプランとして機能する YAML テンプレートセットが提供されています。アンダークラウドは、これらのプランをインポートして、その指示に従い、目的の OpenStack

環境を作成します。このプランには、環境作成プロセスの途中にある特定のポイントで、カスタマイズを組み込めるようにするフックも含まれます。

## コマンドラインツールおよび Web UI

Red Hat OpenStack Platform director は、ターミナルベースのコマンドラインインターフェースまたは Web ベースのユーザーインターフェースで、これらのアンダークラウド機能を実行します。

## アンダークラウドのコンポーネント

アンダークラウドは、OpenStack のコンポーネントをベースのツールセットとして使用します。これには、以下のコンポーネントが含まれます。

- **OpenStack Identity (keystone):** director のコンポーネントの認証および承認
- **OpenStack Bare Metal (Ironic) および OpenStack Compute (Nova):** ベアメタルノードの管理
- **OpenStack Networking (Neutron) および Open vSwitch:** ベアメタルノードのネットワークの制御
- **OpenStack Image サービス (Glance):** ベアメタルマシンへ書き込むイメージの格納
- **OpenStack Orchestration (Heat) および Puppet:** director がオーバークラウドイメージをディスクに書き込んだ後のノードのオーケストレーションおよび設定
- **OpenStack Telemetry (Ceilometer):** 監視とデータの収集。これには、以下が含まれます。
  - **OpenStack Telemetry Metrics (gnocchi):** メトリック向けの時系列データベース
  - **OpenStack Telemetry Alarming (aodh):** モニタリング向けのアラームコンポーネント
  - **OpenStack Telemetry Event Storage (panko):** モニタリング向けのイベントストレージ
- **OpenStack Workflow サービス (mistral):** プランのインポートやデプロイなど、特定の director 固有のアクションに対してワークフローセットを提供します。
- **OpenStack Messaging Service (zaqar):** OpenStack Workflow サービスのメッセージサービスを提供します。
- **OpenStack Object Storage (swift):** 以下のさまざまな OpenStack Platform のコンポーネントに対してオブジェクトストレージを提供します。
  - **OpenStack Image サービスのイメージストレージ**
  - **OpenStack Bare Metal のイントロスペクションデータ**
  - **OpenStack Workflow サービスのデプロイメントプラン**

## 1.2. オーバークラウド

オーバークラウドは、アンダークラウドを使用して構築した Red Hat OpenStack Platform 環境で、以下のノード種別の1つまたは複数で構成されます。これには、作成予定の OpenStack Platform 環境をベースに定義するさまざまなロールが含まれます。アンダークラウドには、以下などのオーバークラウドノードのロールがデフォルトで含まれます。

### コントローラー

OpenStack 環境に管理、ネットワーク、高可用性の機能を提供するノード。理想的な OpenStack 環境には、このノード 3 台で高可用性クラスターを構成することを推奨します。

デフォルトのコントローラーノードには、以下のコンポーネントが含まれます。

- OpenStack Dashboard (horizon)
- OpenStack Identity (keystone)
- OpenStack Compute (nova) API
- OpenStack Networking (neutron)
- OpenStack Image サービス (glance)
- OpenStack Block Storage (cinder)
- OpenStack Object Storage (swift)
- OpenStack Orchestration (heat)
- OpenStack Telemetry (ceilometer)
- OpenStack Telemetry Metrics (gnocchi)
- OpenStack Telemetry Alarming (aodh)
- OpenStack Telemetry Event Storage (panko)
- OpenStack Clustering (sahara)
- OpenStack Shared File Systems (manila)
- OpenStack Bare Metal (ironic)
- MariaDB
- Open vSwitch
- 高可用性サービス向けの Pacemaker および Galera

### Compute

これらのノードは OpenStack 環境にコンピュートリソースを提供します。コンピュートノードをさらに追加して、環境を徐々にスケールアウトすることができます。デフォルトのコンピュートノードには、以下のコンポーネントが含まれます。

- OpenStack Compute (nova)
- KVM/QEMU
- OpenStack Telemetry (ceilometer) エージェント
- Open vSwitch

### ストレージ

OpenStack 環境にストレージを提供するノード。これには、以下のストレージ用のノードが含まれます。

- **Ceph Storage ノード**: ストレージクラスターを構成するために使用します。各ノードには、**Ceph Object Storage Daemon (OSD)** が含まれており、**Ceph Storage ノード**をデプロイする場合には、**director**により **Ceph Monitor**がコンピュータノードにインストールされます。
- **Block storage (Cinder)**: HA コントローラーノードの外部ブロックストレージとして使用します。このノードには、以下のコンポーネントが含まれます。
  - **OpenStack Block Storage (cinder)** ポリユーム
  - **OpenStack Telemetry (ceilometer)** エージェント
  - **Open vSwitch**
- **Object Storage (swift)**: これらのノードは、**OpenStack Swift**の外部ストレージ層を提供します。コントローラーノードは、**Swift** プロキシを介してこれらのノードにアクセスします。このノードには、以下のコンポーネントが含まれます。
  - **OpenStack Object Storage (swift)** のストレージ
  - **OpenStack Telemetry (ceilometer)** エージェント
  - **Open vSwitch**

### 1.3. 高可用性

**Red Hat OpenStack Platform director** は、**OpenStack Platform** 環境に高可用性サービスを提供するためにコントローラーノードクラスターを使用します。**director** は、各コントローラーノードにコンポーネントの複製セットをインストールし、それらをまとめて単一のサービスとして管理します。このタイプのクラスター構成では、1つのコントローラーノードが機能しなくなった場合にフォールバックするので、**OpenStack** のユーザーには一定の運用継続性が提供されます。

**OpenStack Platform director** は、複数の主要なソフトウェアを使用して、コントローラーノード上のコンポーネントを管理します。

- **Pacemaker**: **Pacemaker** はクラスターリソースマネージャーで、クラスター内の全ノードにおける **OpenStack** コンポーネントの可用性を管理/監視します。
- **HA Proxy**: クラスターに負荷分散およびプロキシサービスを提供します。
- **Galera**: クラスター全体の **OpenStack Platform** データベースを複製します。
- **Memcached**: データベースのキャッシュを提供します。



#### 注記

**Red Hat OpenStack Platform director** は複数のコントローラーノードの高可用性を一括に自動設定します。ただし、電源管理制御を有効化するには、ノードを手動で設定する必要があります。本ガイドでは、これらの手順を記載しています。

### 1.4. CEPH STORAGE

一般的に、**OpenStack** を使用する大規模な組織では、数千単位またはそれ以上のクライアントにサービスを提供します。**OpenStack** クライアントは、ブロックストレージリソースを消費する際には、それぞれに固有のニーズがある可能性が高く、**Glance** (イメージ)、**Cinder** (ポリユーム)、**Nova** (コンピュー

ト)を単一ノードにデプロイすると、数千単位のクライアントがある大規模なデプロイメントでの管理ができなくなる可能性があります。このような課題は、**OpenStack** をスケールアウトすることによって解決できます。

ただし、実際には、**Red Hat Ceph Storage** などのソリューションを活用して、ストレージ層を仮想化する必要もでてきます。ストレージ層の仮想化により、**Red Hat OpenStack Platform** のストレージ層を数十テラバイト規模からペタバイトさらにはエクサバイトのストレージにスケールアップすることが可能です。**Red Hat Ceph Storage** は、市販のハードウェアを使用しながらも、高可用性/高パフォーマンスのストレージ仮想化層を提供します。仮想化によってパフォーマンスが低下するというイメージがありますが、**Ceph** はブロックデバイスイメージをクラスター全体でオブジェクトとしてストライプ化するため、大きい**Ceph** のブロックデバイスイメージはスタンドアロンのディスクよりもパフォーマンスが優れているということになります。**Ceph** ブロックデバイスでは、パフォーマンスを強化するために、キャッシュ、**Copy On Write** クローン、**Copy On Read** クローンもサポートされています。

**Red Hat Ceph Storage** に関する情報は、[Red Hat Ceph Storage](#) を参照してください。

## 第2章 要件

本章では、**director** を使用して **Red Hat OpenStack Platform** をプロビジョニングする環境をセットアップするための主要な要件を記載します。これには、**director** のセットアップ/アクセス要件や **OpenStack** サービス用に **director** がプロビジョニングするホストのハードウェア要件が含まれます。



### 注記

**Red Hat OpenStack Platform** をデプロイする前には、利用可能なデプロイメントメソッドの特性を考慮することが重要です。詳しくは、「[Installing and Managing Red Hat OpenStack Platform](#)」の記事を参照してください。

## 2.1. 環境要件

### 最小要件

- Red Hat OpenStack Platform **director** 用のホストマシン 1 台
- Red Hat OpenStack Platform コンピュートノード用のホストマシン 1 台
- Red Hat OpenStack Platform コントローラーノード用のホストマシン 1 台

### 推奨要件

- Red Hat OpenStack Platform **director** 用のホストマシン 1 台
- Red Hat OpenStack Platform コンピュートノード用のホストマシン 3 台
- Red Hat OpenStack Platform コントローラーノード用のホストマシン 3 台
- クラスタ内に Red Hat Ceph Storage ノード用のホストマシン 3 台

以下の点に注意してください。

- 全ノードにはベアメタルシステムを使用することを推奨します。最低でも、コンピュートノードにはベアメタルシステムが必要です。
- **director** は電源管理制御を行うため、オーバークラウドのベアメタルシステムにはすべて、**Intelligent Platform Management Interface (IPMI)** が必要です。
- オーバークラウドのコンピュートノードを **POWER (ppc64le)** ハードウェアにデプロイする場合は、「[付録G POWER 版 Red Hat OpenStack Platform \(テクノロジープレビュー\)](#)」に記載の概要を一読してください。

## 2.2. アンダークラウドの要件

**director** をホストするアンダークラウドシステムは、オーバークラウド内の全ノードのプロビジョニングおよび管理を行います。

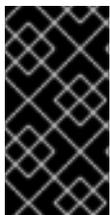
- Intel 64 または AMD64 CPU 拡張機能をサポートする、8 コア 64 ビット x86 プロセッサ
- 最小 16 GB の RAM
- ルートディスク上に最小 100 GB の空きディスク領域。この領域の内訳は以下のとおりです。

- コンテナイメージ用に 10 GB
- QCOW2 イメージの変換とノードのプロビジョニングプロセスのキャッシュ用に 10 GB
- 一般用途、ログの記録、メトリック、および将来の拡張用に 80 GB 以上
- 最小 2 枚の 1 Gbps ネットワークインターフェースカード。ただし、特にオーバークラウド環境で多数のノードをプロビジョニングする場合には、ネットワークトラフィックのプロビジョニング用に 10 Gbps インターフェースを使用することを推奨します。
- ホストのオペレーティングシステムに Red Hat Enterprise Linux の最新のマイナーバージョンがインストール済みであること
- ホスト上で SELinux が **Enforcing** モードで有効化されていること

### 2.2.1. 仮想化サポート

Red Hat は、以下のプラットフォームでのみ仮想アンダークラウドをサポートします。

プラットフォーム	備考
Kernel-based Virtual Machine (KVM)	認定済みのハイパーバイザーとしてリストされている Red Hat Enterprise Linux 5、6、7 でホストされていること
Red Hat Enterprise Virtualization	認定済みのハイパーバイザーとしてリストされている Red Hat Enterprise Virtualization 3.0、3.1、3.2、3.3、3.4、3.5、3.6、4.0 でホストされていること
Microsoft Hyper-V	<a href="#">Red Hat Customer Portal Certification Catalogue</a> に記載の Hyper-V のバージョンでホストされていること
VMware ESX および ESXi	<a href="#">Red Hat Customer Portal Certification Catalogue</a> に記載の ESX および ESXi のバージョンでホストされていること



#### 重要

Red Hat OpenStack Platform director では、ホストのオペレーティングシステムに最新バージョンの Red Hat Enterprise Linux を使用する必要があります。このため、仮想化プラットフォームは下層の Red Hat Enterprise Linux バージョンもサポートする必要があります。

#### 仮想マシンの要件

仮想アンダークラウドのリソース要件は、ベアメタルのアンダークラウドの要件と似ています。ネットワークモデル、ゲスト CPU 機能、ストレージのバックエンド、ストレージのフォーマット、キャッシュモードなどプロビジョニングの際には、さまざまなチューニングオプションを考慮する必要があります。

#### ネットワークの考慮事項

仮想アンダークラウドの場合は、以下にあげるネットワークの考慮事項に注意してください。

## 電源管理

アンダークラウドの仮想マシンには、オーバークラウドのノードにある電源管理のデバイスへのアクセスが必要です。これには、ノードの登録の際に、`pm_addr` パラメーターに IP アドレスを設定してください。

## プロビジョニングネットワーク

プロビジョニング (`ctlplane`) ネットワークに使用する NIC には、オーバークラウドのベアメタルノードの NIC に対する DHCP 要求をブロードキャストして、対応する機能が必要です。仮想マシンの NIC をベアメタルの NIC と同じネットワークに接続するブリッジを作成します。



### 注記

一般的に、ハイパーバイザーのテクノロジーにより、アンダークラウドが不明なアドレスのトラフィックを送信できない場合に問題が発生します。Red Hat Enterprise Virtualization を使用する場合には、`anti-mac-spoofing` を無効にしてこれを回避してください。VMware ESX または ESXi を使用している場合は、偽装転送を承諾してこれを回避します。

## アーキテクチャーの例

これは、KVM サーバーを使用した基本的なアンダークラウドの仮想化アーキテクチャー例です。これは、ネットワークやリソースの要件に合わせてビルド可能な基盤としての使用を目的としています。

KVM ホストは Linux ブリッジを 2 つ使用します。

### br-ex (eth0)

- アンダークラウドへの外部アクセスを提供します。
- 外部ネットワークの DHCP サーバーは、仮想 NIC (eth0) を使用してアンダークラウドにネットワーク設定を割り当てます。
- アンダークラウドがベアメタルサーバーの電源管理インターフェースにアクセスできるようにします。

### br-ctlplane (eth1)

- ベアメタルのオーバークラウドノードと同じネットワークに接続します。
- アンダークラウドは、仮想 NIC (eth1) を使用して DHCP および PXE ブートの要求に対応します。
- オーバークラウドのベアメタルサーバーは、このネットワークの PXE 経由で起動します。

KVM ホストには、以下のパッケージが必要です。

```
$ yum install libvirt-client libvirt-daemon qemu-kvm libvirt-daemon-driver-qemu libvirt-daemon-kvm virt-install bridge-utils rsync
```

以下のコマンドは、KVM ホストにアンダークラウドの仮想マシンとして、適切なブリッジに接続するための仮想 NIC を 2 つ作成します。

```
$ virt-install --name undercloud --memory=16384 --vcpus=4 --location /var/lib/libvirt/images/rhel-server-7.5-x86_64-dvd.iso --disk size=100 --network bridge=br-ex --network bridge=br-ctlplane --graphics=vnc --hvm --
```

```
os-variant=rhel7
```

このコマンドにより、**libvirt** ドメインが起動して **virt-manager** に接続し、段階を追ってインストールプロセスが進められます。または、以下のオプションを使用してキックスタートファイルを指定して、無人インストールを実行することもできます。

```
--initrd-inject=/root/ks.cfg --extra-args "ks=file:/ks.cfg"
```

インストールが完了したら、**root** ユーザーとしてインスタンスに SSH 接続して、「[4章 アンダークラウドのインストール](#)」の手順に従います。

## バックアップ

以下のように、仮想アンダークラウドをバックアップするためのソリューションは複数あります。

- **オプション 1:** 『[Back Up and Restore the Director Undercloud](#)』 ガイドの説明に従います。
- **オプション 2:** アンダークラウドをシャットダウンして、アンダークラウドの仮想マシンストレージのバックアップのコピーを取ります。
- **オプション 3:** ハイパーバイザーがライブまたはアトミックのスナップショットをサポートする場合は、アンダークラウドの仮想マシンのスナップショットを作成します。

KVM サーバーを使用する場合は、以下の手順でスナップショットを作成してください。

1. **qemu-guest-agent** がアンダークラウドのゲスト仮想マシンで実行していることを確認してください。
2. 実行中の仮想マシンのライブスナップショットを作成します。

```
$ virsh snapshot-create-as --domain undercloud --disk-only --atomic --quiesce
```

1. QCOW バッキングファイルのコピー (読み取り専用) を作成します。

```
$ rsync --sparse -avh --progress /var/lib/libvirt/images/undercloud.qcow2 1.qcow2
```

1. QCOW オーバーレイファイルをバッキングファイルにマージして、アンダークラウドの仮想マシンが元のファイルを使用するように切り替えます。

```
$ virsh blockcommit undercloud vda --active --verbose --pivot
```

## 2.3. ネットワーク要件

アンダークラウドのホストには、最低でも 2 つのネットワークが必要です。

- **プロビジョニングネットワーク:** オーバークラウドで使用するベアメタルシステムの検出がしやすくなるように、DHCP および PXE ブート機能を提供します。このネットワークは通常、**director** が PXE ブートおよび DHCP の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 VLAN を使用する必要があります。一部のサーバーのハードウェアの BIOS は、VLAN からの PXE ブートをサポートしていますが、その BIOS が、ブート後に VLAN をネイティブ VLAN に変換する機能もサポートする必要があります。この機能がサポートされていない場合には、アンダークラウドに到達できません。現在この機能を完全にサポートして

いるサーバーハードウェアはごく一部です。プロビジョニングネットワークは、オーバークラウドノード上で **Intelligent Platform Management Interface (IPMI)** により電源管理を制御するのに使用するネットワークでもあります。

- 外部ネットワーク: 全ノードへのリモート接続に使用する別個のネットワーク。このネットワークに接続するこのインターフェースには、静的または外部の **DHCP** サービス経由で動的に定義された、ルーティング可能な **IP** アドレスが必要です。

これは、必要なネットワークの最小数を示します。ただし、**director** は他の **Red Hat OpenStack Platform** ネットワークトラフィックをその他のネットワーク内に分離することができます。**Red Hat OpenStack Platform** は、ネットワークの分離に物理インターフェースとタグ付けされた **VLAN** の両方をサポートしています。

以下の点に注意してください。

- 標準的な最小限のオーバークラウドのネットワーク構成には、以下が含まれます。
  - シングル **NIC** 構成: ネイティブの **VLAN** および異なる種別のオーバークラウドネットワークのサブネットを使用するタグ付けされた **VLAN** 上にプロビジョニングネットワーク用の **NIC** を1つ。
  - デュアル **NIC** 構成: プロビジョニングネットワーク用の **NIC** を1つと、外部ネットワーク用の **NIC** を1つ。
  - デュアル **NIC** 構成: ネイティブの **VLAN** 上にプロビジョニングネットワーク用の **NIC** を1つと、異なる種別のオーバークラウドネットワークのサブネットを使用するタグ付けされた **VLAN** 用の **NIC** を1つ。
  - 複数 **NIC** 構成: 各 **NIC** は、異なる種別のオーバークラウドネットワークのサブセットを使用します。
- 追加の物理 **NIC** は、個別のネットワークの分離、ボンディングインターフェースの作成、タグ付けされた **VLAN** トラフィックの委譲に使用することができます。
- ネットワークトラフィックの種別を分離するのに **VLAN** を使用している場合には、**802.1Q** 標準をサポートするスイッチを使用してタグ付けされた **VLAN** を提供します。
- オーバークラウドの作成時には、全オーバークラウドマシンで1つの名前を使用して **NIC** を参照します。理想としては、混乱を避けるため、対象のネットワークごとに、各オーバークラウドノードで同じ **NIC** を使用してください。たとえば、プロビジョニングネットワークにはプライマリー **NIC** を使用して、**OpenStack** サービスにはセカンダリー **NIC** を使用します。
- プロビジョニングネットワークの **NIC** は **director** マシン上でリモート接続に使用する **NIC** とは異なります。**director** のインストールでは、プロビジョニング **NIC** を使用してブリッジが作成され、リモート接続はドロップされます。**director** システムへリモート接続する場合には、外部 **NIC** を使用します。
- プロビジョニングネットワークには、環境のサイズに適した **IP** 範囲が必要です。以下のガイドラインを使用して、この範囲に含めるべき **IP** アドレスの総数を決定してください。
  - プロビジョニングネットワークに接続されているノード1台につき最小で1 **IP** アドレスを含めます。
  - 高可用性を設定する予定がある場合には、クラスターの仮想 **IP** 用に追加の **IP** アドレスを含めます。
  - 環境のスケーリング用の追加の **IP** アドレスを範囲に追加します。



### 注記

プロビジョニングネットワーク上で IP アドレスが重複するのを避ける必要があります。詳しい説明は、「[ネットワークのプランニング](#)」を参照してください。



### 注記

ストレージ、プロバイダー、テナントネットワークの IP アドレスの使用範囲をプランニングすることに関する情報は、『[Networking Guide](#)』を参照してください。

- すべてのオーバークラウドシステムをプロビジョニング NIC から PXE ブートするように設定して、同システム上の外部 NIC およびその他の NIC の PXE ブートを無効にします。また、プロビジョニング NIC の PXE ブートは、ハードディスクや CD/DVD ドライブよりも優先されるように、起動順序の最上位に指定します。
- オーバークラウドのベアメタルシステムにはすべて、**Intelligent Platform Management Interface (IPMI)** などのサポート対象の電源管理インターフェースが必要です。このインターフェースにより、**director** は各ノードの電源管理を制御することが可能となります。
- 各オーバークラウドシステムの詳細 (プロビジョニング NIC の MAC アドレス、IPMI NIC の IP アドレス、IPMI ユーザー名、IPMI パスワード) をメモしてください。この情報は、後でオーバークラウドノードを設定する際に役立ちます。
- インスタンスが外部のインターネットからアクセス可能である必要がある場合には、パブリックネットワークから **Floating IP** アドレスを割り当て、そのアドレスをインスタンスに関連付けます。インスタンスは、引き続きプライベートの IP アドレスを確保しますが、ネットワークトラフィックは **NAT** を使用して、**Floating IP** アドレスに到達します。**Floating IP** アドレスは、複数のプライベート IP アドレスではなく、単一のインスタンスにのみ割り当て可能である点に注意してください。ただし、**Floating IP** アドレスは、単一のテナントで使用するよう確保され、そのテナントは必要に応じて特定のインスタンスに関連付け/関連付け解除することができます。この構成を使用すると、インフラストラクチャーが外部のインターネットに公開されるので、適切なセキュリティープラクティスを順守しているかどうかを確認する必要があります。
- 1つのブリッジには単一のインターフェースまたは単一のボンディングのみをメンバーにすると、**Open vSwitch** でネットワークループが発生するリスクを緩和することができます。複数のボンディングまたはインターフェースが必要な場合には、複数のブリッジを設定することが可能です。
- オーバークラウドノードが **Red Hat Content Delivery Network** やネットワークタイムサーバーなどの外部のサービスに接続できるようにするには、**DNS** によるホスト名解決を使用することを推奨します。

## 重要

OpenStack Platform の実装のセキュリティーレベルは、その環境のセキュリティーレベルと同等です。ネットワーク環境内の適切なセキュリティー原則に従って、ネットワークアクセスが正しく制御されるようにします。以下に例を示します。

- ネットワークのセグメント化を使用して、ネットワークトラフィックを軽減し、機密データを分離します。フラットなネットワークはセキュリティーレベルがはるかに低くなります。
- サービスアクセスとポートを最小限に制限します。
- 適切なファイアウォールルールとパスワードが使用されるようにします。
- SELinux が有効化されていることを確認します。

システムのセキュリティー保護については、以下のドキュメントを参照してください。

- 『[Red Hat Enterprise Linux 7 セキュリティーガイド](#)』
- 『[Red Hat Enterprise Linux 7 SELinux ユーザーおよび管理者のガイド](#)』

## 2.4. オーバークラウドの要件

以下の項では、オーバークラウドのインストール内の個別システムおよびノードの要件について詳しく説明します。

### 2.4.1. Compute ノードの要件

コンピュータノードは、仮想マシンインスタンスが起動した後にそれらを稼働させる役割を果たします。コンピュータノードは、ハードウェアの仮想化をサポートする必要があります。また、ホストする仮想マシンインスタンスの要件をサポートするのに十分なメモリーとディスク容量も必要です。

#### プロセッサ

- Intel 64 または AMD64 CPU 拡張機能をサポートする 64 ビット x86 プロセッサで Intel VT または AMD-V のハードウェア仮想化拡張機能が有効化されていること。このプロセッサには最小でも 4 つのコアが搭載されていることを推奨しています。
- IBM POWER 8 プロセッサ

#### メモリー

最小で 6 GB のメモリー。これに、仮想マシンインスタンスに割り当てるメモリー容量に基づいて、追加の RAM を加算します。

#### ディスク領域

最小 40 GB の空きディスク領域

#### ネットワークインターフェースカード

最小 1 枚の 1 Gbps ネットワークインターフェースカード (実稼働環境では最低でも NIC を 2 枚使用することを推奨)。タグ付けされた VLAN トラフィックを委譲する場合や、ボンディングインターフェース向けの場合には追加のネットワークインターフェースを使用します。

#### 電源管理

各コンピュータノードには、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェースがサーバーのマザーボードに搭載されている必要があります。

## 2.4.2. コントローラーノードの要件

コントローラーノードは、Red Hat OpenStack Platform 環境の中核となるサービス (例: Horizon Dashboard、バックエンドのデータベースサーバー、Keystone 認証、高可用性サービスなど) をホストする役割を果たします。

### プロセッサ

Intel 64 または AMD64 CPU 拡張機能のサポートがある 64 ビットの x86 プロセッサ

### メモリー

最小のメモリー容量は 32 GB です。ただし、推奨のメモリー容量は、仮想 CPU の数によって異なります (CPU コアをハイパースレッディングの値で乗算した数値に基づいています)。以下の計算を参考にしてください。

- **コントローラーの最小メモリー容量の算出:**
  - 1 仮想 CPU あたり 1.5 GB のメモリーを使用します。たとえば、仮想 CPU が 48 個あるマシンにはメモリーは 72 GB 必要です。
- **コントローラーの推奨メモリー容量の算出:**
  - 1 仮想 CPU あたり 3 GB のメモリーを使用します。たとえば、仮想 CPU が 48 個あるマシンにはメモリーは 144 GB 必要です。

メモリーの要件に関する詳しい情報は、Red Hat カスタマーポータルで「[Red Hat OpenStack Platform で、クラスター化されたコントローラーに必要なハードウェア \(CPU、メモリー\) 要件](#)」の記事を参照してください。

### ディスクストレージとレイアウト

デフォルトでは、Telemetry (gnocchi) と Object Storage (swift) のサービスはいずれもコントローラーにインストールされ、ルートディスクを使用するように設定されます。これらのデフォルトは、コモディティーハードウェア上に構築される小型のオーバークラウドのデプロイに適しています。これは、概念検証およびテストの標準的な環境です。これらのデフォルトにより、最小限のプランニングでオーバークラウドをデプロイすることができますが、ワークロードキャパシティーとパフォーマンスの面ではあまり優れていません。

ただし、Telemetry がストレージに絶えずアクセスするため、エンタープライズ環境では、これによって大きなボトルネックが生じる可能性があります。これにより、ディスク I/O が過度に使用されて、その他すべてのコントローラーサービスに深刻な影響をもたらします。このタイプの環境では、オーバークラウドのプランニングを行って、適切に設定する必要があります。

Red Hat は、Telemetry と Object Storage の両方の推奨設定をいくつか提供しています。詳しくは、『[Deployment Recommendations for Specific Red Hat OpenStack Platform Services](#)』を参照してください。

### ネットワークインターフェースカード

最小 2 枚の 1 Gbps ネットワークインターフェースカード。タグ付けされた VLAN トラフィックを委譲する場合や、ボンディングインターフェース向けの場合には追加のネットワークインターフェースを使用します。

### 電源管理

各コントローラーノードには、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェースがサーバーのマザーボードに搭載されている必要があります。

## 2.4.3. Ceph Storage ノードの要件

Ceph Storage ノードは、Red Hat OpenStack Platform 環境でオブジェクトストレージを提供する役割を果たします。

### プロセッサ

Intel 64 または AMD64 CPU 拡張機能のサポートがある 64 ビットの x86 プロセッサ

### メモリー

メモリー要件はストレージ容量によって異なります。ハードディスク容量 1 TB あたり最小で 1 GB のメモリーを使用するのが理想的です。

### ディスク領域

ストレージ要件はメモリーの容量によって異なります。ハードディスク容量 1 TB あたり最小で 1 GB のメモリーを使用するのが理想的です。

### ディスクのレイアウト

Red Hat Ceph Storage ノードの推奨設定では、少なくとも 3 つ、またはそれ以上のディスクを以下と同様のレイアウトで構成する必要があります。

- **/dev/sda:** ルートディスク。director は、主なオーバークラウドイメージをディスクにコピーします。
- **/dev/sdb:** ジャーナルディスク。このディスクは、**/dev/sdb1**、**/dev/sdb2**、**/dev/sdb3** などのように、Ceph OSD ジャーナル向けにパーティションを分割します。ジャーナルディスクは通常、システムパフォーマンスの向上に役立つ Solid State Drive (SSD) です。
- **/dev/sdc** 以降: OSD ディスク。ストレージ要件で必要な数のディスクを使用します。



### 注記

Red Hat OpenStack Platform director では **ceph-ansible** が使われますが、OSD を Ceph Storage ノードのルートディスクにインストールすることには対応していません。つまり、サポートされる Ceph Storage ノード用に少なくとも 2 つのディスクが必要になります。

### ネットワークインターフェースカード

最小で 1 x 1 Gbps ネットワークインターフェースカード (実稼働環境では、最低でも NIC を 2 つ以上使用することを推奨します)。ボンディングインターフェース向けの場合や、タグ付けされた VLAN トラフィックを委譲する場合には、追加のネットワークインターフェースを使用します。特に大量のトラフィックにサービスを提供する OpenStack Platform 環境を構築する場合には、ストレージノードには 10 Gbps インターフェースを使用することを推奨します。

### 電源管理

各コントローラーノードには、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェースがサーバーのマザーボードに搭載されている必要があります。

Ceph Storage クラスターを使用するオーバークラウドのインストールについては、『[Deploying an Overcloud with Containerized Red Hat Ceph](#)』ガイドを参照してください。

## 2.4.4. Object Storage ノードの要件

Object Storage ノードは、オーバークラウドのオブジェクトストレージ層を提供します。Object Storage プロキシは、コントローラーノードにインストールされます。ストレージ層には、ノードごとに複数のディスクを持つベアメタルノードが必要です。

## プロセッサ

Intel 64 または AMD64 CPU 拡張機能のサポートがある 64 ビットの x86 プロセッサ

## メモリー

メモリー要件はストレージ容量によって異なります。ハードディスク容量 1 TB あたり最小で 1 GB のメモリーを使用するのが理想的です。最適なパフォーマンスを得るには、特にワークロードが小さいファイル (100 GB 未満) の場合にはハードディスク容量 1 TB あたり 2 GB のメモリーを使用することを推奨します。

## ディスク領域

ストレージ要件は、ワークロードに必要とされる容量により異なります。アカウントとコンテナのデータを保存するには SSD ドライブを使用することを推奨します。アカウントおよびコンテナデータとオブジェクトの容量比率は、約 1% です。たとえば、ハードドライブの容量 100 TB ごとに、アカウントおよびコンテナデータの SSD 容量は 1 TB 用意するようにします。

ただし、これは保存したデータの種類により異なります。保存するオブジェクトサイズの大半が小さい場合には、SSD の容量がさらに必要です。オブジェクトが大きい場合には (ビデオ、バックアップなど)、SSD の容量を減らします。

## ディスクのレイアウト

推奨のノード設定には、以下のようなディスクレイアウトが必要です。

- **/dev/sda:** ルートディスク。director は、主なオーバークラウドイメージをディスクにコピーします。
- **/dev/sdb:** アカウントデータに使用します。
- **/dev/sdc:** コンテナデータに使用します。
- **/dev/sdc 以降:** オブジェクトサーバーディスク。ストレージ要件で必要な数のディスクを使用します。

## ネットワークインターフェースカード

最小 2 枚の 1 Gbps ネットワークインターフェースカード。タグ付けされた VLAN トラフィックを委譲する場合や、ボンディングインターフェース向けの場合には追加のネットワークインターフェースを使用します。

## 電源管理

各コントローラーノードには、Intelligent Platform Management Interface (IPMI) 機能などのサポート対象の電源管理インターフェースがサーバーのマザーボードに搭載されている必要があります。

## 2.5. リポジトリーの要件

アンダークラウドおよびオーバークラウドにはいずれも、Red Hat コンテンツ配信ネットワーク (CDN) か Red Hat Satellite 5 または 6 を利用した Red Hat リポジトリーへのアクセスが必要です。Red Hat Satellite サーバーを使用する場合は、必要なリポジトリーをお使いの OpenStack Platform 環境に同期してください。以下の CDN チャンネル名一覧をガイドとして使用してください。

表 2.1 OpenStack Platform リポジトリー

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 7 Server (RPMS)	<b>rhel-7-server-rpms</b>	x86_64 システム用ベースオペレーティングシステムのリポジトリー

名前	リポジトリ	要件の説明
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	<b>rhel-7-server-extras-rpms</b>	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	<b>rhel-7-server-rh-common-rpms</b>	Red Hat OpenStack Platform のデプロイと設定ツールが含まれます。
Red Hat Satellite Tools for RHEL 7 Server RPMs x86_64	<b>rhel-7-server-satellite-tools-6.3-rpms</b>	Red Hat Satellite 6 でのホスト管理ツール
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	<b>rhel-ha-for-rhel-7-server-rpms</b>	Red Hat Enterprise Linux の高可用性ツール。コントローラーノードの高可用性に使用します。
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	<b>rhel-7-server-openstack-13-rpms</b>	Red Hat OpenStack Platform のコアリポジトリ。Red Hat OpenStack Platform director のパッケージも含まれます。
Red Hat Ceph Storage OSD 3 for Red Hat Enterprise Linux 7 Server (RPMs)	<b>rhel-7-server-rhceph-3-osd-rpms</b>	(Ceph Storage ノード向け) Ceph Storage Object Storage デーモンのリポジトリ。Ceph Storage ノードにインストールします。
Red Hat Ceph Storage MON 3 for Red Hat Enterprise Linux 7 Server (RPMs)	<b>rhel-7-server-rhceph-3-mon-rpms</b>	(Ceph Storage ノード向け) Ceph Storage Monitor デーモンのリポジトリ。Ceph Storage ノードを使用して OpenStack 環境にあるコントローラーノードにインストールします。
Red Hat Ceph Storage Tools 3 for Red Hat Enterprise Linux 7 Server (RPMs)	<b>rhel-7-server-rhceph-3-tools-rpms</b>	Ceph Storage クラスターと通信するためのノード用のツールを提供します。このリポジトリは、Ceph Storage クラスターを使用するオーバークラウドをデプロイする際に、全ノードに有効化する必要があります。
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	<b>rhel-7-server-nfv-rpms</b>	NFV 向けのリアルタイム KVM (RT-KVM) のリポジトリ。リアルタイムカーネルを有効化するためのパッケージが含まれています。このリポジトリは、RT-KVM 対象の全コンピュートノードで有効化する必要があります。

## IBM POWER 用の OpenStack Platform リポジトリ

これらのリポジトリーは、「[付録G POWER 版 Red Hat OpenStack Platform \(テクノロジープレビュー\)](#)」で説明する機能に使われます。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux for IBM Power, little endian	<b>rhel-7-for-power-le-rpms</b>	ppc64le システム用ベースオペレーティングシステムのリポジトリー
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	<b>rhel-7-server-openstack-13-for-power-le-rpms</b>	ppc64le システム用 Red Hat OpenStack Platform のコアリポジトリー



### 注記

オフラインのネットワークで Red Hat OpenStack Platform 環境用リポジトリーを設定するには、「[オフライン環境で Red Hat OpenStack Platform Director を設定する](#)」の記事を参照してください。

## 第3章 オーバークラウドのプランニング

以下の項には、ノードロールの定義、ネットワークトポロジーのプランニング、ストレージなど、Red Hat OpenStack Platform 環境のさまざまな面のプランニングに関するガイドラインを記載します。

### 3.1. ノードのデプロイメントロールのプランニング

**director** はオーバークラウドの構築に、デフォルトで複数のノード種別を提供します。これらのノード種別は以下のとおりです。

#### コントローラー

環境を制御するための主要なサービスを提供します。これには、**Dashboard (Horizon)**、**認証 (Keystone)**、**イメージストレージ (Glance)**、**ネットワーク (Neutron)**、**オーケストレーション (Heat)**、高可用性サービスが含まれます。高可用性の場合は、Red Hat OpenStack Platform 環境にコントローラーノードが3台必要です。



#### 注記

1台のノードで構成される環境はテスト目的で使用することができます。2台のノードまたは3台以上のノードで構成される環境はサポートされません。

#### Compute

ハイパーバイザーとして機能し、環境内で仮想マシンを実行するのに必要な処理能力を提供する物理サーバー。基本的な Red Hat OpenStack Platform 環境には少なくとも1つのコンピュータノードが必要です。

#### Ceph-Storage

Red Hat Ceph Storage を提供するホスト。**Ceph Storage** ホストはクラスターに追加され、クラスターをスケールリングします。このデプロイメントロールはオプションです。

#### Cinder-Storage

OpenStack の **Cinder** サービスに外部ブロックストレージを提供するホスト。このデプロイメントロールはオプションです。

#### Swift-Storage

OpenStack の **Swift** サービスに外部オブジェクトストレージを提供するホスト。このデプロイメントロールはオプションです。

以下の表には、オーバークラウドの構成例と各シナリオで使用するノードタイプの定義をまとめています。

表3.1 各種シナリオに使用するノードデプロイメントロール

	コントローラー	Compute	Ceph-Storage	Swift-Storage	Cinder-Storage	合計
小規模のオーバークラウド	1	1	-	-	-	2
中規模のオーバークラウド	1	3	-	-	-	4

追加のオブジェクトおよびブロックストレージのある中規模のオーバークラウド	1	3	-	1	1	6
高可用性の中規模オーバークラウド	3	3	-	-	-	6
高可用性で Ceph Storage のある中規模オーバークラウド	3	3	3	-	-	9

さらに、個別のサービスをカスタムのロールに分割するかどうかを検討します。コンポーザブルロールのアーキテクチャに関する詳しい情報は『**Advanced Overcloud Customization**』ガイドの「[Composable Services and Custom Roles](#)」を参照してください。

### 3.2. ネットワークのプランニング

ロールとサービスを適切にマッピングして相互に正しく通信できるように、環境のネットワークポロジおよびサブネットのプランニングを行うことが重要です。Red Hat OpenStack Platform では、自動的に動作してソフトウェアベースのネットワーク、静的/Floating IP アドレス、DHCP を管理する Neutron ネットワークサービスを使用します。director は、オーバークラウド環境の各コントローラーノードに、このサービスをデプロイします。

Red Hat OpenStack Platform は、さまざまなサービスをマッピングして、お使いの環境の各種サブネットに割り当てられたネットワークトラフィックの種別を分類します。これらのネットワークトラフィック種別は以下のとおりです。

表3.2 ネットワーク種別の割り当て

ネットワーク種別	説明	そのネットワーク種別を使用するノード
IPMI	ノードの電源管理に使用するネットワーク。このネットワークは、アンダークラウドのインストール前に事前定義されます。	全ノード

プロビジョニング/コントロールプレーン	<b>director</b> は、このネットワークラフィック種別を使用して、PXEブートで新規ノードをデプロイし、オーバークラウドベアメタルサーバーに <b>OpenStack Platform</b> のインストールをオーケストレーションします。このネットワークは、アンダークラウドのインストール前に事前定義されます。	全ノード
内部 API	内部 API ネットワークは、API 通信、RPC メッセージ、データベース通信経由で <b>OpenStack</b> のサービス間の通信を行う際に使用します。	コントローラー、コンピューター、Cinder Storage、Swift Storage
テナント	<b>Neutron</b> は、VLAN 分離 (各テナントネットワークがネットワーク VLAN) または VXLAN か GRE 経由のトンネリングを使用した独自のネットワークを各テナントに提供します。ネットワークラフィックは、テナントのネットワークごとに分割されます。テナントネットワークにはそれぞれ IP サブネットが割り当てられています。また、ネットワーク名前空間が複数あると、複数のテナントネットワークが同じアドレスを使用できるので、競合は発生しません。	コントローラー、コンピューター
ストレージ	<b>Block Storage</b> 、NFS、iSCSI など。理想的には、これはパフォーマンス上の理由から、完全に別のスイッチファブリックに分離した方がよいでしょう。	全ノード

ストレージ管理	<p><b>OpenStack Object Storage (swift)</b> は、このネットワークを使用して、参加するレプリカノード間でデータオブジェクトを同期しません。プロキシサービスは、ユーザー要求と背後にあるストレージ層の間の仲介インターフェースとして機能します。プロキシは、受信要求を受け取り、必要なレプリカの位置を特定して要求データを取得します。<b>Ceph</b> バックエンドを使用するサービスは、<b>Ceph</b> と直接対話せずにフロントエンドのサービスを使用するため、ストレージ管理ネットワーク経由で接続を確立します。<b>RBD</b> ドライバーは例外で、このトラフィックは直接 <b>Ceph</b> に接続する点に注意してください。</p>	コントローラー、 <b>Ceph Storage</b> 、 <b>Cinder Storage</b> 、 <b>Swift Storage</b>
外部	<p>グラフィカルシステム管理用の <b>OpenStack Dashboard (Horizon)</b>、<b>OpenStack</b> サービス用のパブリック API をホストして、インスタンスへの受信トラフィック向けに <b>SNAT</b> を実行します。外部ネットワークがプライベート IP アドレスを使用する場合には (<b>RFC-1918</b> に準拠)、インターネットからのトラフィックに対して、さらに <b>NAT</b> を実行する必要があります。</p>	コントローラー

Floating IP	<p>受信トラフィックが Floating IP アドレスとテナントネットワーク内のインスタンスに実際に割り当てられた IP アドレスとの間の 1 対 1 の IP アドレスマッピングを使用してインスタンスに到達できるようにします。外部ネットワークからは分離した VLAN 上で Floating IP をホストする場合には、Floating IP VLAN をコントローラーノードにトランキングして、オーバークラウドの作成後に Neutron を介して VLAN を追加します。これにより、複数のブリッジに接続された複数の Floating IP ネットワークを作成する手段が提供されます。VLAN は、トランキングされますが、インターフェースとしては設定されません。その代わりに、Neutron は各 Floating IP ネットワークに選択したブリッジ上の VLAN セグメンテーション ID を使用して、OVS ポートを作成します。</p>	コントローラー
管理	<p>SSH アクセス、DNS トラフィック、NTP トラフィックなどのシステム管理機能を提供します。このネットワークは、コントローラー以外のノード用のゲートウェイとしても機能します。</p>	全ノード

一般的な Red Hat OpenStack Platform のシステム環境では通常、ネットワーク種別の数は物理ネットワークのリンク数を超えます。全ネットワークを正しいホストに接続するには、オーバークラウドは VLAN タグ付けを使用して、1つのインターフェースに複数のネットワークを提供します。ネットワークの多くは、サブネットが分離されていますが、インターネットアクセスまたはインフラストラクチャーにネットワーク接続ができるようにルーティングを提供するレイヤー 3 のゲートウェイが必要です。



## 注記

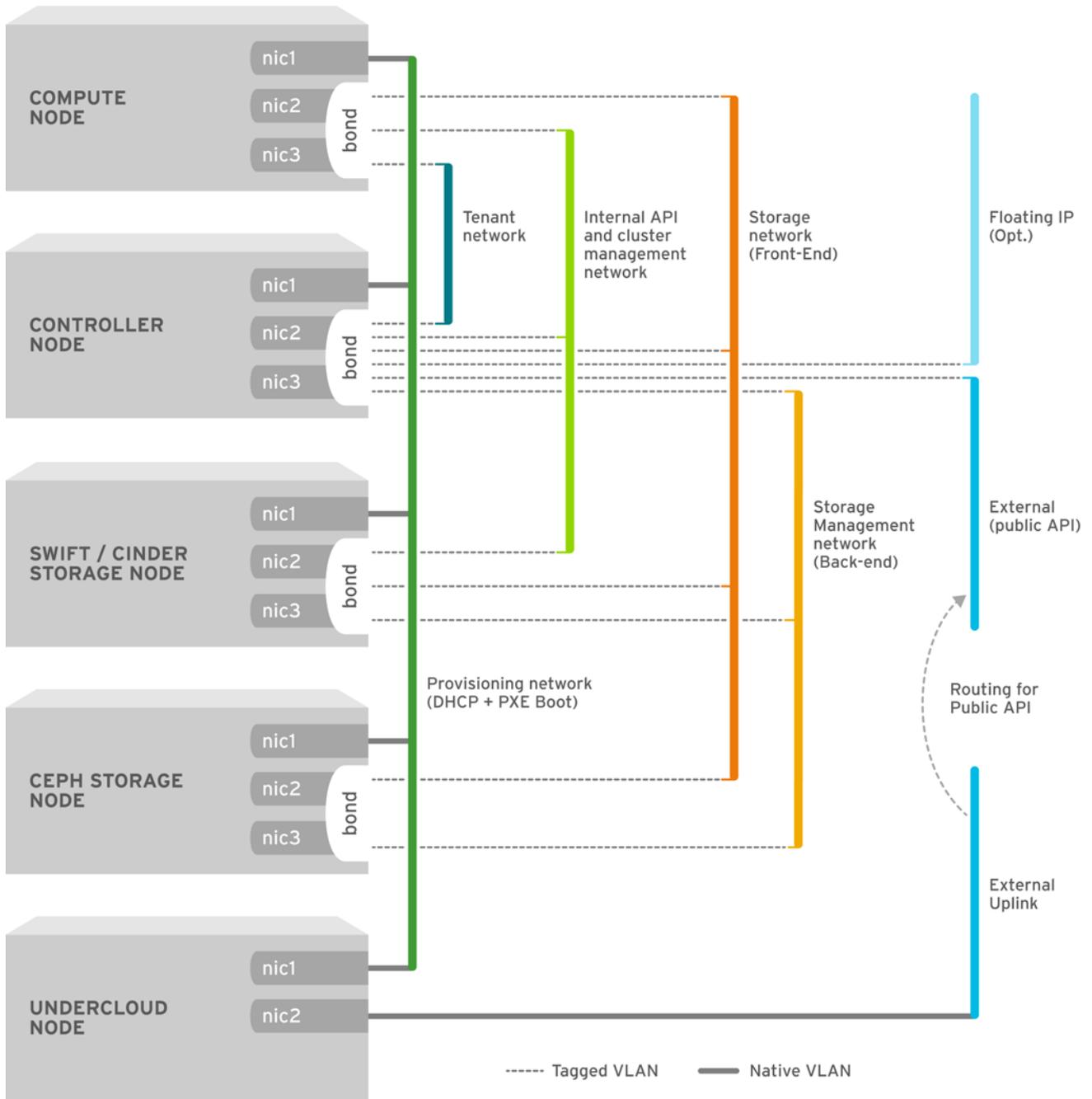
デプロイ時に `neutron VLAN` モード (トンネリングは無効) を使用する場合でも、プロジェクトネットワーク (GRE または VXLAN でトンネリング) をデプロイすることを推奨します。これには、デプロイ時にマイナーなカスタマイズを行う必要があり、将来ユーティリティーネットワークまたは仮想化ネットワークとしてトンネルネットワークを使用するためのオプションが利用可能な状態になります。VLAN を使用してテナントネットワークを作成することは変わりませんが、テナントの VLAN を消費せずに特別な用途のネットワーク用に VXLAN トンネルを作成することも可能です。また、テナント VLAN を使用するデプロイメントに VXLAN 機能を追加することは可能ですが、サービスを中断せずにテナント VLAN を既存のオーバークラウドに追加することはできません。

`director` は、トラフィック種別の中から 6 つを特定のサブネットまたは VLAN にマッピングする方法を提供します。このようなトラフィック種別には、以下が含まれます。

- 内部 API
- ストレージ
- ストレージ管理
- テナントネットワーク
- 外部
- 管理

未割り当てのネットワークは、プロビジョニングネットワークと同じサブネットに自動的に割り当てられます。

下図では、ネットワークが個別の VLAN に分離されたネットワークトポロジーの例を紹介しています。各オーバークラウドノードは、ボンディングで 2 つ (**nic2** および **nic3**) のインターフェースを使用して、対象の VLAN 経由でこれらのネットワークを提供します。また、各オーバークラウドのノードは、ネイティブの VLAN (**nic1**) を使用するプロビジョニングネットワークでアンダークラウドと通信します。



OPENSTACK\_364029\_0715

以下の表は、異なるネットワークのレイアウトをマッピングするネットワークトラフィック例が記載されています。

表3.3 ネットワークマッピング

	マッピング	インターフェースの総数	VLANの総数

外部アクセスのあるフラットネットワーク	<p>ネットワーク 1: プロビジョニング、内部 API、ストレージ、ストレージ管理、テナントネットワーク</p> <p>ネットワーク 2: 外部、Floating IP (オーバークラウドの作成後にマッピング)</p>	2	2
分離ネットワーク	<p>ネットワーク 1: プロビジョニング</p> <p>ネットワーク 2: 内部 API</p> <p>ネットワーク 3: テナントネットワーク</p> <p>ネットワーク 4: ストレージ</p> <p>ネットワーク 5: ストレージ管理</p> <p>ネットワーク 6: ストレージ管理</p> <p>ネットワーク 7: 外部、Floating IP (オーバークラウドの作成後にマッピング)</p>	3 (ボンディングインターフェース 2 つを含む)	7

### 3.3. ストレージのプランニング



#### 注記

任意のドライバーまたはバックエンド種別のバックエンド **cinder** ボリュームを使用するゲストインスタンスで **LVM** を使用すると、パフォーマンスとボリュームの可視性/可用性で問題が生じます。このような問題は、**LVM** フィルターを使用すると緩和することができます。詳しくは、『**Storage Guide**』の「**Section 2.1 Back Ends**」および **KCS** の記事 **3213311** 「**Using LVM on a cinder volume exposes the data to the compute host**」を参照してください。

**director** は、オーバークラウド環境にさまざまなストレージオプションを提供します。オプションは以下のとおりです。

#### Ceph Storage ノード

**director** は、**Red Hat Ceph Storage** を使用して拡張可能なストレージノードセットを作成します。オーバークラウドは、各種ノードを以下の目的で使用します。

- **イメージ: Glance** は仮想マシンのイメージを管理します。イメージは変更できないため、

OpenStack はイメージバイナリープロブとして処理し、それに応じてイメージをダウンロードします。Ceph ブロックデバイスでイメージを格納するには、Glance を使用することができます。

- **ボリューム:** Cinder ボリュームはブロックデバイスです。OpenStack は、仮想マシンの起動や、実行中の仮想マシンへのボリュームのアタッチにボリュームを使用し、Cinder サービスを使用してボリュームを管理します。さらに、イメージの CoW (Copy-on-Write) のクローンを使用して仮想マシンを起動する際には Cinder を使用します。
- **ゲストディスク:** ゲストディスクは、ゲストオペレーティングシステムのディスクです。デフォルトでは、Nova で仮想マシンを起動すると、ディスクは、ハイパーバイザーのファイルシステム上のファイルとして表示されます (通常 `/var/lib/nova/instances/<uuid>/` の配下)。Cinder を使用せずに直接 Ceph 内にあ  
る全仮想マシンを起動することができます。これは、ライブマイグレーションのプロセスで簡単にメンテナンス操作を実行できるため好都合です。また、ハイパーバイザーが停止した場合には、`nova evacuate` をトリガーして仮想マシンをほぼシームレスに別の場所でも実行することもできるので便利です。



### 重要

Ceph で仮想マシンを起動するには (一時バックエンドまたはボリュームからの起動)、Glance のイメージ形式は **RAW** でなければなりません。Ceph は、仮想マシンディスクのホスティングで QCOW2 や VMDK などのその他の形式はサポートしていません。

その他の情報については、『[Red Hat Ceph Storage Architecture Guide](#)』を参照してください。

### Swift Storage ノード

`director` は、外部オブジェクトストレージノードを作成します。これは、オーバークラウド環境でコントローラーノードをスケーリングまたは置き換える必要があるが、高可用性クラスター外にオブジェクトストレージを保持する必要がある場合に便利です。

## 第4章 アンダークラウドのインストール

Red Hat OpenStack Platform 環境の構築では、最初にアンダークラウドシステムに **director** をインストールします。これには、必要なサブスクリプションやリポジトリを有効化するために複数の手順を実行する必要があります。

### 4.1. STACK ユーザーの作成

**director** のインストールプロセスでは、**root** 以外のユーザーがコマンドを実行する必要があります。以下のコマンドを使用して、**stack** という名前のユーザーを作成して、パスワードを設定します。

#### 手順

1. アンダークラウドに **root** ユーザーとしてログインします。
2. **stack** ユーザーを作成します。

```
[root@director ~]# useradd stack
```

3. そのユーザーのパスワードを設定します。

```
[root@director ~]# passwd stack
```

4. **sudo** を使用する場合にパスワードを要求されないようにします。

```
[root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a  
/etc/sudoers.d/stack  
[root@director ~]# chmod 0440 /etc/sudoers.d/stack
```

5. 新規作成した **stack** ユーザーに切り替えます。

```
[root@director ~]# su - stack  
[stack@director ~]$
```

**stack** ユーザーで **director** のインストールを続行します。

### 4.2. アンダークラウドのホスト名の設定

アンダークラウドでは、インストールと設定プロセスにおいて完全修飾ドメイン名が必要です。使用する DNS サーバーは、完全修飾ドメイン名を解決できる必要があります。たとえば、内部またはプライベートの DNS サーバーを使用することができます。これは、アンダークラウドのホスト名を設定する必要がある場合があることを意味します。

#### 手順

1. アンダークラウドのベースおよび完全なホスト名を確認します。

```
[stack@director ~]$ hostname  
[stack@director ~]$ hostname -f
```

2. 上記のコマンドのいずれかで正しいホスト名が出力されなかったり、エラーが表示される場合には、**hostnamectl** でホスト名を設定します。

```
[stack@director ~]$ sudo hostnamectl set-hostname
manager.example.com
[stack@director ~]$ sudo hostnamectl set-hostname --transient
manager.example.com
```

3. **director** では、**/etc/hosts** にシステムのホスト名とベース名も入力する必要があります。**/etc/hosts** の IP アドレスは、アンダークラウドのパブリック API に使用する予定のアドレスと一致する必要があります。たとえば、システムの名前が **manager.example.com** で、IP アドレスに **10.0.0.1** を使用する場合には、**/etc/hosts** に以下のように入力する必要があります。

```
10.0.0.1 manager.example.com manager
```

### 4.3. アンダークラウドの登録と更新

**director** をインストールする前に以下の準備作業を操作を行います。

- **Red Hat Subscription Manager** を使用してアンダークラウドを登録します。
- 関連するリポジトリをサブスクライブして有効化します。
- **Red Hat Enterprise Linux** パッケージの更新を実行します。

#### 手順

1. コンテンツ配信ネットワークにシステムを登録します。プロンプトが表示されたら、カスタマーポータルของผู้ーザー名とパスワードを入力します。

```
[stack@director ~]$ sudo subscription-manager register
```

2. **Red Hat OpenStack Platform director** のエンタイトルメントプール ID を検索します。以下に例を示します。

```
[stack@director ~]$ sudo subscription-manager list --available --all
--matches="Red Hat OpenStack"
Subscription Name:   Name of SKU
Provides:            Red Hat Single Sign-On
                    Red Hat Enterprise Linux Workstation
                    Red Hat CloudForms
                    Red Hat OpenStack
                    Red Hat Software Collections (for RHEL
Workstation)
                    Red Hat Virtualization
SKU:                 SKU-Number
Contract:           Contract-Number
Pool ID:             Valid-Pool-Number-123456
Provides Management: Yes
Available:          1
Suggested:          1
Service Level:      Support-level
Service Type:       Service-Type
Subscription Type:  Sub-type
Ends:                End-date
System Type:        Physical
```

- 3. **Pool ID** の値を特定して、Red Hat OpenStack Platform 13 のエンタイトルメントをアタッチします。

```
[stack@director ~]$ sudo subscription-manager attach --pool=Valid-Pool-Number-123456
```

- 4. デフォルトのリポジトリをすべて無効にしてから、必要な Red Hat Enterprise Linux リポジトリを有効にします。

```
[stack@director ~]$ sudo subscription-manager repos --disable=*
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-13-rpms
```

これらのリポジトリには、**director** のインストールに必要なパッケージが含まれます。



### 重要

「[リポジトリの要件](#)」でリストしたリポジトリのみを有効にします。追加のリポジトリを使用すると、パッケージとソフトウェアの競合が発生する場合があります。他のリポジトリは有効にしないでください。

- 5. システムで更新を実行して、ベースシステムパッケージを最新の状態にします。

```
[stack@director ~]$ sudo yum update -y
[stack@director ~]$ sudo reboot
```

システムは、**director** をインストールできる状態になりました。

## 4.4. DIRECTOR パッケージのインストール

以下の手順に従って、Red hat OpenStack Platform director に関連したパッケージをインストールします。

### 手順

- 1. **director** のインストールと設定を行うためのコマンドラインツールをインストールします。

```
[stack@director ~]$ sudo yum install -y python-tripleoclient
```

- 2. Ceph Storage ノードを使ってオーバークラウドを作成する場合は、さらに **ceph-ansible** パッケージをインストールします。

```
[stack@director ~]$ sudo yum install -y ceph-ansible
```

## 4.5. DIRECTOR の設定

**director** のインストールプロセスには、ネットワーク設定を判断する特定の設定が必要です。この設定は、**stack** ユーザーのホームディレクトリに **undercloud.conf** として配置されているテンプレートに保存されています。以下の手順では、デフォルトのテンプレートをベースに使用して設定を行う方

法についてを説明します。

## 手順

1. Red Hat は、インストールに必要な設定を判断しやすいように、基本テンプレートを提供しています。このテンプレートは、**stack** ユーザーのホームディレクトリーにコピーします。

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

2. **undercloud.conf** ファイルを編集します。このファイルには、アンダークラウドを設定するための設定値が含まれています。パラメーターを省略したり、コメントアウトした場合には、アンダークラウドのインストールでデフォルト値が使用されます。

## 4.6. DIRECTOR の設定パラメーター

**undercloud.conf** ファイルで設定するパラメーターの一覧を以下に示します。

### デフォルト

**undercloud.conf** ファイルの **[DEFAULT]** セクションで定義されているパラメーターを以下に示します。

#### **undercloud\_hostname**

アンダークラウドの完全修飾ホスト名を定義します。設定されている場合には、アンダークラウドのインストールで全システムのホスト名が設定されます。設定されていない場合には、アンダークラウドは現在のホスト名を使用しますが、ユーザーは適切に全システムのホスト名の設定を行う必要があります。

#### **local\_ip**

**director** のプロビジョニング NIC 用に定義する IP アドレス。これは、**director** が DHCP および PXE ブートサービスに使用する IP アドレスでもあります。環境内の既存の IP アドレスまたはサブネットと競合するなど、プロビジョニングネットワークに別のサブネットを使用する場合以外は、この値はデフォルトの **192.168.24.1/24** のままにします。

#### **undercloud\_public\_host**

SSL/TLS を使用する際に、**director** のパブリック API 用に定義する IP アドレス。これは、SSL/TLS で外部の **director** エンドポイントにアクセスするための IP アドレスです。**director** の設定により、この IP アドレスは **/32** ネットマスクを使用するルーティングされた IP アドレスとしてソフトウェアブリッジに接続されます。

#### **undercloud\_admin\_host**

SSL/TLS を使用する際に、**director** の管理 API 用に定義する IP アドレス。これは、SSL/TLS で管理エンドポイントにアクセスするための IP アドレスです。**director** の設定により、この IP アドレスは **/32** ネットマスクを使用するルーティングされた IP アドレスとしてソフトウェアブリッジに接続されます。

#### **undercloud\_nameservers**

アンダークラウドのホスト名解決に使用する DNS ネームサーバーの一覧

#### **undercloud\_ntp\_servers**

アンダークラウドの日付と時間を同期できるようにする Network Time Protocol サーバーの一覧

#### **overcloud\_domain\_name**

オーバークラウドのデプロイ時に使用する DNS ドメイン名



## 注記

オーバークラウドのパラメーター **CloudDomain** は対応する一致する値に設定する必要があります。

## サブネット

プロビジョニングおよびイントロスペクション用のルーティングネットワークのサブネットの一覧。詳しくは、「[サブネット](#)」を参照してください。デフォルト値に含まれるのは、**ctlplane-subnet** サブネットのみです。

### local\_subnet

PXE ブートと DHCP インターフェースに使用するローカルサブネット。**local\_ip** アドレスがこのサブネットに含まれている必要があります。デフォルトは **ctlplane-subnet** です。

### undercloud\_service\_certificate

OpenStack SSL/TLS 通信の証明書の場所とファイル名。理想的には、信頼できる認証局から、この証明書を取得します。それ以外の場合は、「[付録A SSL/TLS 証明書の設定](#)」のガイドラインを使用して独自の自己署名の証明書を作成します。これらのガイドラインには、自己署名の証明書が認証局からの証明書に拘らず、証明書の SELinux コンテキストを設定する方法が含まれています。

### generate\_service\_certificate

アンダークラウドのインストール時に SSL/TLS 証明書を生成するかを定義します。これは **undercloud\_service\_certificate** パラメーターに使用します。アンダークラウドのインストールで、作成された証明書 **/etc/pki/tls/certs/undercloud-[undercloud\_public\_vip].pem** を保存します。**certificate\_generation\_ca** パラメーターで定義される CA はこの証明書を署名します。

### certificate\_generation\_ca

要求した証明書を署名する CA の **certmonger** のニックネーム。**generate\_service\_certificate** パラメーターを設定した場合のみこのオプションを使用します。**local** CA を選択する場合は、**certmonger** はローカルの CA 証明書を **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** に抽出して、トラストチェーンに追加します。

### service\_principal

この証明書を使用するサービスの Kerberos プリンシパル。CA で FreeIPA などの Kerberos プリンシパルが必要な場合にのみ使用します。

### local\_interface

**director** のプロビジョニング NIC 用に選択するインターフェース。これは、**director** が DHCP および PXE ブートサービスに使用するデバイスでもあります。どのデバイスが接続されているかを確認するには、**ip addr** コマンドを使用します。以下に **ip addr** コマンドの出力結果の例を示します。

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 52:54:00:75:24:09 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.178/24 brd 192.168.122.255 scope global dynamic
eth0
    valid_lft 3462sec preferred_lft 3462sec
    inet6 fe80::5054:ff:fe75:2409/64 scope link
    valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noop state
DOWN
    link/ether 42:0b:c2:a5:c1:26 brd ff:ff:ff:ff:ff:ff
```

この例では、外部 NIC は `eth0` を、プロビジョニング NIC は未設定の `eth1` を使用します。今回は、`local_interface` を `eth1` に設定します。この設定スクリプトにより、このインターフェースが `inspection_interface` パラメーターで定義したカスタムのブリッジにアタッチされます。

### local\_mtu

`local_interface` に使用する MTU

### hieradata\_override

`hieradata` オーバーライドファイルへのパス。設定されている場合は、アンダークラウドのインストールでこのファイルが `/etc/puppet/hieradata` にコピーされ、この階層の最初のファイルとして設定されます。サービスに対して、`undercloud.conf` パラメーター以外に、サービスに対するカスタム設定を行うには、これを使用します。

### net\_config\_override

ネットワーク設定のオーバーライドテンプレートへのパス。これが設定されている場合にはアンダークラウドは JSON 形式のテンプレートを使用して `os-net-config` でネットワークを設定します。これは、`undercloud.conf` に設定されているネットワークパラメーターを無視します。`/usr/share/instack-undercloud/templates/net-config.json.template` の例を参照してください。

### inspection\_interface

ノードのイントロスペクションに `director` が使用するブリッジ。これは、`director` の設定により作成されるカスタムのブリッジです。`LOCAL_INTERFACE` でこのブリッジをアタッチします。これは、デフォルトの `br-ctlplane` のままにします。

### inspection\_iprange

`director` のイントロスペクションサービスが PXE ブートとプロビジョニングプロセスの際に使用する IP アドレス範囲。開始アドレスと終了アドレスの定義には、`192.168.24.100,192.168.24.120` などのように、コンマ区切りの値を使用します。この範囲には、使用するノードに十分な数の IP アドレスが含まれるようにし、`dhcp_start` と `dhcp_end` の範囲とは競合しないように設定してください。

### inspection\_extras

イントロスペクション時に追加のハードウェアコレクションを有効化するかどうかを定義します。イントロスペクションイメージでは `python-hardware` または `python-hardware-detect` パッケージが必要です。

### inspection\_runbench

ノードイントロスペクション時に一連のベンチマークを実行します。有効にするには、`true` に設定します。このオプションは、登録ノードのハードウェアを検査する際にベンチマーク分析を実行する場合に必要です。詳細は、「[ノードのハードウェアの検査](#)」を参照してください。

### inspection\_enable\_uefi

UEFI のみのファームウェアを使用するノードのイントロスペクションをサポートするかどうかを定義します。詳しくは、「[付録D 代替ブートモード](#)」を参照してください。

### enable\_node\_discovery

イントロスペクションの `ramdisk` を PXE ブートする不明なノードを自動的に登録します。新規ノードは、`fake_pxe` ドライバーをデフォルトとして使用しますが、`discovery_default_driver` を設定して上書きすることもできます。また、イントロスペクションルールを使用して、新しく登録したノードにドライバーの情報を指定することもできます。

### discovery\_default\_driver

自動的に登録されるノード用のデフォルトドライバーを設定します。`enable_node_discovery` を有効化して、`enabled_drivers` 一覧にそのドライバーを追加する必要があります。サポート対象のドライバー一覧は、「[付録B 電源管理ドライバー](#)」を参照してください。

### undercloud\_debug

アンダークラウドサービスのログレベルを **DEBUG** に設定します。この値は **true** に設定して有効化します。

### undercloud\_update\_packages

アンダークラウドのインストール時にパッケージを更新するかどうかを定義します。

### enable\_tempest

検証ツールをインストールするかどうかを定義します。デフォルトは、**false** に設定されていますが、**true** で有効化することができます。

### enable\_telemetry

アンダークラウドに OpenStack Telemetry サービス (ceilometer、aodh、panko、gnocchi) をインストールするかどうかを定義します。Red Hat OpenStack Platform では、Telemetry のメトリックバックエンドは gnocchi によって提供されます。**enable\_telemetry** パラメーターを **true** に設定すると、Telemetry サービスが自動的にインストール/設定されます。このノードで Telemetry サービスを使用しない場合には、この値は **false** に指定してください。

### enable\_ui

director の Web UI をインストールするかどうかを定義します。これにより、グラフィカル Web インターフェースを使用して、オーバークラウドのプランニングやデプロイメントが可能になります。詳しい情報は「[7章 WEB UI を使用した基本的なオーバークラウドの設定](#)」を参照してください。UI は、**undercloud\_service\_certificate** または **generate\_service\_certificate** のいずれかを使用して SSL/TLS を有効化している場合にのみ使用できる点にご注意ください。

### enable\_validations

検証の実行に必要なアイテムをインストールするかどうかを定義します。

### enable\_novajoin

アンダークラウドの novajoin メタデータサービスをインストールするかどうかを定義します。

### ipa\_otp

IPA サーバーにアンダークラウドノードを登録するためのワンタイムパスワードを定義します。これは、**enable\_novajoin** が有効な場合に必要です。

### ipxe\_enabled

iPXE か標準の PXE のいずれを使用するか定義します。デフォルトは **true** で iPXE を有効化します。**false** に指定すると、標準の PXE に設定されます。詳しい情報は、「[付録D 代替ブートモード](#)」を参照してください。

### scheduler\_max\_attempts

スケジューラーがインスタンスのデプロイを試行する最大回数。これは、スケジューリング時に競合状態にならないように、1度にデプロイする予定のベアメタルノードの数以上に指定するようにしてください。

### clean\_nodes

デプロイメントを再実行する前とイントロスペクションの後にハードドライブを消去するかどうかを定義します。

### enabled\_hardware\_types

アンダークラウドを有効にするためのハードウェア種別の一覧。サポートされているドライバーの一覧は、「[付録B 電源管理ドライバー](#)」を参照してください。

## パスワード

**undercloud.conf** ファイルの **[auth]** セクションで定義されているパラメーターを以下に示します。

## undercloud\_db\_password、undercloud\_admin\_token、undercloud\_admin\_password、undercloud\_glance\_passwmm など

残りのパラメーターは、全 **director** サービスのアクセス詳細を指定します。値を変更する必要はありません。**undercloud.conf** で空欄になっている場合には、これらの値は **director** の設定スクリプトによって自動的に生成されます。設定スクリプトの完了後には、すべての値を取得することができます。



### 重要

これらのパラメーターの設定ファイルの例では、プレースホルダーの値に **<None>** を使用しています。これらの値を **<None>** に設定すると、デプロイメントでエラーが発生します。

## サブネット

**undercloud.conf** ファイルには、各プロビジョニングサブネットの名前が付いたセクションがあります。たとえば、**ctlplane-subnet** という名前のサブネットを作成するとセクションは以下のようになります。

```
[ctlplane-subnet]
cidr = 192.168.24.0/24
dhcp_start = 192.168.24.5
dhcp_end = 192.168.24.24
inspection_iprange = 192.168.24.100,192.168.24.120
gateway = 192.168.24.1
masquerade = true
```

プロビジョニングネットワークは、環境に応じて、必要なだけ指定することができます。

## ゲートウェイ

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



### 注記

**director** の設定スクリプトは、適切な **sysctl** カーネルパラメーターを使用して IP フォワーディングを自動的に有効にする操作も行います。

## network\_cidr

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

## マスカレード

外部アクセス向けにマスカレードするネットワークを定義します。これにより、プロビジョニングネットワークにネットワークアドレス変換 (NAT) の範囲が提供され、**director** 経由で外部アクセスが可能になります。プロビジョニングネットワークに別のサブネットを使用しない限り、この値はデフォルト (**192.168.24.0/24**) のままにします。

## dhcp\_start; dhcp\_end

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

これらのパラメーターの値は、構成に応じて変更してください。完了したら、ファイルを保存します。

## 4.7. DIRECTOR のインストール

以下の手順では、**director** をインストールしてインストール後の基本的なタスクを実行します。

### 手順

1. 以下のコマンドを実行して、アンダークラウドに **director** をインストールします。

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

このコマンドで、**director** の設定スクリプトを起動します。**director** により、追加のパッケージがインストールされ、**undercloud.conf** の設定に合わせてサービスを設定します。このスクリプトは、完了までに数分かかります。

スクリプトにより、完了時には 2 つのファイルが生成されます。

- **undercloud-passwords.conf**: **director** サービスの全パスワード一覧
- **stackrc**: **director** のコマンドラインツールへアクセスできるようにする初期化変数セット

2. このスクリプトは、全 OpenStack Platform のサービスを自動的に起動します。以下のコマンドを使用して、有効化されたサービスを確認してください。

```
[stack@director ~]$ sudo systemctl list-units openstack-*
```

3. スクリプトにより、**docker** グループに **stack** ユーザーも追加され、その **stack** ユーザーはコンテナ管理コマンドにアクセスできるようになります。**stack** ユーザーのパーミッションを最新の状態に更新するには、以下のコマンドを実行します。

```
[stack@director ~]$ exec su -l stack
```

このコマンドでは再度ログインを要求されます。**stack** ユーザーのパスワードを入力します。

4. **stack** ユーザーを初期化してコマンドラインツールを使用するには、以下のコマンドを実行します。

```
[stack@director ~]$ source ~/stackrc
```

プロンプトには、OpenStack コマンドがアンダークラウドに対して認証および実行されることが表示されるようになります。

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

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

## 4.8. オーバークラウドノードのイメージの取得

**director** では、オーバークラウドのノードをプロビジョニングする際に、複数のディスクが必要です。必要なディスクは以下のとおりです。

- イントロスペクションのカーネルおよび **ramdisk**: PXE ブートでベアメタルシステムのイントロスペクションに使用
- デプロイメントカーネルおよび **ramdisk**: システムのプロビジョニングおよびデプロイメントに使用
- オーバークラウドカーネル、**ramdisk**、完全なイメージ: ノードのハードディスクに書き込まれるベースのオーバークラウドシステム

以下の手順は、これらのイメージの取得およびインストールの方法について説明します。

## 手順

1. **stackrc** ファイルを読み込んで、**director** のコマンドラインツールを有効にします。

```
[stack@director ~]$ source ~/stackrc
```

2. **rhosp-director-images** および **rhosp-director-images-ipa** パッケージをインストールします。

```
(undercloud) [stack@director ~]$ sudo yum install rhosp-director-images rhosp-director-images-ipa
```

3. **stack** ユーザーのホームの **images** ディレクトリー (**/home/stack/images**) にアーカイブを展開します。

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar; do tar -xvf $i; done
```

4. これらのイメージを **director** にインポートします。

```
(undercloud) [stack@director images]$ openstack overcloud image upload --image-path /home/stack/images/
```

このコマンドにより、以下のイメージが **director** にアップロードされます。

- **bm-deploy-kernel**
- **bm-deploy-ramdisk**
- **overcloud-full**
- **overcloud-full-initrd**
- **overcloud-full-vmlinuz**

これらは、デプロイメントおよびオーバークラウド用のイメージです。スクリプトにより、**director** の PXE サーバー上にイントロスペクションイメージもインストールされます。

5. これらのイメージが正常にアップロードされたことを確認します。

```
(undercloud) [stack@director images]$ openstack image list
+-----+-----+
| ID                | Name                |
+-----+-----+
| 765a46af-4417-4592-91e5-a300ead3faf6 | bm-deploy-ramdisk   |
| 09b40e3d-0382-4925-a356-3a4b4f36b514 | bm-deploy-kernel   |
| ef793cd0-e65c-456a-a675-63cd57610bd5 | overcloud-full     |
| 9a51a6cb-4670-40de-b64b-b70f4dd44152 | overcloud-full-initrd |
| 4f7e33f4-d617-47c1-b36f-cbe90f132e5d | overcloud-full-vmlinuz |
+-----+-----+
```

この一覧には、イントロスペクションの PXE イメージは表示されません。director は、これらのファイルを `/httpboot` にコピーします。

```
(undercloud) [stack@director images]$ ls -l /httpboot
total 341460
-rwxr-xr-x. 1 root          root          5153184 Mar 31
06:58 agent.kernel
-rw-r--r--. 1 root          root          344491465 Mar 31
06:59 agent.ramdisk
-rw-r--r--. 1 ironic-inspector ironic-inspector 337 Mar 31
06:23 inspector.ipxe
```



### 注記

デフォルトの `overcloud-full.qcow2` イメージは、フラットなパーティションイメージですが、ディスクイメージ全体をインポート、使用することも可能です。詳しい情報は、「[付録C 完全なディスクイメージ](#)」を参照してください。

## 4.9. コントロールプレーン用のネームサーバーの設定

オーバークラウドで `cdn.redhat.com` などの外部のホスト名を解決する予定の場合は、オーバークラウドノード上にネームサーバーを設定することを推奨します。ネットワークを分離していない標準のオーバークラウドの場合には、ネームサーバーはアンダークラウドのコントロールプレーンのサブネットを使用して定義されます。環境でネームサーバーを定義するには、以下の手順に従ってください。

### 手順

1. `stackrc` ファイルを読み込んで、director のコマンドラインツールを有効にします。

```
[stack@director ~]$ source ~/stackrc
```

2. `ctlplane-subnet` サブネット用のネームサーバーを設定します。

```
(undercloud) [stack@director images]$ openstack subnet set --dns-
nameserver [nameserver1-ip] --dns-nameserver [nameserver2-ip]
ctlplane-subnet
```

各ネームサーバーに `--dns-nameserver` オプションを使用します。

3. サブネットを表示してネームサーバーを確認します。

```
(undercloud) [stack@director images]$ openstack subnet show  
ctlplane-subnet
```

```
+-----+-----+  
---+  
| Field          | Value  
|  
+-----+-----+  
---+  
| ...            |  
| dns_nameservers | 8.8.8.8  
| ...            |  
|  
+-----+-----+  
---+
```



### 重要

サービストラフィックを別のネットワークに分離する場合は、オーバークラウドのノードはネットワーク環境ファイルの **DnsServer** パラメーターを使用します。

## 4.10. 次のステップ

これでアンダークラウドの設定が完了しました。次の章では、ノードの登録、検査、さまざまなノードロールのタグ付けなど、オーバークラウドの基本的な設定について説明します。

## 第5章 コンテナイメージのソースの設定

コンテナ化されたオーバークラウドには、必要なコンテナイメージを含むレジストリーへのアクセスが必要です。本章では、Red Hat OpenStack Platform 向けのコンテナイメージを使用するためのレジストリーおよびオーバークラウドの設定の準備方法について説明します。

- このガイドでは、オーバークラウドでレジストリーを使用するように設定するユースケースをいくつか記載しています。その方法についての説明は、「[レジストリーメソッド](#)」を参照してください。
- イメージを準備するコマンドの使用法をよく理解しておくことを推奨します。詳しくは、「[container image prepare コマンドの使用法](#)」を参照してください。
- コンテナイメージのソースを準備する最も一般的な方法で作業を開始するには、「[ローカルレジストリーとしてアンダークラウドを使用する方法](#)」を参照してください。

### 5.1. レジストリーメソッド

Red Hat OpenStack Platform は以下のレジストリータイプをサポートしています。

#### リモートレジストリー

オーバークラウドは、[registry.access.redhat.com](#) から直接コンテナイメージをプルします。これは、初期設定を生成するための最も簡単な方法ですが、それぞれのオーバークラウドノードが Red Hat Container Catalog から各イメージを直接プルするので、ネットワークの輻輳が生じてデプロイメントが遅くなる可能性があります。また、Red Hat Container Catalog にアクセスするためのインターネットアクセスが全オーバークラウドノードに必要です。

#### ローカルレジストリー

アンダークラウドにローカルレジストリーを作成し、[registry.access.redhat.com](#) からプルしたイメージを同期します。オーバークラウドは、アンダークラウドからコンテナイメージをプルします。この方法では、内部にレジストリーを保管することが可能なので、デプロイメントを迅速化してネットワークの輻輳を軽減することができます。ただし、アンダークラウドは基本的なレジストリーとしてのみ機能し、コンテナイメージのライフサイクル管理は限定されます。

#### Satellite サーバー

Red Hat Satellite 6 サーバーを介して、コンテナイメージの全アプリケーションライフサイクルを管理し、イメージを公開します。オーバークラウドは、Satellite サーバーからイメージをプルします。この方法は、Red Hat OpenStack Platform コンテナを保管、管理、デプロイするためのエンタープライズ級のソリューションを提供します。

上記のリストから方法を選択し、レジストリー情報の設定を続けます。

### 5.2. CONTAINER IMAGE PREPARE コマンドの使用法

本項では、`openstack overcloud container image prepare` コマンドの使用法について説明します。これには、このコマンドのさまざまなオプションについての概念的な情報も含まれます。

#### オーバークラウド用のコンテナイメージ環境ファイルの生成

`openstack overcloud container image prepare` コマンドの主要な用途の1つに、オーバークラウドが使用するイメージの一覧が記載されたファイルの作成があります。このファイルは、`openstack overcloud deploy` などのオーバークラウドのデプロイのコマンドで追加します。`openstack overcloud container image prepare` コマンドは、この機能に以下のオプションを使用します。

**--output-env-file**

作成される環境ファイルの名前を定義します。

以下のスニペットは、このファイルの内容の例を示しています。

```
parameter_defaults:
  DockerAodhApiImage: registry.access.redhat.com/rhosp13/openstack-aodh-
  api:latest
  DockerAodhConfigImage: registry.access.redhat.com/rhosp13/openstack-
  aodh-api:latest
  ...
```

**インポート方法に対応したコンテナイメージ一覧の生成**

OpenStack Platform コンテナイメージを異なるレジストリーソースにインポートする必要がある場合には、イメージの一覧を生成することができます。この一覧の構文は主に、アンダークラウド上のコンテナレジストリーにコンテナをインポートするのに使用されますが、Red Hat Satellite 6 などの別の方法に適した形式の一覧に変更することができます。

**openstack overcloud container image prepare** コマンドでは、この機能に以下のオプションを使用します。

**--output-images-file**

作成されるインポート一覧のファイル名を定義します。

このファイルの内容の例を以下に示します。

```
container_images:
- imagename: registry.access.redhat.com/rhosp13/openstack-aodh-api:latest
- imagename: registry.access.redhat.com/rhosp13/openstack-aodh-
  evaluator:latest
  ...
```

**コンテナイメージの名前空間の設定**

**--output-env-file** と **--output-images-file** のオプションには、作成されるイメージの場所を生成するための名前空間が必要です。**openstack overcloud container image prepare** コマンドでは、以下のオプションを使用して、プルするコンテナイメージの場所を設定します。

**--namespace**

コンテナイメージ用の名前空間を定義します。これには通常、ホスト名または IP アドレスにディレクトリーを付けて指定します。

**--prefix**

イメージ名の前に追加するプレフィックスを定義します。

その結果、**director** は以下のような形式のイメージ名を生成します。

- **[NAMESPACE]/[PREFIX][IMAGE NAME]**

**コンテナイメージタグの設定**

**openstack overcloud container image prepare** コマンドは、デフォルトでは各コンテナイメージに **latest** タグを使用しますが、以下のオプションのいずれか1つを使用すると、イメージバージョンに特定のタグを選択することができます。

**--tag-from-label**

指定したコンテナイメージラベルの値を使用して、全イメージのバージョンタグを検出します。

**--tag**

全イメージ用の特定のタグを設定します。すべての OpenStack Platform コンテナイメージで、同じタグを使用してバージョンの同期性を提供します。**--tag-from-label** と併用する場合には、バージョンタグはこのタグから最初に検出されます。

### 5.3. 追加のサービス用のコンテナイメージ

**director** は、OpenStack Platform のコアサービス用のコンテナイメージのみを作成します。一部の追加機能には、追加のコンテナイメージを必要とするサービスが使われます。これらのサービスは、環境ファイルで有効化することができます。**openstack overcloud container image prepare** コマンドでは、以下のオプションを使用して環境ファイルと対応するコンテナイメージを追加します。

**-e**

追加のコンテナイメージを有効化するための環境ファイルを指定します。

以下の表は、コンテナイメージを使用する追加のサービスのサンプル一覧とそれらの対応する環境ファイルがある `/usr/share/openstack-tripleo-heat-templates` ディレクトリー内の場所をまとめています。

サービス	環境ファイル
Ceph Storage	<code>environments/ceph-ansible/ceph-ansible.yaml</code>
Collectd	<code>environments/services-docker/collectd.yaml</code>
Congress	<code>environments/services-docker/congress.yaml</code>
Fluentd	<code>environments/services-docker/fluentd-client.yaml</code>
OpenStack Bare Metal (ironic)	<code>environments/services-docker/ironic.yaml</code>
OpenStack Data Processing (sahara)	<code>environments/services-docker/sahara.yaml</code>
OpenStack EC2-API	<code>environments/services-docker/ec2-api.yaml</code>
OpenStack Key Manager (barbican)	<code>environments/services-docker/barbican.yaml</code>
OpenStack Load Balancing-as-a-Service (octavia)	<code>environments/services-docker/octavia.yaml</code>
OpenStack Shared File System Storage (manila)	<code>environments/services-docker/manila.yaml</code>
Sensu	<code>environments/services-docker/sensu-client.yaml</code>

以下の項には、追加するサービスの例を記載します。

## Ceph Storage

Red Hat Ceph Storage クラスタをオーバークラウドでデプロイする場合には、`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml` 環境ファイルを追加する必要があります。このファイルは、オーバークラウドで、コンテナ化されたコンポーザブルサービスを有効化します。`director` は、これらのサービスが有効化されていることを確認した上で、それらのイメージを準備する必要があります。

この環境ファイルに加えて、**Ceph Storage** コンテナの場所を定義する必要があります。これは、**OpenStack Platform** サービスの場所とは異なります。`--set` オプションを使用して以下のパラメーターを **Ceph Storage** 固有に設定してください。

### `--set ceph_namespace`

**Ceph Storage** コンテナイメージ用の名前空間を定義します。これは、`--namespace` オプションと同様に機能します。

### `--set ceph_image`

**Ceph Storage** コンテナイメージの名前を定義します。通常は `rhceph-3-rhel7` という名前です。

### `--set ceph_tag`

**Ceph Storage** コンテナイメージに使用するタグを定義します。これは、`--tag` オプションと同じように機能します。`--tag-from-label` が指定されている場合には、バージョンタグはこのタグから検出が開始されます。

以下のスニペットは、コンテナイメージファイル内に **Ceph Storage** が含まれている例です。

```
$ openstack overcloud container image prepare \  
...  
-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-  
ansible/ceph-ansible.yaml \  
--set ceph_namespace=registry.access.redhat.com/rhceph \  
--set ceph_image=rhceph-3-rhel7 \  
--tag-from-label {version}-{release} \  
...
```

## OpenStack Bare Metal (ironic)

オーバークラウドで **OpenStack Bare Metal (ironic)** をデプロイする場合には、`/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml` 環境ファイルを追加して、`director` がイメージを準備できるようにする必要があります。以下のスニペットは、この環境ファイルの追加方法の例を示しています。

```
$ openstack overcloud container image prepare \  
...  
-e /usr/share/openstack-tripleo-heat-templates/environments/services-  
docker/ironic.yaml \  
...
```

## OpenStack Data Processing (sahara)

オーバークラウドで **OpenStack Data Processing (sahara)** をデプロイする場合には、`/usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml` 環境ファイルを追加して、`director` がイメージを準備できるようにする必要があります。

あります。以下のスニペットは、この環境ファイルの追加方法の例を示しています。

```
$ openstack overcloud container image prepare \  
...  
-e /usr/share/openstack-tripleo-heat-templates/environments/services-  
docker/sahara.yaml \  
...
```

## 5.4. RED HAT レジストリーをリモートレジストリーソースとして使用する 方法

Red Hat では、オーバークラウドのコンテナイメージを **registry.access.redhat.com** でホストしています。リモートレジストリーからイメージをプルする方法では、レジストリーはすでに設定済みで、必要なのはプルするイメージの URL と名前空間だけなので、最も簡単です。ただし、オーバークラウドの作成中には、オーバークラウドノードがリモートリポジトリからすべてのイメージをプルするので、外部接続で輻輳が生じる場合があります。これが問題となる場合には、以下のいずれかの手段を取ることができます。

- ローカルレジストリーの設定
- Red Hat Satellite 6 上でのイメージのホスティング

### 手順

1. イメージを直接 **registry.access.redhat.com** からオーバークラウドデプロイメントにプルするには、イメージパラメーターを指定するための環境ファイルが必要となります。以下のコマンドにより、この環境ファイルが自動的に作成されます。

```
(undercloud) $ openstack overcloud container image prepare \  
--namespace=registry.access.redhat.com/rhosp13 \  
--prefix=openstack- \  
--tag-from-label {version}-{release} \  
--output-env-file=/home/stack/templates/overcloud_images.yaml
```

- 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
  - **Ceph Storage** を使用している場合には、**Ceph Storage** 用のコンテナイメージの場所を定義する追加のパラメーターを指定します: **--set ceph\_namespace**、**--set ceph\_image**、**--set ceph\_tag**
2. これで、イメージの場所が記載された **overcloud\_images.yaml** 環境ファイルがアンダークラウド上に作成されます。このファイルをデプロイメントで指定します。

## 5.5. ローカルレジストリーとしてアンダークラウドを使用する方法

アンダークラウド上でローカルレジストリーを設定して、オーバークラウドのコンテナイメージを保管することができます。この方法は、以下の操作を伴います。

- **director** が、**registry.access.redhat.com** から各イメージをプルします。
- **director** がオーバークラウドを作成します。
- オーバークラウドの作成中に、ノードが適切なイメージをアンダークラウドからプルします。

これにより、コンテナイメージのネットワークトラフィックは、内部ネットワーク内に留まるので、外部ネットワークとの接続で輻輳が発生せず、デプロイメントプロセスを迅速化することができます。

## 手順

1. ローカルアンダークラウドレジストリーのアドレスを特定します。アドレスは、以下のパターンを使用します。

```
<REGISTRY IP ADDRESS>:8787
```

アンダークラウドのIPアドレスを使用します。これは **undercloud.conf** ファイルの **local\_ip** パラメーターで設定済みのアドレスです。以下のコマンドでは、アドレスが **192.168.24.1:8787** であることを前提としています。

2. イメージをローカルレジストリーにアップロードするためのテンプレートと、それらのイメージを参照する環境ファイルを作成します。

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp13 \
  --push-destination=192.168.24.1:8787 \
  --prefix=openstack- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml \
  --output-images-file /home/stack/local_registry_images.yaml
```

- 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
  - **Ceph Storage** を使用している場合には、**Ceph Storage** 用のコンテナイメージの場所を定義する追加のパラメーターを指定します: **--set ceph\_namespace**、**--set ceph\_image**、**--set ceph\_tag**
3. これで2つのファイルが作成されます。
    - リモートソースからのコンテナイメージの情報が含まれている **local\_registry\_images.yaml**。このファイルを使用して、Red Hat Container Registry (**registry.access.redhat.com**) からアンダークラウドにイメージをプルします。
    - アンダークラウド上の最終的なイメージの場所が記載されている **overcloud\_images.yaml**。このファイルはデプロイメントで指定します。両方のファイルが存在することを確認します。
  4. **registry.access.redhat.com** からアンダークラウドにコンテナイメージをプルします。

```
(undercloud) $ sudo openstack overcloud container image upload \
  --config-file /home/stack/local_registry_images.yaml \
  --verbose
```

ネットワークおよびアンダークラウドディスクの速度によっては、必要なイメージをプルするのに時間がかかる場合があります。



## 注記

コンテナイメージは、およそ 10 GB のディスク領域を使用します。

レジストリーの設定の準備が整いました。

## 5.6. SATELLITE サーバーをレジストリーとして使用する手順

Red Hat Satellite 6 には、レジストリーの同期機能が備わっています。これにより、複数のイメージを Satellite サーバーにプルし、アプリケーションライフサイクルの一環として管理することができます。また、他のコンテナ対応システムも Satellite をレジストリーとして使うことができます。コンテナイメージ管理の詳細は、『Red Hat Satellite 6 コンテンツ管理ガイド』の「[コンテナイメージの管理](#)」を参照してください。

以下の手順は、Red Hat Satellite 6 の **hammer** コマンドラインツールを使用した例を示しています。組織には、例として **ACME** という名称を使用しています。この組織は、実際に使用する Satellite 6 の組織に置き換えてください。

### 手順

1. イメージをローカルレジストリーにプルするためのテンプレートを作成します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud container image prepare \
  --namespace=rhosp13 \
  --prefix=openstack- \
  --output-images-file /home/stack/satellite_images \
```

- 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
- Ceph Storage を使用している場合には、Ceph Storage 用のコンテナイメージの場所を定義する追加のパラメーターを指定します: **--set ceph\_namespace**、**--set ceph\_image**、**--set ceph\_tag**



## 注記

このステップの **openstack overcloud container image prepare** コマンドは、**registry.access.redhat.com** 上のレジストリーをターゲットにして、イメージのリストを生成します。ここでは、後半のステップで使用する **openstack overcloud container image prepare** コマンドとは異なる値を指定します。

2. これで、コンテナイメージの情報が含まれた **satellite\_images** という名前のファイルが作成されます。このファイルを使用して、コンテナイメージを Satellite 6 サーバーに同期します。
3. **satellite\_images** ファイルから YAML 固有の情報を削除して、イメージ一覧のみが記載されたフラットファイルに変換します。この操作は、以下の **sed** コマンドで実行します。

```
(undercloud) $ awk -F ':' '{if (NR!=1) {gsub("[[:space:]]", "");  
print $2}}' ~/satellite_images > ~/satellite_images_names
```

これにより、Satellite サーバーにプルするイメージのリストが提供されます。

4. `satellite_images_names` ファイルを、Satellite 6 の `hammer` ツールが含まれるシステムにコピーします。あるいは、『[Hammer CLI ガイド](#)』に記載の手順に従って、`hammer` ツールをアンダークラウドにインストールします。
5. 以下の `hammer` コマンドを実行して、実際の Satellite 組織に新規製品 (**OSP13 Containers**) を作成します。

```
$ hammer product create \
  --organization "ACME" \
  --name "OSP13 Containers"
```

このカスタム製品に、イメージを保管します。

6. 製品にベースコンテナイメージを追加します。

```
$ hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.access.redhat.com \
  --docker-upstream-name rhosp13/openstack-base \
  --name base
```

7. `satellite_images` ファイルからオーバークラウドのコンテナイメージを追加します。

```
$ while read IMAGE; do \
  IMAGENAME=$(echo $IMAGE | cut -d"/" -f2 | sed "s/openstack-//g" |
  sed "s/:.*/g") ; \
  hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
  --url https://registry.access.redhat.com \
  --docker-upstream-name $IMAGE \
  --name $IMAGENAME ; done < satellite_images_names
```

8. コンテナイメージを同期します。

```
$ hammer product synchronize \
  --organization "ACME" \
  --name "OSP13 Containers"
```

Satellite サーバーが同期を完了するまで待ちます。



### 注記

設定によっては、`hammer` から Satellite サーバーのユーザー名およびパスワードが要求される場合があります。設定ファイルを使って自動的にログインするように `hammer` を設定することができます。詳細は、『[Hammer CLI ガイド](#)』の「[認証](#)」セクションを参照してください。

9. Satellite 6 サーバーでコンテンツビューを使用している場合には、新規コンテンツビューを作成して、イメージを取り入れます。

10. **base** イメージに利用可能なタグを確認します。

```
$ hammer docker tag list --repository "base" \
  --organization "ACME" \
  --product "OSP13 Containers"
```

これにより、OpenStack Platform コンテナイメージのタグが表示されます。

11. アンダークラウドに戻り、Satellite サーバー上のイメージ用に環境ファイルを生成します。環境ファイルを生成するコマンドの例を以下に示します。

```
(undercloud) $ openstack overcloud container image prepare \
  --namespace=satellite6.example.com:5000 \
  --prefix=acme-osp13_containers- \
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```



### 注記

このステップの **openstack overcloud container image prepare** コマンドは、Satellite サーバーをターゲットにします。ここでは、前のステップで使用した **openstack overcloud container image prepare** コマンドとは異なる値を指定します。

このコマンドを実行する際には、以下の情報を含めてください。

- **--namespace:** Satellite サーバー上のレジストリーの URL およびポート。Red Hat Satellite のデフォルトのレジストリーポートは 5000 です。例: **--namespace=satellite6.example.com:5000**
- **--prefix=:** プレフィックスは Satellite 6 の命名規則に基づきます。これは、コンテンツビューを使用するかどうかによって異なります。
  - コンテンツビューを使用する場合、構成は **[org]-[environment]-[content view]-[product]-** となります (例: **acme-production-myosp13-osp13\_containers-**)。
  - コンテンツビューを使用する場合、構成は **[org]-[product]-** となります (例: **acme-osp13\_containers-**)。
- **--tag-from-label {version}-{release}:** 各イメージの最新のタグを特定します。
- **-e:** オプションのサービスの環境ファイルを指定します。
- **--set ceph\_namespace、--set ceph\_image、--set ceph\_tag:** Ceph Storage を使用する場合には、Ceph Storage のコンテナイメージの場所を定義する追加のパラメーターを指定します。**ceph\_image** に Satellite 固有のプレフィックスが追加された点に注意してください。このプレフィックスは、**--prefix** オプションと同じ値です。以下に例を示します。

```
--set ceph_image=acme-osp13_containers-rhceph-3-rhel7
```

これにより、オーバークラウドは Satellite の命名規則の Ceph コンテナイメージを使用することができます。

- これで、Satellite サーバー上のイメージの場所が記載された `overcloud_images.yaml` 環境ファイルが作成されます。このファイルをデプロイメントで指定します。

レジストリーの設定の準備が整いました。

## 5.7. 次のステップ

コンテナイメージのソースが記載された `overcloud_images.yaml` 環境ファイルができました。今後のデプロイメント操作ではすべてこのファイルを追加してください。

## 第6章 CLI ツールを使用した基本的なオーバークラウドの設定

本章では、CLI ツールを使用した OpenStack Platform 環境の基本的な設定手順を説明します。基本設定のオーバークラウドには、カスタムの機能は含まれていませんが、『[Advanced Overcloud Customization](#)』ガイドに記載の手順に従って、この基本的なオーバークラウドに高度な設定オプションを追加して、仕様に合わせてカスタマイズすることができます。

本章の例では、すべてのノードが電源管理に IPMI を使用したベアメタルシステムとなっています。電源管理の種別およびそのオプションに関する詳細は、『[付録B 電源管理ドライバ](#)』を参照してください。

### ワークフロー

1. ノード定義のテンプレートを作成して `director` で空のノードを登録します。
2. 全ノードのハードウェアを検査します。
3. ロールにノードをタグ付けします。
4. 追加のノードプロパティを定義します。

### 要件

- [4章 アンダークラウドのインストール](#)で作成された `director` ノード
- ノードに使用するベアメタルマシンのセット。必要なノード数は、作成予定のオーバークラウドのタイプにより異なります(オーバークラウドロールに関する情報は『[ノードのデプロイメントロールのプランニング](#)』を参照してください)。これらのマシンは、各ノード種別の要件セットに従う必要があります。これらの要件については、『[オーバークラウドの要件](#)』を参照してください。これらのノードにはオペレーティングシステムは必要ありません。`director` が Red Hat Enterprise Linux 7 のイメージを各ノードにコピーします。
- ネイティブ VLAN として設定したプロビジョニングネットワーク用のネットワーク接続1つ。全ノードは、このネイティブに接続して、『[ネットワーク要件](#)』で設定した要件に準拠する必要があります。この章の例では、以下の IP アドレスの割り当てで、プロビジョニングサブネットとして `192.168.24.0/24` を使用します。

表6.1 プロビジョニングネットワークの IP 割り当て

ノード名	IP アドレス	MAC アドレス	IPMI IP アドレス
director	192.168.24.1	aa:aa:aa:aa:aa:aa	不要
コントローラー	定義済みの DHCP	bb:bb:bb:bb:bb:bb	192.168.24.205
Compute	定義済みの DHCP	cc:cc:cc:cc:cc:cc	192.168.24.206

- その他のネットワーク種別はすべて、OpenStack サービスにプロビジョニングネットワークを使用しますが、ネットワークトラフィックの他のタイプに追加でネットワークを作成することができます。
- コンテナイメージのソース。コンテナイメージのソースを含む環境ファイルの生成方法については、『[5章 コンテナイメージのソースの設定](#)』を参照してください。

## 6.1. オーバークラウドへのノードの登録

`director` では、手動で作成したノード定義のテンプレートが必要です。このファイル (`instackenv.json`) は、JSON 形式のファイルを使用して、ノードのハードウェアおよび電源管理の情報が含まれます。たとえば、2つのノードを登録するテンプレートは、以下のようになります。

```
{
  "nodes": [
    {
      "mac": [
        "bb:bb:bb:bb:bb:bb"
      ],
      "name": "node01",
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "pxe_ipmitool",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.168.24.205"
    },
    {
      "mac": [
        "cc:cc:cc:cc:cc:cc"
      ],
      "name": "node02",
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "pxe_ipmitool",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.168.24.206"
    }
  ]
}
```

このテンプレートでは、以下の属性を使用します。

### **name**

ノードの論理名

### **pm\_type**

使用する電源管理ドライバー。この例では IPMI ドライバーを使用します (`pxe_ipmitool`)。

### **pm\_user; pm\_password**

IPMI のユーザー名およびパスワード

### **pm\_addr**

IPMI デバイスの IP アドレス

### **mac**

(オプション) ノード上のネットワークインターフェースの MAC アドレス一覧。各システムのプロビジョニング NIC の MAC アドレスのみを使用します。

**cpu**

(オプション) ノード上の CPU 数

**memory**

(オプション) メモリーサイズ (MB)

**disk**

(オプション) ハードディスクのサイズ (GB)

**arch**

(オプション) システムアーキテクチャー

**注記**

電源管理の種別およびそのオプションに関する詳細は、「[付録B 電源管理ドライバー](#)」を参照してください。

テンプレートの作成後に、**stack** ユーザーのホームディレクトリー (`/home/stack/instackenv.json`) にファイルを保存して、以下のコマンドを使用して **director** にインポートします。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud node import ~/instackenv.json
```

このコマンドでテンプレートをインポートして、テンプレートから **director** に各ノードを登録します。

ノードを登録して設定が完了した後に、CLI でこれらのノードの一覧を表示します。

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

## 6.2. ノードのハードウェアの検査

**director** は各ノードでイントロスペクションプロセスを実行することができます。このプロセスを実行すると、各ノードが PXE を介してイントロスペクションエージェントを起動します。このエージェントは、ノードからハードウェアのデータを収集して、**director** に送り返します。次に **director** は、**director** 上で実行中の **OpenStack Object Storage (swift)** サービスにこのイントロスペクションデータを保管します。**director** は、プロファイルのタグ付け、ベンチマーキング、ルートディスクの手動割り当てなど、さまざまな目的でハードウェア情報を使用します。

**注記**

ポリシーファイルを作成して、イントロスペクションの直後にノードをプロファイルに自動でタグ付けすることも可能です。ポリシーファイルを作成してイントロスペクションプロセスに組み入れる方法に関する詳しい情報は、「[付録E プロファイルの自動タグ付け](#)」を参照してください。または、「[プロファイルへのノードのタグ付け](#)」に記載の手順に従って、ノードをプロファイルに手動でタグ付けすることもできます。

以下のコマンドを実行して、各ノードのハードウェア属性を検証します。

```
(undercloud) $ openstack overcloud node introspect --all-manageable --provide
```

- **--all-manageable** オプションは、管理状態のノードのみをイントロスペクションします。上記の例では、すべてのノードが対象です。
- **--provide** オプションは、イントロスペクション後に全ノードを **available** の状態にします。

別のターミナルウィンドウで以下のコマンドを使用してイントロスペクションの進捗状況をモニタリングします。

```
(undercloud) $ sudo journalctl -l -u openstack-ironic-inspector -u
openstack-ironic-inspector-dnsmasq -u openstack-ironic-conductor -f
```



### 重要

このプロセスが最後まで実行されて正常に終了したことを確認してください。ベアメタルの場合には、通常 15 分ほどかかります。

イントロスペクション完了後には、すべてのノードが **available** の状態に変わります。

### ノードイントロスペクションの個別実行

**available** の状態のノードで個別にイントロスペクションを実行するには、ノードを管理モードに設定して、イントロスペクションを実行します。

```
(undercloud) $ openstack baremetal node manage [NODE UUID]
(undercloud) $ openstack overcloud node introspect [NODE UUID] --provide
```

イントロスペクションが完了すると、ノードは **available** の状態に変わります。

### 初回のイントロスペクション後のノードイントロスペクションの実行

**--provide** オプションを指定したので、初回のイントロスペクションの後には、全ノードが **available** の状態に入るはずですが、初回のイントロスペクション後に全ノードにイントロスペクションを実行するには、すべてのノードを **manageable** の状態にして、一括のイントロスペクションコマンドを実行します。

```
(undercloud) $ for node in $(openstack baremetal node list --fields uuid -
f value) ; do openstack baremetal node manage $node ; done
(undercloud) $ openstack overcloud node introspect --all-manageable --
provide
```

イントロスペクション完了後には、すべてのノードが **available** の状態に変わります。

### ネットワークイントロスペクションの実行によるインターフェース情報の取得

ネットワークイントロスペクションにより、Link Layer Discovery Protocol (LLDP) データがネットワークスイッチから取得されます。以下のコマンドにより、ノード上の全インターフェースに関する LLDP 情報のサブセット、または特定のノードおよびインターフェースに関するすべての情報が表示されます。この情報は、トラブルシューティングに役立ちます。director では、デフォルトで LLDP データ収集が有効になっています。

ノード上のインターフェースのリストを取得するには、以下のコマンドを実行します。

```
(undercloud) $ openstack baremetal introspection interface list [NODE
UUID]
```

例:

```
(undercloud) $ openstack baremetal introspection interface list c89397b7-
a326-41a0-907d-79f8b86c7cd9
+-----+-----+-----+-----+
| Interface | MAC Address          | Switch Port VLAN IDs | Switch Chassis
ID | Switch Port ID |
+-----+-----+-----+-----+
| p2p2      | 00:0a:f7:79:93:19   | [103, 102, 18, 20, 42] |
64:64:9b:31:12:00 | 510                 |
| p2p1      | 00:0a:f7:79:93:18   | [101]                   |
64:64:9b:31:12:00 | 507                 |
| em1       | c8:1f:66:c7:e8:2f   | [162]                   |
08:81:f4:a6:b3:80 | 515                 |
| em2       | c8:1f:66:c7:e8:30   | [182, 183]              |
08:81:f4:a6:b3:80 | 559                 |
+-----+-----+-----+-----+
|-----+-----+-----+-----+
|
```

インターフェースのデータおよびスイッチポートの情報を表示するには、以下のコマンドを実行します。

```
(undercloud) $ openstack baremetal introspection interface show [NODE
UUID] [INTERFACE]
```

例:

```
(undercloud) $ openstack baremetal introspection interface show c89397b7-
a326-41a0-907d-79f8b86c7cd9 p2p1
+-----+-----+-----+-----+
|-----+-----+-----+-----+
| Field                | Value
|
+-----+-----+-----+-----+
|-----+-----+-----+-----+
| interface            | p2p1
|
| mac                  | 00:0a:f7:79:93:18
|
| node_ident           | c89397b7-a326-41a0-907d-
79f8b86c7cd9
|
| switch_capabilities_enabled | [u'Bridge', u'Router']
|
| switch_capabilities_support | [u'Bridge', u'Router']
|
| switch_chassis_id    | 64:64:9b:31:12:00
|
| switch_port_autonegotiation_enabled | True
|
```

```

| switch_port_autonegotiation_support | True
|
| switch_port_description             | ge-0/0/2.0
|
| switch_port_id                     | 507
|
| switch_port_link_aggregation_enabled | False
|
| switch_port_link_aggregation_id     | 0
|
| switch_port_link_aggregation_support | True
|
| switch_port_management_vlan_id      | None
|
| switch_port_mau_type                | Unknown
|
| switch_port_mtu                    | 1514
|
| switch_port_physical_capabilities   | [u'1000BASE-T fdx', u'100BASE-TX
fdx', u'100BASE-TX hdx', u'10BASE-T fdx', u'10BASE-T hdx', u'Asym and Sym
PAUSE fdx'] |
| switch_port_protocol_vlan_enabled   | None
|
| switch_port_protocol_vlan_ids       | None
|
| switch_port_protocol_vlan_support   | None
|
| switch_port_untagged_vlan_id        | 101
|
| switch_port_vlan_ids                | [101]
|
| switch_port_vlans                   | [{u'name': u'RHOS13-PXE', u'id':
101}]
|
| switch_protocol_identities          | None
|
| switch_system_name                  | rhos-compute-node-sw1
|
+-----+-----+
-----+-----+
-----+-----+

```

## ハードウェアイントロスペクション情報の取得

Bare Metal サービスでは、ハードウェア検査時に追加のハードウェア情報を取得するためのパラメーター (**inspection\_extras**) がデフォルトで有効になっています。これらのハードウェア情報を使って、オーバークラウドを設定することができます。**undercloud.conf** ファイルの **inspection\_extras** パラメーターに関する詳細は、「[director の設定](#)」を参照してください。

たとえば、**numa\_topology** コレクターは、このハードウェア **inspection\_extras** の一部で、各 NUMA ノードに関する以下の情報が含まれます。

- RAM (キロバイト単位)
- 物理 CPU コアおよびそのシブリングスレッド
- NUMA ノードに関連付けられた NIC

この情報を取得するには、ベアメタルノードの **UUID** を指定して、**openstack baremetal introspection data save \_UUID\_ | jq .numa\_topology** コマンドを実行します。

取得されるベアメタルノードの NUMA 情報の例を、以下に示します。

```
{
  "cpus": [
    {
      "cpu": 1,
      "thread_siblings": [
        1,
        17
      ],
      "numa_node": 0
    },
    {
      "cpu": 2,
      "thread_siblings": [
        10,
        26
      ],
      "numa_node": 1
    },
    {
      "cpu": 0,
      "thread_siblings": [
        0,
        16
      ],
      "numa_node": 0
    },
    {
      "cpu": 5,
      "thread_siblings": [
        13,
        29
      ],
      "numa_node": 1
    },
    {
      "cpu": 7,
      "thread_siblings": [
        15,
        31
      ],
      "numa_node": 1
    },
    {
      "cpu": 7,
      "thread_siblings": [
        7,
        23
      ],
      "numa_node": 0
    },
    {
```

```
"cpu": 1,
"thread_siblings": [
  9,
  25
],
"numa_node": 1
},
{
"cpu": 6,
"thread_siblings": [
  6,
  22
],
"numa_node": 0
},
{
"cpu": 3,
"thread_siblings": [
  11,
  27
],
"numa_node": 1
},
{
"cpu": 5,
"thread_siblings": [
  5,
  21
],
"numa_node": 0
},
{
"cpu": 4,
"thread_siblings": [
  12,
  28
],
"numa_node": 1
},
{
"cpu": 4,
"thread_siblings": [
  4,
  20
],
"numa_node": 0
},
{
"cpu": 0,
"thread_siblings": [
  8,
  24
],
"numa_node": 1
},
{
```

```
    "cpu": 6,
    "thread_siblings": [
      14,
      30
    ],
    "numa_node": 1
  },
  {
    "cpu": 3,
    "thread_siblings": [
      3,
      19
    ],
    "numa_node": 0
  },
  {
    "cpu": 2,
    "thread_siblings": [
      2,
      18
    ],
    "numa_node": 0
  }
],
"ram": [
  {
    "size_kb": 66980172,
    "numa_node": 0
  },
  {
    "size_kb": 67108864,
    "numa_node": 1
  }
],
"nics": [
  {
    "name": "ens3f1",
    "numa_node": 1
  },
  {
    "name": "ens3f0",
    "numa_node": 1
  },
  {
    "name": "ens2f0",
    "numa_node": 0
  },
  {
    "name": "ens2f1",
    "numa_node": 0
  },
  {
    "name": "ens1f1",
    "numa_node": 0
  },
  {

```

```

    "name": "ens1f0",
    "numa_node": 0
  },
  {
    "name": "eno4",
    "numa_node": 0
  },
  {
    "name": "eno1",
    "numa_node": 0
  },
  {
    "name": "eno3",
    "numa_node": 0
  },
  {
    "name": "eno2",
    "numa_node": 0
  }
]
}

```

### 6.3. ベアメタルノードの自動検出

**auto-discovery** を使用すると、最初に **instackenv.json** ファイルを作成せずに、アンダークラウドノードを登録してそのメタデータを生成することができます。この改善により、最初のノード情報取得に費す時間を短縮できます。たとえば、IPMI IP アドレスを照合し、その後に **instackenv.json** ファイルを作成する必要がなくなります。

#### 要件

- すべてのオーバークラウドノードの BMC が、IPMI を通じて **director** にアクセスできるように設定されていること
- すべてのオーバークラウドノードが、アンダークラウドのコントロールプレーンネットワークに接続された NIC から PXE ブートするように設定されていること

#### 自動検出の有効化

1. **undercloud.conf** でベアメタルの自動検出を有効にします。

```

enable_node_discovery = True
discovery_default_driver = pxe_ipmitool

```

- **enable\_node\_discovery**: 有効にすると、PXE を使ってイントロスペクション **ramdisk** をブートするすべてのノードが **Ironic** に登録されます。
  - **discovery\_default\_driver**: 検出されたノードに使用するドライバーを設定します。  
例: **pxe\_ipmitool**
2. IPMI の認証情報を **Ironic** に追加します。
    - a. IPMI の認証情報を **ipmi-credentials.json** という名前のファイルに追加します。以下の例で使用しているユーザー名とパスワードの値は、お使いの環境に応じて置き換える必要があります。

```
[
  {
    "description": "Set default IPMI credentials",
    "conditions": [
      {"op": "eq", "field": "data://auto_discovered",
"value": true}
    ],
    "actions": [
      {"action": "set-attribute", "path":
"driver_info/ipmi_username",
"value": "SampleUsername"},
      {"action": "set-attribute", "path":
"driver_info/ipmi_password",
"value": "RedactedSecurePassword"},
      {"action": "set-attribute", "path":
"driver_info/ipmi_address",
"value": "{data[inventory][bmc_address]}"}
    ]
  }
]
```

3. IPMI の認証情報ファイルを **Ironic** にインポートします。

```
$ openstack baremetal introspection rule import ipmi-
credentials.json
```

### 自動検出のテスト

1. 必要なノードの電源をオンにします。
2. **openstack baremetal node list** を実行します。新しいノードが **enroll** の状態でリストに表示されるはずですが。

```
$ openstack baremetal node list
+-----+-----+-----+-----+
| UUID                                | Name | Instance UUID |
Power State | Provisioning State | Maintenance |
+-----+-----+-----+-----+
| c6e63aec-e5ba-4d63-8d37-bd57628258e8 | None | None          |
power off   | enroll              | False        |
| 0362b7b2-5b9c-4113-92e1-0b34a2535d9b | None | None          |
power off   | enroll              | False        |
+-----+-----+-----+-----+
```

3. 各ノードにリソースクラスを設定します。

```
$ for NODE in `openstack baremetal node list -c UUID -f value` ; do
openstack baremetal node set $NODE --resource-class baremetal ; done
```

4. 各ノードにカーネルと **ramdisk** を設定します。

```
$ for NODE in `openstack baremetal node list -c UUID -f value` ; do
openstack baremetal node manage $NODE ; done
$ openstack overcloud node configure --all-manageable
```

5. 全ノードを利用可能な状態に設定します。

```
$ for NODE in `openstack baremetal node list -c UUID -f value` ; do
openstack baremetal node provide $NODE ; done
```

## ルールを使用して異なるベンダーのハードウェアを検出する方法

異種のハードウェアが混在する環境では、イントロスペクションルールを使って、認証情報の割り当てやリモート管理を行うことができます。たとえば、DRAC を使用する Dell ノードを処理するには、別の検出ルールが必要になる場合があります。

1. 以下の内容で、**dell-drac-rules.json** という名前のファイルを作成します。この例で使用しているユーザー名およびパスワードの値は、お使いの環境に応じて置き換える必要があります。

```
[
  {
    "description": "Set default IPMI credentials",
    "conditions": [
      {"op": "eq", "field": "data://auto_discovered", "value":
true},
      {"op": "ne", "field":
"data://inventory.system_vendor.manufacturer",
"value": "Dell Inc."}
    ],
    "actions": [
      {"action": "set-attribute", "path":
"driver_info/ipmi_username",
"value": "SampleUsername"},
      {"action": "set-attribute", "path":
"driver_info/ipmi_password",
"value": "RedactedSecurePassword"},
      {"action": "set-attribute", "path":
"driver_info/ipmi_address",
"value": "{data[inventory][bmc_address]}"}
    ]
  },
  {
    "description": "Set the vendor driver for Dell hardware",
    "conditions": [
      {"op": "eq", "field": "data://auto_discovered", "value":
true},
      {"op": "eq", "field":
"data://inventory.system_vendor.manufacturer",
"value": "Dell Inc."}
    ],
    "actions": [
      {"action": "set-attribute", "path": "driver", "value":
"idrac"},
      {"action": "set-attribute", "path":
"driver_info/drac_username",
```

```

        "value": "SampleUsername"},
        {"action": "set-attribute", "path":
"driver_info/drac_password",
        "value": "RedactedSecurePassword"},
        {"action": "set-attribute", "path":
"driver_info/drac_address",
        "value": "{data[inventory][bmc_address]}"}
    ]
}
]

```

2. ルールを **Ironic** にインポートします。

```
$ openstack baremetal introspection rule import dell-drac-rules.json
```

## 6.4. プロファイルへのノードのタグ付け

各ノードのハードウェアを登録、検査した後は、特定のプロファイルにノードをタグ付けします。このプロファイルタグにより、ノードがフレーバーに照合され、次にそのフレーバーがデプロイメントロールに割り当てられます。以下の例では、コントローラーノードのロール、フレーバー、プロファイル、ノード間の関係を示しています。

タイプ	説明
ロール	<b>Controller</b> ロールはコントローラーノードの設定方法を定義します。
フレーバー	<b>control</b> フレーバーは、コントローラーとして使用するためにノードのハードウェアプロファイルを定義します。使用するノードを <b>director</b> が決定できるように、このフレーバーを <b>Controller</b> ロールに割り当てます。
プロファイル	<b>control</b> プロファイルは、 <b>control</b> フレーバーに適用するタグで、このフレーバーに所属するノードを定義します。
ノード	また、各ノードに <b>control</b> プロファイルタグを適用して、 <b>control</b> フレーバーにグループ化します。これにより、 <b>director</b> が <b>Controller</b> ロールを使用してノードを設定します。

アンダークラウドのインストール時に、デフォルトプロファイルのフレーバー **compute**、**control**、**swift-storage**、**ceph-storage**、**block-storage** が作成され、大半の環境で変更なしに使用することができます。



### 注記

多くのノードでは、プロファイルの自動タグ付けを使用します。詳しい情報は、「[付録E プロファイルの自動タグ付け](#)」を参照してください。

特定のプロファイルにノードをタグ付けする場合には、各ノードの **properties/capabilities** パラメーターに **profile** オプションを追加します。たとえば、2つのノードをタグ付けしてコントローラプロファイルとコンピュープロファイルをそれぞれ使用するには、以下のコマンドを実行します。

```
(undercloud) $ openstack baremetal node set --property
capabilities='profile:compute,boot_option:local' 58c3d07e-24f2-48a7-bbb6-
6843f0e8ee13
(undercloud) $ openstack baremetal node set --property
capabilities='profile:control,boot_option:local' 1a4e30da-b6dc-499d-ba87-
0bd8a3819bc0
```

**profile:compute** と **profile:control** オプションを追加することで、この2つのノードがそれぞれのプロファイルにタグ付けされます。

これらのコマンドは、各ノードのブート方法を定義する **boot\_option:local** パラメーターも設定します。お使いのハードウェアによっては、**boot\_mode** パラメーターを **uefi** に追加して、ノードが BIOS モードの代わりに UEFI を使用してブートするようになる必要がある場合もあります。詳しい情報は、「[UEFI ブートモード](#)」を参照してください。

ノードのタグ付けが完了した後は、割り当てたプロファイルまたはプロファイルの候補を確認します。

```
(undercloud) $ openstack overcloud profiles list
```

### カスタムロールのプロファイル

カスタムロールを使用する場合には、これらの新規ロールに対応するために追加のフレーバーやプロファイルを作成する必要があるかもしれません。たとえば、**Networker** ロールの新規フレーバーを作成するには、以下のコマンドを実行します。

```
(undercloud) $ openstack flavor create --id auto --ram 4096 --disk 40 --
vcpus 1 networker
(undercloud) $ openstack flavor set --property "cpu_arch"="x86_64" --
property "capabilities:boot_option"="local" --property
"capabilities:profile"="networker" networker
```

この新規プロファイルにノードを割り当てます。

```
(undercloud) $ openstack baremetal node set --property
capabilities='profile:networker,boot_option:local' dad05b82-0c74-40bf-
9d12-193184bfc72d
```

## 6.5. ノードのルートディスクの定義

一部のノードは、複数のディスクを使用する場合があります。このため、**director** はプロビジョニング中に、ルートディスクに使用するディスクを特定する必要があります。

**director** がルートディスクを容易に特定できるようにするには、以下のようなプロパティを使用することができます。

- **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)
- **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** (文字列): デバイス名 (例: /dev/sdb1)



### 重要

**name** は、永続デバイス名が付いたデバイスのみを使用します。**name** で他のデバイスのルートディスクを設定しないでください。この値は、ノードのブート時に変更される可能性があります。

以下の例では、**root** デバイスを特定するディスクのシリアル番号を使用して、オーバークラウドイメージをデプロイするドライブを指定します。

各ノードのハードウェアイントロスペクションからのディスク情報を確認します。以下のコマンドは、ノードからのディスク情報を表示します。

```
(undercloud) $ openstack baremetal introspection data save 1a4e30da-b6dc-499d-ba87-0bd8a3819bc0 | jq ".inventory.disks"
```

たとえば、1つのノードのデータで3つのディスクが表示される場合があります。

```
[
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sda",
    "wwn_vendor_extension": "0x1ea4dcc412a9632b",
    "wwn_with_extension": "0x61866da04f3807001ea4dcc412a9632b",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380700",
    "serial": "61866da04f3807001ea4dcc412a9632b"
  }
  {
    "size": 299439751168,
    "rotational": true,
    "vendor": "DELL",
    "name": "/dev/sdb",
    "wwn_vendor_extension": "0x1ea4e13c12e36ad6",
    "wwn_with_extension": "0x61866da04f380d001ea4e13c12e36ad6",
    "model": "PERC H330 Mini",
    "wwn": "0x61866da04f380d00",
    "serial": "61866da04f380d001ea4e13c12e36ad6"
  }
]
```

```

}
{
  "size": 299439751168,
  "rotational": true,
  "vendor": "DELL",
  "name": "/dev/sdc",
  "wwn_vendor_extension": "0x1ea4e31e121cfb45",
  "wwn_with_extension": "0x61866da04f37fc001ea4e31e121cfb45",
  "model": "PERC H330 Mini",
  "wwn": "0x61866da04f37fc00",
  "serial": "61866da04f37fc001ea4e31e121cfb45"
}
]

```

以下の例では、ルートデバイスを、シリアル番号 **61866da04f380d001ea4e13c12e36ad6** の **disk 2** に設定します。そのためには、ノードの定義に **root\_device** パラメーターを追加する必要があります。

```

(undercloud) $ openstack baremetal node set --property
root_device='{"serial": "61866da04f380d001ea4e13c12e36ad6"}' 1a4e30da-
b6dc-499d-ba87-0bd8a3819bc0

```



### 注記

各ノードの BIOS を設定して、選択したルートディスクからの起動が含まれるようにします。推奨のブート順は最初がネットワークブートで、次にルートディスクブートです。

これにより、**director** がルートディスクとして使用する特定のディスクを識別しやすくなります。オーバークラウドの作成の開始時には、**director** はこのノードをプロビジョニングして、オーバークラウドのイメージをこのディスクに書き込みます。

## 6.6. 環境ファイルを使用したオーバークラウドのカスタマイズ

アンダークラウドには、オーバークラウドの作成プランとして機能するさまざまな **Heat** テンプレートが含まれます。YAML フォーマットの環境ファイルを使って、オーバークラウドの特性をカスタマイズすることができます。このファイルで、コア **Heat** テンプレートコレクションのパラメーターおよびリソースを上書きします。必要に応じていくつでも環境ファイルを追加することができますが、後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順番は重要です。以下の一覧は、環境ファイルの順序の例です。

- 各ロールおよびそのフレーバーごとのノード数。オーバークラウドを作成するには、この情報の追加は不可欠です。
- コンテナ化された **OpenStack** サービスのコンテナイメージの場所。このファイルは、「[5章 コンテナイメージのソースの設定](#)」で説明したオプションのいずれかで作成されたものです。
- 任意のネットワーク分離ファイル。Heat テンプレートコレクションの初期化ファイル (**environments/network-isolation.yaml**) から開始して、次にカスタムの NIC 設定ファイル、最後に追加のネットワーク設定の順番です。
- 外部のロードバランシングの環境ファイル

- Ceph Storage、NFS、iSCSI などのストレージ環境ファイル
- Red Hat CDN または Satellite 登録用の環境ファイル
- その他のカスタム環境ファイル

カスタム環境ファイルは、別のディレクトリーで管理することを推奨します (たとえば、**templates** ディレクトリー)。

『[Advanced Overcloud Customization](#)』ガイドを使用して、オーバークラウドの詳細機能をカスタマイズできます。



### 重要

基本的なオーバークラウドでは、ブロックストレージにローカルの LVM ストレージを使用しますが、この設定はサポートされません。ブロックストレージには、外部ストレージソリューション (Red Hat Ceph Storage 等) を使用することを推奨します。

## 6.7. CLI ツールを使用したオーバークラウドの作成

OpenStack 環境作成における最後の段階では、**openstack overcloud deploy** コマンドを実行して OpenStack 環境を作成します。このコマンドを実行する前に、キーオプションやカスタムの環境ファイルの追加方法を十分に理解しておく必要があります。



### 警告

バックグラウンドプロセスとして **openstack overcloud deploy** を実行しないでください。バックグラウンドのプロセスとして開始された場合にはオーバークラウドの作成は途中で停止してしまう可能性があります。

### オーバークラウドのパラメーター設定

以下の表では、**openstack overcloud deploy** コマンドを使用する際の追加パラメーターを一覧表示します。

表6.2 デプロイメントパラメーター

パラメーター	説明
<b>--templates [TEMPLATES]</b>	デプロイする Heat テンプレートが格納されているディレクトリー。空欄にした場合には、コマンドはデフォルトのテンプレートの場所である <b>/usr/share/openstack-tripleo-heat-templates/</b> を使用します。
<b>--stack STACK</b>	作成または更新するスタックの名前
<b>-t [TIMEOUT], --timeout [TIMEOUT]</b>	デプロイメントのタイムアウト (分単位)

パラメーター	説明
<code>--libvirt-type [LIBVIRT_TYPE]</code>	ハイパーバイザーに使用する仮想化タイプ
<code>--ntp-server [NTP_SERVER]</code>	時刻の同期に使用する Network Time Protocol (NTP) サーバー。コンマ区切りリストで複数の NTP サーバーを指定することも可能です (例: <code>--ntp-server 0.centos.pool.org,1.centos.pool.org</code> )。高可用性クラスターのデプロイメントの場合には、コントローラーが一貫して同じタイムソースを参照することが必須となります。標準的な環境には、確立された慣行によって、NTP タイムソースがすでに指定されている可能性がある点に注意してください。
<code>--no-proxy [NO_PROXY]</code>	環境変数 <code>no_proxy</code> のカスタム値を定義します。これにより、プロキシ通信から特定のホスト名は除外されます。
<code>--overcloud-ssh-user OVERCLOUD_SSH_USER</code>	オーバークラウドノードにアクセスする SSH ユーザーを定義します。通常、SSH アクセスは <code>heat-admin</code> ユーザーで実行されます。
<code>-e [EXTRA HEAT TEMPLATE],--extra-template [EXTRA HEAT TEMPLATE]</code>	オーバークラウドデプロイメントに渡す追加の環境ファイル。複数回指定することが可能です。 <code>openstack overcloud deploy</code> コマンドに渡す環境ファイルの順序が重要である点に注意してください。たとえば、逐次的に渡される各環境ファイルは、前の環境ファイルのパラメーターを上書きします。
<code>--environment-directory</code>	デプロイメントに追加する環境ファイルが格納されているディレクトリー。このコマンドは、これらの環境ファイルを最初に番号順、その後アルファベット順で処理します。
<code>--validation-errors-nonfatal</code>	オーバークラウドの作成プロセスでは、デプロイメントの前に一連のチェックが行われます。このオプションは、デプロイメント前のチェックで何らかの致命的でないエラーが発生した場合に終了します。どのようなエラーが発生してもデプロイメントが失敗するので、このオプションを使用することを推奨します。
<code>--validation-warnings-fatal</code>	オーバークラウドの作成プロセスでは、デプロイメントの前に一連のチェックが行われます。このオプションは、デプロイメント前のチェックで何らかのクリティカルではない警告が発生した場合に終了します。

パラメーター	説明
<code>--dry-run</code>	オーバークラウドに対する検証チェックを実行しますが、オーバークラウドを実際には作成しません。
<code>--skip-postconfig</code>	オーバークラウドのデプロイ後の設定を省略します。
<code>--force-postconfig</code>	オーバークラウドのデプロイ後の設定を強制的に行います。
<code>--skip-deploy-identifier</code>	<b>DeployIdentifier</b> パラメーターの一意的 ID 生成を省略します。ソフトウェア設定のデプロイメントステップは、実際に設定が変更された場合にしか実行されません。このオプションの使用には注意が必要です。特定のロールをスケールアウトする時など、ソフトウェア設定の実行が明らかに不要な場合にしか使用しないでください。
<code>--answers-file ANSWERS_FILE</code>	引数とパラメーターが記載された YAML ファイルへのパス
<code>--rhel-reg</code>	カスタマーポータルまたは Satellite 6 にオーバークラウドノードを登録します。
<code>--reg-method</code>	オーバークラウドノードに使用する登録方法。Red Hat Satellite 6 または Red Hat Satellite 5 は <b>satellite</b> 、カスタマーポータルは <b>portal</b>
<code>--reg-org [REG_ORG]</code>	登録に使用する組織
<code>--reg-force</code>	すでに登録済みでもシステムを登録します。
<code>--reg-sat-url [REG_SAT_URL]</code>	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、 <a href="https://satellite.example.com">https://satellite.example.com</a> ではなく <a href="http://satellite.example.com">http://satellite.example.com</a> を使用します。オーバークラウドの作成プロセスではこの URL を使用して、どのサーバーが Red Hat Satellite 5 または Red Hat Satellite 6 サーバーであるかを判断します。Red Hat Satellite 6 サーバーの場合は、オーバークラウドは <b>katello-ca-consumer-latest.noarch.rpm</b> ファイルを取得して <b>subscription-manager</b> に登録し、 <b>katello-agent</b> をインストールします。Red Hat Satellite 5 サーバーの場合にはオーバークラウドは <b>RHN-ORG-TRUSTED-SSL-CERT</b> ファイルを取得して <b>rhnreg_ks</b> に登録します。

パラメーター	説明
<b>--reg-activation-key</b> [REG_ACTIVATION_KEY]	登録に使用するアクティベーションキー

環境ファイルの `parameter_defaults` セクションに追加する Heat テンプレートのパラメーターの使用が優先されるため、一部のコマンドラインパラメーターは古いか非推奨となっています。以下の表では、非推奨となったパラメーターと、それに相当する Heat テンプレートのパラメーターを対照しています。

表6.3 非推奨の CLI パラメーターと Heat テンプレートのパラメーターの対照表

パラメーター	説明	Heat テンプレートのパラメーター
<b>--control-scale</b>	スケールアウトするコントローラーノード数	<b>ControllerCount</b>
<b>--compute-scale</b>	スケールアウトするコンピューターノード数	<b>ComputeCount</b>
<b>--ceph-storage-scale</b>	スケールアウトする Ceph Storage ノードの数	<b>CephStorageCount</b>
<b>--block-storage-scale</b>	スケールアウトする Cinder ノード数	<b>BlockStorageCount</b>
<b>--swift-storage-scale</b>	スケールアウトする Swift ノード数	<b>ObjectStorageCount</b>
<b>--control-flavor</b>	コントローラーノードに使用するフレーバー	<b>OvercloudControllerFlavor</b>
<b>--compute-flavor</b>	コンピューターノードに使用するフレーバー	<b>OvercloudComputeFlavor</b>
<b>--ceph-storage-flavor</b>	Ceph Storage ノードに使用するフレーバー	<b>OvercloudCephStorageFlavor</b>
<b>--block-storage-flavor</b>	Cinder ノードに使用するフレーバー	<b>OvercloudBlockStorageFlavor</b>
<b>--swift-storage-flavor</b>	Swift Storage ノードに使用するフレーバー	<b>OvercloudSwiftStorageFlavor</b>

パラメーター	説明	Heat テンプレートのパラメーター
<code>--neutron-flat-networks</code>	フラットなネットワークが <code>neutron</code> プラグインで設定されるように定義します。外部ネットワークを作成できるようにデフォルトは「 <code>datacentre</code> 」に設定されています。	<code>NeutronFlatNetworks</code>
<code>--neutron-physical-bridge</code>	各ハイパーバイザーで作成する Open vSwitch ブリッジ。デフォルト値は「 <code>br-ex</code> 」で、通常この値は変更する必要はないはずです。	<code>HypervisorNeutronPhysicalBridge</code>
<code>--neutron-bridge-mappings</code>	使用する論理ブリッジから物理ブリッジへのマッピング。ホスト ( <code>br-ex</code> ) の外部ブリッジを物理名 ( <code>datacentre</code> ) にマッピングするようにデフォルト設定されています。これは、デフォルトの Floating ネットワークに使用されます。	<code>NeutronBridgeMappings</code>
<code>--neutron-public-interface</code>	ネットワークノード向けにインターフェースを <code>br-ex</code> にブリッジするインターフェースを定義します。	<code>NeutronPublicInterface</code>
<code>--neutron-network-type</code>	Neutron のテナントネットワーク種別	<code>NeutronNetworkType</code>
<code>--neutron-tunnel-types</code>	<code>neutron</code> テナントネットワークのトンネリング種別。複数の値を指定するには、コンマ区切りの文字列を使用します。	<code>NeutronTunnelTypes</code>
<code>--neutron-tunnel-id-ranges</code>	テナントネットワークを割り当てるに使用できる GRE トンネリングの ID 範囲	<code>NeutronTunnelIdRanges</code>
<code>--neutron-vni-ranges</code>	テナントネットワークを割り当てるに使用できる VXLAN VNI の ID 範囲	<code>NeutronVniRanges</code>

パラメーター	説明	Heat テンプレートのパラメーター
<code>--neutron-network-vlan-ranges</code>	サポートされる Neutron ML2 および Open vSwitch VLAN マッピングの範囲。デフォルトでは、物理ネットワーク「 <code>datacentre</code> 」上の VLAN を許可するように設定されています。	<code>NeutronNetworkVLANRanges</code>
<code>--neutron-mechanism-drivers</code>	neutron テナントネットワークのメカニズムドライバー。デフォルトでは、「 <code>openvswitch</code> 」に設定されており、複数の値を指定するにはコンマ区切りの文字列を使用します。	<code>NeutronMechanismDrivers</code>
<code>--neutron-disable-tunneling</code>	VLAN で区切られたネットワークまたは neutron でのフラットネットワークを使用するためにトンネリングを無効化します。	パラメーターのマッピングなし
<code>--validation-errors-fatal</code>	オーバークラウドの作成プロセスでは、デプロイメントの前に一連のチェックが行われます。このオプションは、デプロイメント前のチェックで何らかの致命的なエラーが発生した場合に終了します。どのようなエラーが発生してもデプロイメントが失敗するので、このオプションを使用することを推奨します。	パラメーターのマッピングなし

これらのパラメーターは、Red Hat OpenStack Platform の今後のリリースで削除される予定です。



### 注記

オプションの完全一覧については、以下のコマンドを実行します。

```
(undercloud) $ openstack help overcloud deploy
```

## 6.8. オーバークラウド作成時の環境ファイルの追加

オーバークラウドをカスタマイズするには、`-e` を指定して、環境ファイルを追加します。必要に応じていくつでも環境ファイルを追加することができますが、後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順番は重要です。以下の一覧は、環境ファイルの順序の例です。

- 各ロールおよびそのフレーバーごとのノード数。オーバークラウドを作成するには、この情報の追加は不可欠です。

- コンテナ化された OpenStack サービスのコンテナイメージの場所。このファイルは、「[5章 コンテナイメージのソースの設定](#)」で説明したオプションのいずれかで作成されたものです。
- 任意のネットワーク分離ファイル。Heat テンプレートコレクションの初期化ファイル (**environments/network-isolation.yaml**) から開始して、次にカスタムの NIC 設定ファイル、最後に追加のネットワーク設定の順番です。
- 外部のロードバランシングの環境ファイル
- Ceph Storage、NFS、iSCSI などのストレージ環境ファイル
- Red Hat CDN または Satellite 登録用の環境ファイル
- その他のカスタム環境ファイル

-e オプションを使用してオーバークラウドに追加した環境ファイルはいずれも、オーバークラウドのスタック定義の一部となります。以下のコマンドは、追加するカスタム環境ファイルを使用してオーバークラウドの作成を開始する方法の一例です。

```
(undercloud) $ openstack overcloud deploy --templates \
  -e /home/stack/templates/node-info.yaml \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-
ansible/ceph-ansible.yaml \
  -e /home/stack/templates/ceph-custom-config.yaml \
  --ntp-server pool.ntp.org \
```

このコマンドでは、以下の追加オプションも使用できます。

#### --templates

**/usr/share/openstack-tripleo-heat-templates** の Heat テンプレートコレクションをベースとして使用し、オーバークラウドを作成します。

#### -e /home/stack/templates/node-info.yaml

各ロールに使用するノード数とフレーバーを定義する環境ファイルを追加します。以下に例を示します。

```
parameter_defaults:
  OvercloudControllerFlavor: control
  OvercloudComputeFlavor: compute
  OvercloudCephStorageFlavor: ceph-storage
  ControllerCount: 3
  ComputeCount: 3
  CephStorageCount: 3
```

#### -e /home/stack/templates/overcloud\_images.yaml

コンテナイメージのソースが記載された環境ファイルを追加します。詳しくは、「[5章 コンテナイメージのソースの設定](#)」を参照してください。

#### -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml

オーバークラウドデプロイメントのネットワーク分離を初期化する環境ファイルを追加します。



## 注記

**network-isolation.j2.yaml** は、このテンプレートの Jinja2 バージョンです。 **openstack overcloud deploy** コマンドは、Jinja2 テンプレートをプレーンの YAML ファイルにレンダリングします。このため、**openstack overcloud deploy** コマンドを実行する際には、レンダリングされる YAML ファイルの名前（この場合は **network-isolation.yaml**）を指定する必要があります。

### **-e /home/stack/templates/network-environment.yaml**

ネットワーク分離をカスタマイズする環境ファイルを追加します。

### **-e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**

Ceph Storage サービスを有効化するための環境ファイルを追加します。

### **-e /home/stack/templates/ceph-custom-config.yaml**

Ceph Storage の設定をカスタマイズするための環境ファイルを追加します。

### **--ntp-server pool.ntp.org**

時刻の同期に NTP サーバーを使用します。コントローラーノードクラスターの同期を保つには、このオプションが必要です。

**director** は、「[9章 オーバークラウド作成後のタスクの実行](#)」に記載の再デプロイおよびデプロイ後の機能にこれらの環境ファイルを必要とします。これらのファイルが含まれていない場合には、オーバークラウドが破損する可能性があります。

オーバークラウド設定を後で変更する予定の場合には、以下の作業を行う必要があります。

1. カスタムの環境ファイルおよび Heat テンプレートのパラメーターを変更します。
2. 同じ環境ファイルを指定して **openstack overcloud deploy** コマンドを再度実行します。

オーバークラウドを手動で編集しても、**director** を使用してオーバークラウドスタックの更新を行う際に **director** の設定で上書きされてしまうので、設定は直接編集しないでください。

## 環境ファイルディレクトリーの追加

**--environment-directory** オプションを使用して、環境ファイルを格納しているディレクトリー全体を追加することも可能です。デプロイメントコマンドにより、このディレクトリー内の環境ファイルは、最初に番号順、その後にアルファベット順で処理されます。この方法を使用する場合には、ファイル名に数字のプレフィックスを使用することを推奨します。以下に例を示します。

```
(undercloud) $ ls -1 ~/templates
00-node-info.yaml
10-network-isolation.yaml
20-network-environment.yaml
30-storage-environment.yaml
40-rhel-registration.yaml
```

以下のデプロイメントコマンドを実行してディレクトリーを追加します。

```
(undercloud) $ openstack overcloud deploy --templates --environment-
directory ~/templates
```

## 回答ファイルの使用

回答ファイルは、テンプレートおよび環境ファイルの追加を簡素化する YAML ファイルです。回答ファイルでは、以下のパラメーターを使用します。

### テンプレート

使用するコア Heat テンプレートコレクション。これは、**--templates** のコマンドラインオプションの代わりとして機能します。

### 環境

追加する環境ファイルの一覧。これは、**--environment-file (-e)** のコマンドラインオプションの代わりとして機能します。

たとえば、回答ファイルには以下の内容を含めることができます。

```
templates: /usr/share/openstack-tripleo-heat-templates/
environments:
  - ~/templates/00-node-info.yaml
  - ~/templates/10-network-isolation.yaml
  - ~/templates/20-network-environment.yaml
  - ~/templates/30-storage-environment.yaml
  - ~/templates/40-rhel-registration.yaml
```

以下のデプロイメントコマンドを実行して回答ファイルを追加します。

```
(undercloud) $ openstack overcloud deploy --answers-file ~/answers.yaml
```

## 6.9. オーバークラウドプランの管理

**openstack overcloud deploy** コマンドを使用する代わりに、**director** を使用してインポートしたプランを管理することもできます。

新規プランを作成するには、以下のコマンドを **stack** ユーザーとして実行します。

```
(undercloud) $ openstack overcloud plan create --templates
/usr/share/openstack-tripleo-heat-templates my-overcloud
```

このコマンドは、**/usr/share/openstack-tripleo-heat-templates** 内のコア Heat テンプレートコレクションからプランを作成します。**director** は、入力内容に基づいてプランに名前を指定します。この例では、**my-overcloud** という名前です。**director** は、この名前をオブジェクトストレージコンテナー、ワークフロー環境、オーバークラウドスタック名のラベルとして使用します。

以下のコマンドを使用して環境ファイルからパラメーターを追加します。

```
(undercloud) $ openstack overcloud parameters set my-overcloud
~/templates/my-environment.yaml
```

以下のコマンドを使用してプランをデプロイします。

```
(undercloud) $ openstack overcloud plan deploy my-overcloud
```

以下のコマンドを使用して既存のプランを削除します。

```
(undercloud) $ openstack overcloud plan delete my-overcloud
```



## 注記

**openstack overcloud deploy** コマンドは基本的に、これらのコマンドをすべて使用して既存のプランの削除、環境ファイルを使用した新規プランのアップロード、プランのデプロイを行います。

## 6.10. オーバークラウドのテンプレートおよびプランの検証

オーバークラウドの作成またはスタックの更新を実行する前に、**Heat** テンプレートと環境ファイルにエラーがないかどうかを検証します。

### レンダリングされたテンプレートの作成

オーバークラウドのコア **Heat** テンプレートは、**Jinja2** 形式となっています。テンプレートを検証するには、以下のコマンドを使用して **Jinja 2** のフォーマットなしでバージョンをレンダリングします。

```
(undercloud) $ openstack overcloud plan create --templates
/usr/share/openstack-tripleo-heat-templates overcloud-validation
(undercloud) $ mkdir ~/overcloud-validation
(undercloud) $ cd ~/overcloud-validation
(undercloud) $ swift download overcloud-validation
```

その後の検証テストには **~/overcloud-validation** のレンダリング済みテンプレートを使用します。

### テンプレート構文の検証

以下のコマンドを使用して、テンプレート構文を検証します。

```
(undercloud) $ openstack orchestration template validate --show-nested --
template ~/overcloud-validation/overcloud.yaml -e ~/overcloud-
validation/overcloud-resource-registry-puppet.yaml -e [ENVIRONMENT FILE] -
e [ENVIRONMENT FILE]
```



## 注記

検証には、**overcloud-resource-registry-puppet.yaml** の環境ファイルにオーバークラウド固有のリソースを含める必要があります。環境ファイルをさらに追加するには、このコマンドに **-e** オプションを使用してください。また、**--show-nested** オプションを追加して、ネストされたテンプレートからパラメーターを解決します。

このコマンドは、テンプレート内の構文エラーを特定します。テンプレートの構文の検証が正常に行われた場合には、出力には、作成されるオーバークラウドのテンプレートのプレビューが表示されます。

## 6.11. オーバークラウド作成の監視

オーバークラウドの作成プロセスが開始され、**director** によりノードがプロビジョニングされます。このプロセスは完了するまで多少時間がかかります。オーバークラウドの作成のステータスを確認するには、**stack** ユーザーとして別のターミナルを開き、以下のコマンドを実行します。

```
(undercloud) $ source ~/stackrc
(undercloud) $ openstack stack list --nested
```

`openstack stack list --nested` コマンドは、オーバークラウド作成の現在の段階を表示します。

## 6.12. オーバークラウドへのアクセス

`director` は、`director` ホストからオーバークラウドに対話するための設定を行い、認証をサポートするスクリプトを作成して、`stack` ユーザーのホームディレクトリーにこのファイル (`overcloudrc`) を保存します。このファイルを使用するには、以下のコマンドを実行します。

```
(undercloud) $ source ~/overcloudrc
```

これで、`director` のホストの CLI からオーバークラウドと対話するために必要な環境変数が読み込まれます。コマンドプロンプトが変わり、オーバークラウドと対話していることが示されます。

```
(overcloud) $
```

`director` のホストとの対話に戻るには、以下のコマンドを実行します。

```
(overcloud) $ source ~/stackrc  
(undercloud) $
```

オーバークラウドの各ノードには、`heat-admin` と呼ばれるユーザーが含まれます。`stack` ユーザーには、各ノードに存在するこのユーザーに SSH 経由でアクセスすることができます。SSH でノードにアクセスするには、希望のノードの IP アドレスを特定します。

```
(undercloud) $ openstack server list
```

次に、`heat-admin` ユーザーとノードの IP アドレスを使用して、ノードに接続します。

```
(undercloud) $ ssh heat-admin@192.168.24.23
```

## 6.13. オーバークラウド作成の完了

これで、コマンドラインツールを使用したオーバークラウドの作成が完了しました。作成後の機能については、「[9章 オーバークラウド作成後のタスクの実行](#)」を参照してください。

## 第7章 WEB UI を使用した基本的なオーバークラウドの設定

本章では、Web UI を使用した OpenStack Platform 環境の基本的な設定手順を説明します。基本設定のオーバークラウドには、カスタムの機能は含まれていませんが、『[Advanced Overcloud Customization](#)』ガイドに記載の手順に従って、この基本的なオーバークラウドに高度な設定オプションを追加して、仕様に合わせてカスタマイズすることができます。

本章の例では、すべてのノードが電源管理に IPMI を使用したベアメタルシステムとなっています。電源管理の種別およびそのオプションに関する詳細は、『[付録B 電源管理ドライバ](#)』を参照してください。

### ワークフロー

1. ノードの定義テンプレートと手動の登録を使用して、空のノードを登録します。
2. 全ノードのハードウェアを検査します。
3. `director` にオーバークラウドプランをアップロードします。
4. ノードをロールに割り当てます。

### 要件

- 『[4章 アンダークラウドのインストール](#)』で作成して UI を有効化した `director` ノード
- ノードに使用するベアメタルマシンのセット。必要なノード数は、作成予定のオーバークラウドのタイプにより異なります(オーバークラウドロールに関する情報は『[ノードのデプロイメントロールのプランニング](#)』を参照してください)。これらのマシンは、各ノード種別の要件セットに従う必要があります。これらの要件については、『[オーバークラウドの要件](#)』を参照してください。これらのノードにはオペレーティングシステムは必要ありません。`director` が Red Hat Enterprise Linux 7 のイメージを各ノードにコピーします。
- ネイティブ VLAN として設定したプロビジョニングネットワーク用のネットワーク接続1つ。全ノードは、このネイティブに接続して、『[ネットワーク要件](#)』で設定した要件に準拠する必要があります。
- その他のネットワーク種別はすべて、OpenStack サービスにプロビジョニングネットワークを使用しますが、ネットワークトラフィックの他のタイプに追加でネットワークを作成することができます。

### 7.1. WEB UI へのアクセス

`director` の Web UI には SSL 経由でアクセスします。たとえば、アンダークラウドの IP アドレスが `192.168.24.1` の場合には、UI にアクセスするためのアドレスは `https://192.168.24.1` です。Web UI はまず、以下のフィールドが含まれるログイン画面を表示します。

- **Username:** `director` の管理ユーザー。デフォルトは `admin` です。
- **Password:** 管理ユーザーのパスワード。アンダークラウドホストのターミナルで `stack` ユーザーとして `sudo hiera admin_password` を実行してパスワードを確認してください。

UI へのログイン時に、UI は OpenStack Identity のパブリック API にアクセスして、他のパブリック API サービスのエンドポイントを取得します。これらのサービスには、以下が含まれます。

コンポーネント	UI の目的
OpenStack Identity ( <b>keystone</b> )	UI への認証と他のサービスのエンドポイント検出
OpenStack Orchestration ( <b>heat</b> )	デプロイメントのステータス
OpenStack Bare Metal ( <b>ironic</b> )	ノードの制御
OpenStack Object Storage ( <b>swift</b> )	Heat テンプレートコレクションまたはオーバークラウドの作成に使用したプランのストレージ
OpenStack Workflow ( <b>mistral</b> )	<b>director</b> タスクのアクセスおよび実行
OpenStack Messaging ( <b>zaqar</b> )	特定のタスクのステータスを検索する <b>Websocket</b> ベースのサービス

UI は、これらのパブリック API と直接対話します。そのため、クライアントシステムにはそのエンドポイントへのアクセスが必要です。**director** は、パブリック仮想 IP (**undercloud.conf** ファイルの **undercloud\_public\_host**) 上の SSL/TLS で暗号化されたパスを介してこれらのエンドポイントを公開します。各パスは、サービスに対応します。たとえば、**https://192.168.24.2:443/keystone** は OpenStack Identity パブリック API にマッピングします。

エンドポイントを変更したり、エンドポイントアクセスに別の IP を使用したりする予定の場合には、**director** UI は **/var/www/openstack-tripleo-ui/dist/tripleo\_ui\_config.js** ファイルから設定を読み取ります。このファイルは以下のパラメーターを使用します。

パラメーター	説明
<b>keystone</b>	OpenStack Identity ( <b>keystone</b> ) サービスのパブリック API。UI は自動的にこのサービスを使用して、他のサービスのエンドポイントを自動検出するので、このパラメーターだけを定義するだけで結構です。ただし、必要に応じて、他のエンドポイントのカスタム URL を定義することができます。
<b>heat</b>	OpenStack Orchestration ( <b>heat</b> ) サービスのパブリック API
<b>ironic</b>	OpenStack Bare Metal ( <b>ironic</b> ) サービスのパブリック API
<b>swift</b>	OpenStack Object Storage ( <b>heat</b> ) サービスのパブリック API
<b>mistral</b>	OpenStack Workflow ( <b>mistral</b> ) サービスのパブリック API

パラメーター	説明
<b>zaqar-websocket</b>	OpenStack Messaging ( <b>zaqar</b> ) サービスの Websocket
<b>zaqar_default_queue</b>	OpenStack Messaging ( <b>zaqar</b> ) サービスに使用するメッセージングキュー。デフォルトは <b>tripleo</b> です。
<b>excludedLanguages</b>	UI は複数の言語に翻訳されており、ログイン画面から、または UI 内で選択することができます。ITEF 言語コードを元に、特定の言語を除外することができます。次の言語コードを除外することができます: <b>de</b> 、 <b>en-GB</b> 、 <b>es</b> 、 <b>fr</b> 、 <b>id</b> 、 <b>ja</b> 、 <b>ko-KR</b> 、 <b>tr-TR</b> 、 <b>zh-CN</b>

以下は **tripleo\_ui\_config.js** ファイルのサンプルです。**192.168.24.2** はアンダークラウドのパブリック仮想 IP です。

```

window.tripleoUiConfig = {
  'keystone': 'https://192.168.24.2:443/keystone/v2.0',
  'heat': 'https://192.168.24.2:443/heat/v1/(tenant_id)s',
  'ironic': 'https://192.168.24.2:443/ironic',
  'mistral': 'https://192.168.24.2:443/mistral/v2',
  'swift': 'https://192.168.24.2:443/swift/v1/AUTH_(tenant_id)s',
  'zaqar-websocket': 'wss://192.168.24.2:443/zaqar',
  'zaqar_default_queue': "tripleo",
  'excludedLanguages': [],
  'loggers': ["console", "zaqar"]
};

```

## 7.2. WEB UI のナビゲーション

UI は主に 3 つのセクションで構成されています。

### プラン

UI 上部のメニューアイテム。このページは UI のメインセクションとして機能し、オーバークラウドの作成に使用するプラン、各ロールに割り当てるノード、現在のオーバークラウドのステータスを定義できます。このセクションでは、デプロイメントワークフローが提供され、デプロイメントのパラメーターの設定やロールへのノードの割り当てなどオーバークラウドの作成プロセスをステップごとにガイドします。

overcloud

## 1 Prepare Hardware ⓘ

+ Register Nodes

## 2 Specify Deployment Configuration ⓘ

Base resources configuration, Containerized Deployment, environments/docker-ha.yaml [Edit Configuration](#)

## 3 Configure Roles and Assign Nodes ⓘ

7 Nodes available to assign

## 4 Deploy ⓘ

Validate and Deploy

## ノード

UI 上部のメニューアイテム。このページはノードの設定セクションとして機能し、新規ノードの登録や登録したノードのイントロスペクションの手段を提供します。このセクションには、電源の状態、イントロスペクションのステータス、プロビジョニングの状態、およびハードウェア情報なども表示されます。

Nodes

Refresh Results + Register Nodes

Name	↓	↑	Introspect Nodes	Provide Nodes	
9 Nodes <span style="float: right;">Select All</span>					
> <input type="checkbox"/>		node01	Off   Introspection: finished   Provision State: available	Profile: compute	1 CPU Core 4096 MB RAM 49 GB Disk
> <input type="checkbox"/>		node02	Off   Introspection: finished   Provision State: available	Profile: compute	1 CPU Core 4096 MB RAM 49 GB Disk
> <input type="checkbox"/>		node03	Off   Introspection: finished   Provision State: available	Profile: control	2 CPU Cores 10240 MB RAM 99 GB Disk
> <input type="checkbox"/>		node04	Off   Introspection: finished   Provision State: available	Profile: -	2 CPU Cores 10240 MB RAM 99 GB Disk
> <input type="checkbox"/>		node05	Off   Introspection: finished   Provision State: available	Profile: -	2 CPU Cores 10240 MB RAM 99 GB Disk

各ノードの右端にあるオーバーフローメニュー項目 (3つの点) をクリックすると、選択したノードのディスク情報が表示されます。

Node Drives - fb2d6c82-1063-4a5e-86e4-58c4b920616e

✕

✓ /dev/sda
✕

Root Device

Type: **HDD**    Size: **299.44** GB

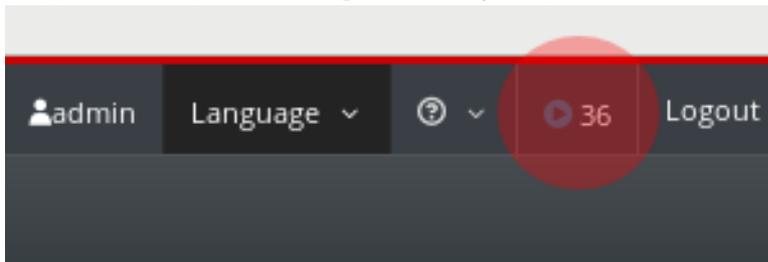
---

<b>Model:</b>	PERC H330 Mini	✕
<b>Serial:</b>	61866da04f37e8001ea4e109127d48f0	
<b>Vendor:</b>	DELL	
<b>WWN:</b>	0x61866da04f37e800	
<b>WWN Vendor Extension:</b>	0x1ea4e109127d48f0	
<b>WWN with Extension:</b>	0x61866da04f37e8001ea4e109127d48f0	

Close

## 検証

**検証** メニューオプションをクリックすると、ページの右側にパネルが表示されます。



このセクションには、以下の時点で実施される、さまざまなシステムチェックが表示されます。

- デプロイメント前
- デプロイメント後
- イントロスペクション前
- アップグレード前
- アップグレード後

これらの検証タスクは、デプロイメントの特定の時点で自動的に実行されますが、手動で実行することもできます。実行する検証タスクの**再生** ボタンをクリックします。各検証タスクのタイトルをクリックして実行するか、検証タイトルをクリックして詳細情報を表示します。

Validations
 Refresh

Name ▾

**32 Validations**

---

✓

**Undercloud Services Debug Check**

The undercloud's openstack services should `_not_ hav...`

pre-deployment

---

✓

**Validate the Heat environment file for netwo...**

This validates the network environment and nic-config...

pre-deployment

---

▶

**Verify NoOpFirewallDriver is set in Nova**

When using Neutron, the ``firewall_driver`` option in N...

post-deployment

### 7.3. WEB UI を使用したオーバークラウドプランのインポート

director UI には、オーバークラウドの設定の前にプランが必要です。このプランは通常、`/usr/share/openstack-tripleo-heat-templates` にあるアンダークラウド上のテンプレートなど、Heat テンプレートコレクションです。さらに、ハードウェアや環境の要件に合わせてプランをカスタマイズすることができます。オーバークラウドのカスタマイズに関する詳しい情報は『[Advanced Overcloud Customization](#)』ガイドを参照してください。

プランには、オーバークラウドの設定に関する主要な手順が表示されます。

1. **ハードウェアの準備:** ノードの登録およびイントロスペクション
2. **デプロイメントの設定の指定:** オーバークラウドのパラメーターの設定と追加する環境ファイルの定義
3. **ロールの設定とノードの割り当て:** ロールへのノード割り当てと、ロール固有のパラメーターの変更
4. **デプロイ:** オーバークラウド作成の開始

アンダークラウドのインストールと設定を実行すると、プランが自動的にアップロードされます。Web UI に複数のプランをインポートすることも可能です。プラン画面のパンくずリストですべてのプラン

をクリックすると、現在のプランのリストが表示されます。プランを切り替えるには、カードをクリックします。

プランのインポートをクリックすると、以下の情報を尋ねるウィンドウが表示されます。

- **プラン名:** `overcloud` など、プランのプレーンテキスト名
- **アップロードの種別:** **Tar アーカイブ (tar.gz)**、全 ローカルフォルダー (Google Chrome のみ) のいずれをアップロードするかを選択します。
- **プランファイル:** ブラウザーをクリックして、ローカルのファイルシステム上のプランを選択します。

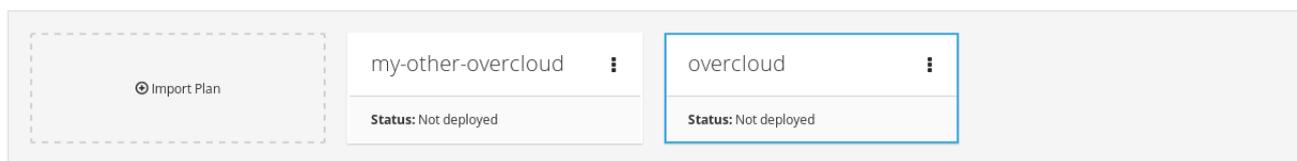
`director` の Heat テンプレートコレクションをクライアントのマシンにコピーする必要がある場合には、ファイルをアーカイブしてコピーします。

```
$ cd /usr/share/openstack-tripleo-heat-templates/
$ tar -cf ~/overcloud.tar *
$ scp ~/overcloud.tar user@10.0.0.55:~/.
```

`director` UI がプランをアップロードしたら、プランがプランのリストに表示され、設定を行うことができます。選択するプランのカードをクリックします。

Plans

+ Import Plan



## 7.4. WEB UI を使用したノードの登録

オーバークラウド設定の最初の手順は、ノードの登録です。以下のいずれかで、ノードの登録プロセスを開始します。

- プラン画面の **1ハードウェアの準備** にある **ノードの登録** をクリックします。
- ノード画面の **ノードの登録** をクリックします。

ノードの登録ウィンドウが表示されます。

Register Nodes
✕

+ Add New or Upload From File

⊖
Undefined Node
✕

Node Detail

General

Name

Management

Driver

IPMI IP Address or FQDN \*

IPMI Port

IPMI Username \*

IPMI Password \*

Hardware

Architecture

CPU count

Memory (MB)

Disk (GB)

Networking

NIC MAC Addresses \*

Please enter a valid MAC Address.  
If you are specifying multiple MAC Addresses, please enter a comma separated list. (e.g. aa:bb:cc:dd:ee:ff,12:34:56:78:90:xx,do:re:mi:fa:so:ra)

Cancel
Register Nodes

director には、登録するノードの一覧が必要です。以下のいずれかの方法でリストを取得できます。

1. **ノード定義テンプレートのアップロード**: これには、**ファイルからアップロード** ボタンをクリックして、ファイルを選択してください。ノードの定義テンプレートの構文については、「[オーバークラウドへのノードの登録](#)」を参照してください。
2. **各ノードの手動登録**: これには、**新規追加** をクリックして、ノードの詳細を提供します。

手動登録に必要な情報は以下のとおりです。

### 名前

ノードのプレーンテキスト名。RFC3986 の非予約文字のみを使用するようにしてください。

### ドライバー

使用する電源管理ドライバー。この例では IPMI ドライバーを使用しますが (pxe\_ipmitool)、他のドライバーも利用できます。利用可能なドライバーについては、「[付録B 電源管理ドライバー](#)」を参照してください。

### IPMI IP アドレス

IPMI デバイスの IP アドレス

### IPMI ポート

IPMI デバイスにアクセスするためのポート

### IPMI のユーザー名およびパスワード

IPMI のユーザー名およびパスワード

### アーキテクチャー

(オプション)システムアーキテクチャー

#### CPU 数

(オプション)ノード上の CPU 数

#### メモリー (MB)

(オプション)メモリーサイズ (MB)

#### ディスク (GB)

(オプション)ハードディスクのサイズ (GB)

#### NIC の MAC アドレス

ノード上のネットワークインターフェースの MAC アドレス一覧。各システムのプロビジョニング NIC の MAC アドレスのみを使用します。



#### 注記

UI では、**Dell Remote Access Controller (DRAC)** 電源管理を使用するノードの登録を行うこともできます。これらのノードでは、**pxe\_drac** ドライバーが使われます。詳細は、「[Dell Remote Access Controller \(DRAC\)](#)」を参照してください。

ノードの情報を入力した後に、ウィンドウの下の **ノードの登録** をクリックします。

**director** によりノードが登録されます。登録が完了すると、UI を使用してノードのイントロスペクションを実行できるようになります。

## 7.5. WEB UI を使用したノードのハードウェアのイントロスペクション

**director** UI は各ノードでイントロスペクションプロセスを実行することができます。このプロセスを実行すると、各ノードが PXE を介してイントロスペクションエージェントを起動します。このエージェントは、ノードからハードウェアのデータを収集して、**director** に送り返します。次に **director** は、**director** 上で実行中の **OpenStack Object Storage (swift)** サービスにこのイントロスペクションデータを保管します。**director** は、プロファイルのタグ付け、ベンチマーキング、ルートディスクの手動割り当てなど、さまざまな目的でハードウェア情報を使用します。

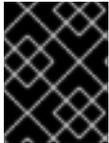


#### 注記

ポリシーファイルを作成して、イントロスペクションの直後にノードをプロファイルに自動でタグ付けすることも可能です。ポリシーファイルを作成してイントロスペクションプロセスに組み入れる方法に関する詳しい情報は、「[付録E プロファイルの自動タグ付け](#)」を参照してください。または、UI を使用してプロファイルにノードをタグ付けすることもできます。手動でのノードのタグ付けに関する情報は、「[Web UI を使用したルールへのノードの割り当て](#)」を参照してください。

イントロスペクションのプロセスを開始するには以下のステップを実行します。

1. ノード画面に移動します。
2. イントロスペクションを行うノードをすべて選択します。
3. イントロスペクションをクリックします。



## 重要

このプロセスが最後まで実行されて正常に終了したことを確認してください。ペアメタルの場合には、通常 15 分ほどかかります。

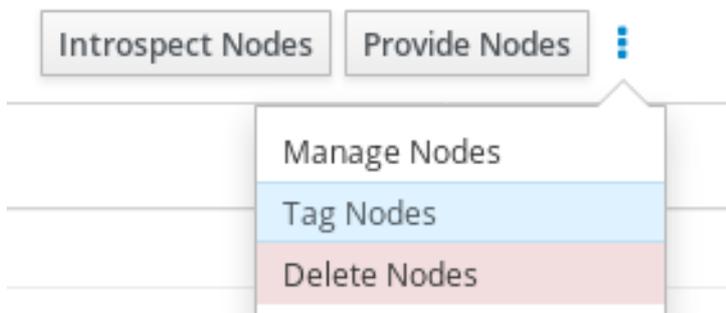
イントロスペクションのプロセスを完了したら、**プロビジョニングの状態**が **manageable** に設定されているノードをすべて選択して、**プロビジョン** ボタンをクリックします。**プロビジョニングの状態**が **available** になるまで待ちます。

ノードのタグ付けおよびプロビジョニングの準備ができました。

## 7.6. WEB UI を使用したプロファイルへのノードのタグ付け

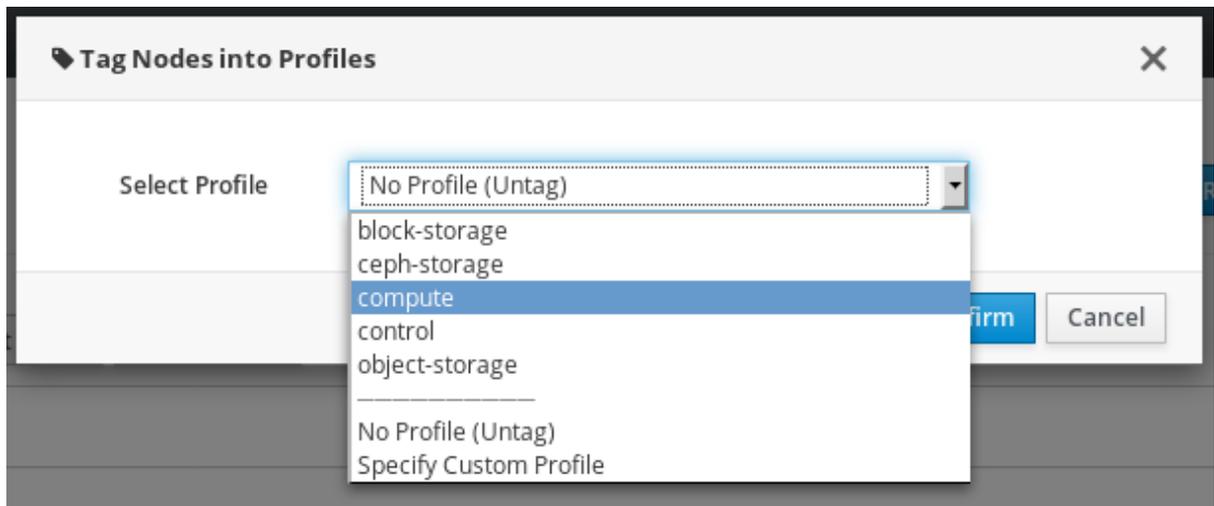
各ノードにプロファイルのセットを割り当てることができます。それぞれのプロファイルが、個々のフレーバーおよびロールに対応します (詳細は、「[プロファイルへのノードのタグ付け](#)」を参照してください)。

ノード画面には、さらにトグルメニューがあり、ノードのタグ付けなどのその他のノード管理操作を選択することができます。



ノードのセットをタグ付けするには、以下の手順を実行します。

1. タグ付けするノードをチェックボックスで選択します。
2. メニュートグルをクリックします。
3. ノードのタグ付けをクリックします。
4. 既存のプロファイルを選択します。新規プロファイルを作成するには、**カスタムプロファイル**を指定するを選択して**カスタムプロファイル**に名前を入力します。



### 注記

カスタムプロファイルを作成する場合は、プロファイルタグを新規フレーバーに割り当てる必要もあります。新規フレーバー作成についての詳しい情報は、「[プロファイルへのノードのタグ付け](#)」を参照してください。

5. **確認** をクリックしてノードをタグ付けします。

## 7.7. WEB UI を使用したオーバークラウドプランのパラメーターの編集

プラン画面では、アップロードしたプランをカスタマイズすることができます。**2 デプロイメントの設定の指定**で、**設定の編集** リンクをクリックして、ベースのオーバークラウドの設定を変更します。

ウィンドウには、2つの主要なタブが表示されます。

### 全体の設定

このタブでは、オーバークラウドからの異なる機能を追加する方法を提供します。これらの機能は、プランの **capabilities-map.yaml** ファイルに定義されており、機能毎に異なるファイルを使用します。たとえば、**Storage** で **Storage Environment** を選択すると、プランは **environments/storage-environment.yaml** ファイルにマッピングされ、オーバークラウドの NFS、iSCSI、Ceph の設定を行うことができます。**Other** タブには、プランで検出されているが、プランに含まれるカスタムの環境ファイルを追加するのに役立つ **capabilities-map.yaml** には記載されていない環境ファイルが一覧表示されます。追加する機能を選択したら、**変更の保存** をクリックしてください。

**Deployment Configuration** ✕

Overall Settings

Parameters

Security

Network Configuration

Nova Extensions

Utilities

Operational Tools

Neutron Plugin Configuration

Other

Additional Services

Storage

General Deployment Options

**Security Hardening Options**

**TLS**

**SSL on OpenStack Public Endpoints**  
Use this option to pass in certificates for SSL deployments. For these values to take effect, one of the TLS endpoints options must also be used.

---

**TLS Endpoints**

**Deploy All SSL Endpoints as DNS names**  
Use this option when deploying an overcloud where all the endpoints are DNS names and there's TLS in all endpoint types.

**SSL-enabled deployment with DNS name as public endpoint**  
Use this option when deploying an SSL-enabled overcloud where the public endpoint is a DNS name.

**SSL-enabled deployment with IP address as public endpoint**  
Use this option when deploying an SSL-enabled overcloud where the public endpoint is an IP address.

---

**SSH Banner Text**  
Enables population of SSH Banner Text

**SSH Banner Text**

---

**Horizon Password Validation**  
Enable Horizon Password validation

**Horizon Password Validation**

---

**AuditD Rules**  
Management of AuditD rules

**AuditD Rule Management**

---

**Keystone CADF auditing**  
Enable CADF notifications in Keystone for auditing

**Keystone CADF auditing**

## パラメーター

このタブには、オーバークラウドのさまざまなベースレベルパラメーターおよび環境ファイルパラメーターが含まれます。パラメーターを変更した場合には **変更の保存** をクリックしてください。

**Deployment Configuration** ✕

Overall Settings

Parameters

General

Base resources configuration

Containerized Deployment

environments/docker-ha.yaml

**AddVipsToEtcHosts**  Set to true to append per network Vips to /etc/hosts on each node.

**BlockStorageCount**   
Number of BlockStorage nodes to deploy

**BlockStorageExtraConfig**   
Role specific additional hiera configuration to inject into the cluster.

**BlockStorageHostnameFormat**   
Format for BlockStorage node hostnames Note %index% is translated into the index of the node, e.g 0/1/2 etc and %stackname% is replaced with the stack name e.g overcloud

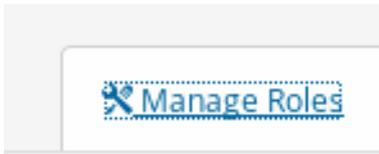
**BlockStorageParameters**   
Optional Role Specific parameters to be provided to service

**BlockStorageRemovalPolicies**   
List of resources to be removed from BlockStorage ResourceGroup when doing an update which requires removal of specific resources. Example format ComputeRemovalPolicies: [{"resource\_list": ["0"]}]

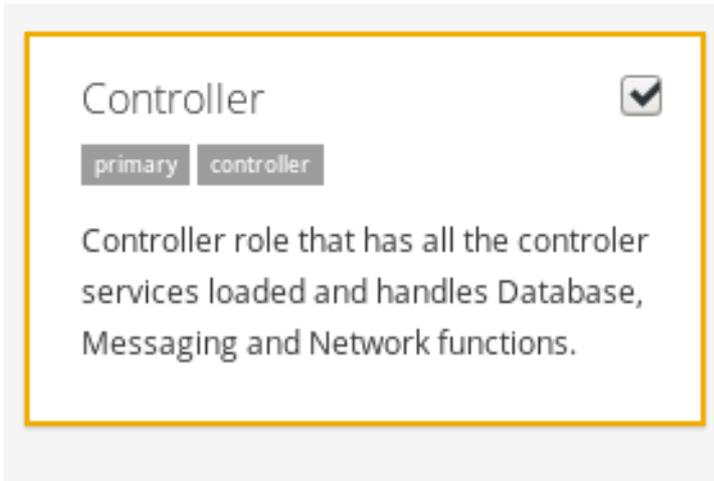
**BlockStorageSchedulerHints**   
Optional scheduler hints to pass to nova

## 7.8. WEB UI でのロールの追加

ロールの設定とノードの割り当てセクションの右下には、**ロールの管理**アイコンがあります。



このアイコンをクリックすると、環境に追加できるロールを示すカードの選択肢が表示されます。ロールを追加するには、そのロールの右上にあるチェックボックスにチェックを付けます。

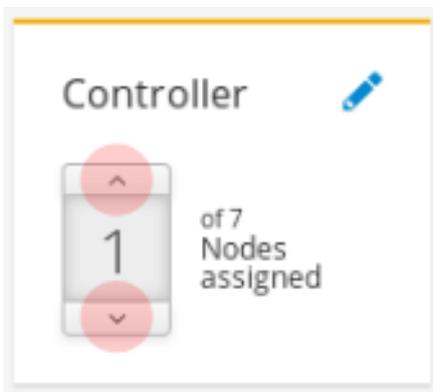


ロールを選択したら、**変更の保存** をクリックします。

## 7.9. WEB UI を使用したロールへのノードの割り当て

各ノードのハードウェアを登録、検査した後は、プランの画面からロールに割り当てます。

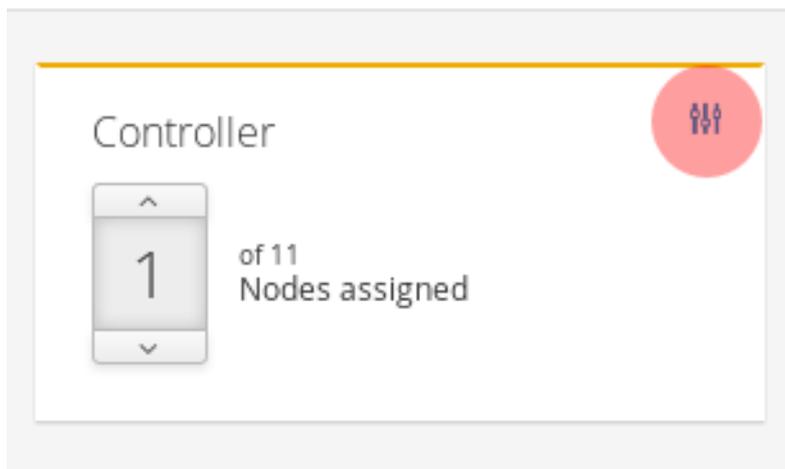
ロールにノードを割り当てるには、プランの画面で **3** **ロールの設定とノードの割り当て** セクションまでスクロールします。各ロールで、スピナーウィジェットを使ってノード数をロールに割り当てます。それぞれのロールで利用可能なノードの数は、「[Web UI を使用したプロフィールへのノードのタグ付け](#)」でタグ付けしたノードに基づきます。



この操作により、各ロールの **\*Count** パラメーターが変更されます。たとえば、コントローラーロールのノード数を **3** に変更すると、**ControllerCount** パラメーターが **3** に設定されます。デプロイメント設定の **パラメータータブ** で、これらのカウント値を表示および編集することもできます。詳細は、「[WEB UI を使用したオーバークラウドプランのパラメーターの編集](#)」を参照してください。

## 7.10. WEB UI を使用したロールパラメーターの編集

各ノードのロールにより、ロール固有のパラメーターを設定する手段が提供されます。プランの画面で **3** ロールの設定とノードの割り当てセクションまでスクロールします。ロール名の横にある **Edit Role Parameters** アイコンをクリックします。



ウィンドウには、2つの主要なタブが表示されます。

### パラメーター

これには、さまざまなロール固有のパラメーターが含まれます。たとえば、コントローラーロールを編集する場合には、**OvercloudControlFlavor** パラメーターを使用して、そのロールのデフォルトのフレーバーを変更することができます。ロール固有のパラメーターを変更したら、**変更の保存** をクリックします。

Controller Role
✕

Parameters

Services

Network Configuration

CloudDomain	localdomain	The DNS domain used for the hosts. This must match the overcloud_domain_name configured on the undercloud.
ConfigCollectSplay	30	Maximum amount of time to possibly to delay configuration collection polling. Defaults to 30 seconds. Set to 0 to disable it which will cause the configuration collection to occur as soon as the collection process starts. This setting is used to prevent the configuration collection processes from polling all at the exact same time.
ConfigCommand	os-refresh-config --timeout 14400	Command which will be run whenever configuration data changes
ControllerExtraConfig	{}	Role specific additional hiera configuration to inject into the cluster.
controllerExtraConfig	{}	DEPRECATED use ControllerExtraConfig instead
ControllerImage	overcloud-full	The disk image file to use for the role.

### サービス

これにより、選択したロールのサービス固有のパラメーターが定義されます。左のパネルでは、選択して変更したサービス一覧が表示されます。たとえば、タイムゾーンを変更するには、**OS::TripleO::Services::Timezone** サービスをクリックして **TimeZone** パラメーターを希

望のタイムゾーンに変更します。サービス固有のパラメーターを変更したら、**変更の保存** をクリックしてください。

**Controller Role** [X]

Parameters | **Services** | Network Configuration

**AodhApi**

- AodhApiPolicies: 

A hash of policies to configure for Aodh API. e.g. { aodh-context\_is\_admin: { key: context\_is\_admin, value: 'role:admin' } }
- AodhDebug: 

Set to True to enable debugging Aodh services.
- AodhPassword: 

The password for the aodh services.
- ApacheMaxRequestWorkers: 

Maximum number of simultaneously processed requests.
- ApacheServerLimit: 

Maximum number of Apache processes.
- Debug: 

Set to True to enable debugging on all services.
- DockerAodhApiImage: 

image

## ネットワーク設定

ネットワーク設定では、オーバークラウドのさまざまなネットワークに対して IP アドレスまたはサブネットの範囲を定義できます。

**Controller Role** [X]

Parameters | Services | **Network Configuration**

Software Config to drive os-net-config for a simple bridge.

- ControlPlaneIp:
- ExternalIpSubnet: 

IP address/subnet on the external network
- InternalApiIpSubnet: 

IP address/subnet on the internal\_api network
- ManagementIpSubnet: 

IP address/subnet on the management network
- StorageIpSubnet: 

IP address/subnet on the storage network
- StorageMgmtIpSubnet: 

IP address/subnet on the storage\_mgmt network
- TenantIpSubnet:



### 重要

ロールのサービスパラメーターは UI に表示されますが、デフォルトではサービスは無効になっている場合があります。「[WEB UI を使用したオーバークラウドプランのパラメーターの編集](#)」の説明に沿って、これらのサービスを有効化することができます。これらのサービスの有効化に関する情報は、『[Advanced Overcloud Customization](#)』ガイドのコンポーザブルロールのセクションも参照してください。

## 7.11. WEB UI を使用したオーバークラウドの作成開始

オーバークラウドプランが設定されたら、オーバークラウドのデプロイメントを開始することができます。これには、**4 デプロイ** セクションまでスクロールして、**検証とデプロイ** をクリックしてください。



アンダークラウドの検証をすべて実行しなかった場合や、すべての検証に合格しなかった場合には、警告メッセージが表示されます。アンダークラウドのホストが要件を満たしていることを確認してから、デプロイメントを実行してください。



## Deploy Plan overcloud

**Summary:** Base resources configuration, Containerized Deployment, environments/docker-ha.yaml



**Not all pre-deployment validations have passed.**

It is highly recommended that you resolve all validation issues before continuing.

Are you sure you want to deploy this plan?



デプロイメントの準備ができたなら **デプロイ** をクリックしてください。

UI では、定期的に オーバークラウド作成の進捗がモニタリングされ、現在の進捗の割合を示すプログレスバーが表示されます。[詳細情報の表示](#) リンクでは、オーバークラウドにおける現在の **OpenStack Orchestration** スタックのログが表示されます。

**Plan overcloud deployment** ✕

○ Deployment in progress 31%

Resources

Name	Status	Updated Time
MysqlRootPassword	CREATE_COMPLETE	2016-11-24T07:00:08Z
PcsdPassword	CREATE_COMPLETE	2016-11-24T07:00:08Z
VipMap	CREATE_COMPLETE	2016-11-24T07:00:08Z
RabbitCookie	CREATE_COMPLETE	2016-11-24T07:00:08Z
Controller	INIT_COMPLETE	2016-11-24T07:00:08Z
ObjectStorage	INIT_COMPLETE	2016-11-24T07:00:08Z
ObjectStorageIplListMap	INIT_COMPLETE	2016-11-24T07:00:08Z
ControllerIplListMap	INIT_COMPLETE	2016-11-24T07:00:08Z
BlockStorageServiceChain	CREATE_IN_PROGRESS	2016-11-24T07:00:08Z
ComputeHostsDeployment	INIT_COMPLETE	2016-11-24T07:00:08Z
RedisVirtualIP	CREATE_COMPLETE	2016-11-24T07:00:08Z
StorageVirtualIP	CREATE_COMPLETE	2016-11-24T07:00:08Z

オーバークラウドのデプロイメントが完了するまで待ちます。

オーバークラウドの作成プロセスが完了したら、**4 デプロイ** セクションに、現在のオーバークラウドの

状況と以下の詳細が表示されます。

- **オーバークラウドの IP アドレス:** オーバークラウドにアクセスするための IP アドレス
- **パスワード:** オーバークラウドの **admin** ユーザーのパスワード

この情報を使用してオーバークラウドにアクセスします。

**Deployment succeeded**

Stack CREATE completed successfully

Overcloud information:

- Overcloud IP address: [REDACTED]
- Username: admin
- Password: [REDACTED]

 Delete Deployment

## 7.12. オーバークラウド作成の完了

これで **director** の UI を使用したオーバークラウドの作成が完了しました。作成後の機能については、「[9章 オーバークラウド作成後のタスクの実行](#)」を参照してください。

## 第8章 事前にプロビジョニングされたノードを使用した基本的なオーバークラウドの設定

本章では、OpenStack Platform 環境を設定します。事前にプロビジョニングされたノードを使用して OpenStack Platform 環境を設定する基本的な手順を説明します。以下のシナリオは、標準のオーバークラウド作成のシナリオとはさまざまな点で異なります。

- 外部ツールを使用してノードをプロビジョニングしてから、**director** でオーバークラウドの設定のみを制御することができます。
- **director** のプロビジョニングの方法に依存せずにノードを使用することができます。これは、電源管理制御なしでオーバークラウドを作成する場合や、DHCP/PXE ブートの制限があるネットワークを使用する場合に便利です。
- **director** は、ノードの管理に OpenStack Compute (nova)、OpenStack Bare Metal (ironic) または OpenStack Image (glance) を使用しません。
- 事前にプロビジョニングされたノードは、カスタムのパーティションレイアウトを使用しません。

このシナリオでは、カスタム機能のない基本的な設定を行いますが、『[Advanced Overcloud Customization](#)』ガイドに記載の手順に従って、この基本的なオーバークラウドに高度な設定オプションを追加して、仕様に合わせてカスタマイズすることができます。



### 重要

事前プロビジョニングされたノードと **director** がプロビジョニングしたノードが混在するオーバークラウド環境はサポートされていません。

### 要件

- [4章アンダークラウドのインストール](#)で作成した **director** ノード
- ノードに使用するベアメタルマシンのセット。必要なノード数は、作成予定のオーバークラウドのタイプにより異なります (オーバークラウドロールに関する情報は「[ノードのデプロイメントロールのプランニング](#)」を参照してください)。これらのマシンは、各ノード種別の要件セットに従う必要があります。これらの要件については、「[オーバークラウドの要件](#)」を参照してください。これらのノードには Red Hat Enterprise Linux オペレーティングシステムの最新のマイナーバージョンが必要です。
- 事前にプロビジョニングされたノードを管理するためのネットワーク接続1つ。このシナリオでは、オーケストレーションエージェントの設定のために、ノードへの SSH アクセスが中断されないようにする必要があります。
- コントロールプレーンネットワーク用のネットワーク接続1つ。このネットワークには、主に2つのシナリオがあります。
  - プロビジョニングネットワークをコントロールプレーンとして使用するデフォルトのシナリオ。このネットワークは通常、事前にプロビジョニングされたノードから **director** への Layer 3 (L3) を使用したルーティング可能なネットワーク接続です。このシナリオの例では、以下の IP アドレスの割り当てを使用します。

表8.1 プロビジョニングネットワークの IP 割り当て

ノード名	IP アドレス
director	192.168.24.1
コントローラー	192.168.24.2
Compute	192.168.24.3

- 別のネットワークを使用するシナリオ。director のプロビジョニングネットワークがプライベートのルーティング不可能なネットワークの場合には、サブネットからノードの IP アドレスを定義して、パブリック API エンドポイント経由で director と通信することができます。このシナリオには特定の注意事項があります。これについては、本章の後半の「[オーバークラウドノードに別のネットワークを使用する方法](#)」で考察します。
- この例で使用するその他すべてのネットワーク種別には、OpenStack サービス用のコントロールプレーンネットワークも使用しますが、他のネットワークトラフィック種別用に追加のネットワークを作成することができます。

## 8.1. ノード設定のためのユーザーの作成

このプロセスの後半では、director がオーバークラウドノードに **stack** ユーザーとして SSH アクセスする必要があります。

1. 各オーバークラウドノードで、**stack** という名前のユーザーを作成して、それぞれにパスワードを設定します。たとえば、コントローラーノードでは以下のコマンドを使用します。

```
[root@controller ~]# useradd stack
[root@controller ~]# passwd stack # specify a password
```

2. **sudo** を使用する際に、このユーザーがパスワードを要求されないようにします。

```
[root@controller ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/stack
[root@controller ~]# chmod 0440 /etc/sudoers.d/stack
```

3. 事前にプロビジョニングされた全ノードで **stack** ユーザーの作成と設定が完了したら、director ノードから各オーバークラウドノードに **stack** ユーザーの公開 SSH 鍵をコピーします。たとえば、director の公開 SSH 鍵をコントローラーノードにコピーするには、以下のコマンドを実行します。

```
[stack@director ~]$ ssh-copy-id stack@192.168.24.2
```

## 8.2. ノードのオペレーティングシステムの登録

ノードごとに Red Hat サブスクリプションへのアクセスが必要です。以下の手順は、各ノードを Red Hat コンテンツ配信ネットワークに登録する方法を説明しています。各ノードで以下の手順を実行してください。

1. 登録コマンドを実行して、プロンプトが表示されたらカスタマーポータルユーザー名とパスワードを入力します。

```
[root@controller ~]# sudo subscription-manager register
```

2. Red Hat OpenStack Platform 13 のエンタイトルメントプールを検索します。

```
[root@controller ~]# sudo subscription-manager list --available --all --matches="Red Hat OpenStack"
```

3. 上記のステップで特定したプール ID を使用して、Red Hat OpenStack Platform 13 のエンタイトルメントをアタッチします。

```
[root@controller ~]# sudo subscription-manager attach --pool=pool_id
```

4. デフォルトのリポジトリをすべて無効にします。

```
[root@controller ~]# sudo subscription-manager repos --disable=*
```

5. 必要な Red Hat Enterprise Linux リポジトリを有効にします。

- a. x86\_64 システムの場合には、以下のコマンドを実行します。

```
[root@controller ~]# sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-13-rpms --enable=rhel-7-server-rhceph-2-osd-rpms --enable=rhel-7-server-rhceph-2-mon-rpms --enable=rhel-7-server-rhceph-2-tools-rpms
```

- b. POWER システムの場合には、以下のコマンドを実行します。

```
[root@controller ~]# sudo subscription-manager repos --enable=rhel-7-for-power-le-rpms --enable=rhel-7-server-openstack-13-for-power-le-rpms
```



### 重要

「[リポジトリの要件](#)」でリストしたリポジトリのみを有効にします。追加のリポジトリを使用すると、パッケージとソフトウェアの競合が発生する場合があります。他のリポジトリは有効にしないでください。

6. システムを更新して、ベースシステムのパッケージを最新の状態にします。

```
[root@controller ~]# sudo yum update -y
[root@controller ~]# sudo reboot
```

このノードをオーバークラウドに使用する準備ができました。

## 8.3. ノードへのユーザーエージェントのインストール

事前にプロビジョニングされたノードはそれぞれ、OpenStack Orchestration (heat) エージェントを使用して director と通信します。各ノード上のエージェントは、director をポーリングして、そのノードに合わせたメタデータを取得します。このメタデータにより、エージェントは各ノードを設定できます。

各ノードでオーケストレーションエージェントの初期パッケージをインストールします。

```
[root@controller ~]# sudo yum -y install python-heat-agent*
```

## 8.4. DIRECTOR への SSL/TLS アクセスの設定

**director** が SSL/TLS を使用する場合は、事前にプロビジョニングされたノードには、**director** の SSL/TLS 証明書の署名に使用する認証局ファイルが必要です。独自の認証局を使用する場合には、各オーバークラウドノード上で以下のステップを実行します。

1. 事前にプロビジョニングされた各ノードの `/etc/pki/ca-trust/source/anchors/` ディレクトリに認証局ファイルをコピーします。
2. 各オーバークラウドノード上で以下のコマンドを実行します。

```
[root@controller ~]# sudo update-ca-trust extract
```

これにより、オーバークラウドノードが **director** のパブリック API に SSL/TLS 経由でアクセスできるようになります。

## 8.5. コントロールプレーンのネットワークの設定

事前にプロビジョニングされたオーバークラウドノードは、標準の HTTP 要求を使用して **director** からメタデータを取得します。これは、オーバークラウドノードでは以下のいずれかに対して L3 アクセスが必要であることを意味します。

- **director** のコントロールプレーンネットワーク。これは、`undercloud.conf` ファイルの `network_cidr` パラメーターで定義されたサブネットです。ノードには、このサブネットへの直接アクセスまたはルーティング可能なアクセスのいずれかが必要です。
- `undercloud.conf` ファイルの `undercloud_public_host` パラメーターとして指定された **director** のパブリック API のエンドポイント。コントロールプレーンへの L3 ルートがない場合や、**director** をポーリングしてメタデータを取得するのに SSL/TLS 通信を使用する場合に、このオプションを利用できます。オーバークラウドがパブリック API エンドポイントを使用するための追加の設定手順については、「[オーバークラウドノードに別のネットワークを使用する方法](#)」を参照してください。

**director** は、コントロールプレーンネットワークを使用して標準のオーバークラウドを管理、設定します。事前にプロビジョニングされたノードを使用したオーバークラウドの場合には、**director** が事前にプロビジョニングされたノードと通信する方法に対応するために、ネットワーク設定を変更する必要があります。

### ネットワーク分離の使用

ネットワークを分離すると、コントロールプレーンなど、固有のネットワークを使用するようにサービスをグループ化できます。『[Advanced Overcloud Customization](#)』ガイドには、ネットワーク分離の方法が複数記載されています。また、コントロールプレーン上のノードに固有の IP アドレスを定義することも可能です。ネットワーク分離や予測可能なノード配置方法の策定に関する詳しい情報は、『[Advanced Overcloud Customizations](#)』ガイドの以下のセクションを参照してください。

- 「[Isolating Networks](#)」
- 「[Controlling Node Placement](#)」



## 注記

ネットワーク分離を使用する場合には、NIC テンプレートに、アンダークラウドのアクセスに使用する NIC を含めないようにしてください。これらのテンプレートにより NIC が再構成され、デプロイメント時に接続性や設定の問題が発生する可能性があります。

## IP アドレスの割り当て

ネットワーク分離を使用しない場合には、単一のコントロールプレーンを使用して全サービスを管理することができます。これには、各ノード上のコントロールプレーンの NIC を手動で設定して、コントロールプレーンネットワークの範囲内の IP アドレスを使用するようにする必要があります。director のプロビジョニングネットワークをコントロールプレーンとして使用する場合には、選択したオーバークラウドの IP アドレスが、プロビジョニング (**dhcp\_start** および **dhcp\_end**) とイントロスペクション (**inspection\_iprange**) の両方の DHCP 範囲外になるようにしてください。

標準のオーバークラウド作成中には、director はプロビジョニング/コントロールプレーンネットワークのオーバークラウドノードに IP アドレスを自動的に割り当てるための **OpenStack Networking (neutron)** ポートを作成します。ただし、これにより、各ノードに手動で設定した IP アドレスとは異なるアドレスを director が割り当ててしまう可能性があります。このような場合には、予測可能な IP アドレス割り当て方法を使用して、director がコントロールプレーン上で事前にプロビジョニングされた IP の割り当てを強制的に使用するようにしてください。

予測可能な IP アドレス割り当て方法の例では、以下のように IP アドレスを割り当てた環境ファイル (**ctlplane-assignments.yaml**) を使用します。

```
resource_registry:
  OS::TripleO::DeployedServer::ControlPlanePort: /usr/share/openstack-tripleo-heat-templates/deployed-server/deployed-neutron-port.yaml

parameter_defaults:
  DeployedServerPortMap:
    controller-ctlplane:
      fixed_ips:
        - ip_address: 192.168.24.2
      subnets:
        - cidr: 24
    compute-ctlplane:
      fixed_ips:
        - ip_address: 192.168.24.3
      subnets:
        - cidr: 24
```

この例では、**OS::TripleO::DeployedServer::ControlPlanePort** リソースはパラメーターセットを director に渡して、事前にプロビジョニングされたノードの IP 割り当てを定義します。**DeployedServerPortMap** パラメーターは、各オーバークラウドノードに対応する IP アドレスおよびサブネット CIDR を定義します。このマッピングは以下を定義します。

1. 割り当ての名前は、**<node\_hostname>-<network>** の形式です (例: **controller-ctlplane**、**compute-ctlplane**)。
2. 以下のパラメーターパターンを使用する IP 割り当て
  - **fixed\_ips/ip\_address**: コントロールプレーンの固定 IP アドレスを定義します。複数の IP アドレスを定義する場合には、複数の **ip\_address** パラメーターを一覧で指定してください。

- **subnets/cidr**: サブネットの CIDR 値を定義します。

本章の後半のステップでは、作成された環境ファイル (**ctlplane-assignments.yaml**) を **openstack overcloud deploy** コマンドの一部として使用します。

## 8.6. オーバークラウドノードに別のネットワークを使用する方法

デフォルトでは、**director** はオーバークラウドのコントロールプレーンとしてプロビジョニングネットワークを使用しますが、このネットワークが分離されて、ルーティング不可能な場合には、ノードは設定中に **director** の内部 API との通信ができません。このような状況では、ノードに別のネットワークを定義して、パブリック API 経由で **director** と通信できるように設定する必要がある場合があります。

このシナリオには、以下のような複数の要件があります。

- オーバークラウドノードは、「[コントロールプレーンのネットワークの設定](#)」からの基本的なネットワーク設定に対応する必要があります。
- パブリック API エンドポイントを使用できるように **director** 上で SSL/TLS を有効化する必要があります。詳しい情報は、「[director の設定パラメーター](#)」と「[付録A SSL/TLS 証明書の設定](#)」を参照してください。
- **director** 向けにアクセス可能な完全修飾ドメイン名 (FQDN) を定義する必要があります。この FQDN は、**director** にルーティング可能な IP アドレスを解決する必要があります。**undercloud.conf** ファイルの **undercloud\_public\_host** パラメーターを使用して、この FQDN を設定します。

本項に記載する例では、主要なシナリオとは異なる IP アドレスの割り当てを使用します。

表8.2 プロビジョニングネットワークの IP 割り当て

ノード名	IP アドレスまたは FQDN
director (内部 API)	192.168.24.1 (プロビジョニングネットワークおよびコントロールプレーン)
director (パブリック API)	10.1.1.1 / director.example.com
オーバークラウドの仮想 IP	192.168.100.1
コントローラー	192.168.100.2
Compute	192.168.100.3

以下の項では、オーバークラウドノードに別のネットワークが必要な場合の追加の設定について説明します。

### オーケストレーションの設定

アンダークラウドの SSL/TLS 通信を有効化している場合には、**director** は、大半のサービスにパブリック API エンドポイントを提供します。ただし、**OpenStack Orchestration (heat)** は、メタデータのデフォルトのプロバイダーとして内部エンドポイントを使用します。そのため、オーバークラウドノード

ドがパブリックエンドポイントの OpenStack Orchestration にアクセスできるように、アンダークラウドを変更する必要があります。この変更には、director 上の Puppet hieradata の変更などが含まれません。

**undercloud.conf** の **hieradata\_override** を使用すると、アンダークラウド設定用に追加で Puppet hieradata を指定することができます。以下の手順を使用して、OpenStack Orchestration に関連する hieradata を変更してください。

1. **hieradata\_override** ファイルをまだ使用していない場合には、新しいファイルを作成します。以下の例では、**/home/stack/hieradata.yaml** にあるファイルを使用します。
2. **/home/stack/hieradata.yaml** に以下の hieradata を追加します。

```
heat_clients_endpoint_type: public
heat::engine::default_deployment_signal_transport: TEMP_URL_SIGNAL
```

これにより、デフォルトの **internal** から **public** にエンドポイントが変更され、TempURL を使用するシグナルの方法が OpenStack Object Storage (swift) から変更されます。

3. **undercloud.conf** で、**hieradata\_override** パラメーターを hieradata ファイルのパスに設定します。

```
hieradata_override = /home/stack/hieradata.yaml
```

4. **openstack overcloud install** コマンドを再度実行して、新規設定オプションを実装します。

これにより、オーケストレーションメタデータが director のパブリック API 上の URL を使用するようになり、切り替えられます。

## IP アドレスの割り当て

IP の割り当て方法は、「[コントロールプレーンのネットワークの設定](#)」と似ていますが、コントロールプレーンはデプロイしたサーバーからルーティング可能ではないので、**DeployedServerPortMap** パラメーターを使用して、コントロールプレーンにアクセスする仮想 IP アドレスなど、選択したオーバークラウドノードのサブネットから IP アドレスを割り当てます。以下は、このネットワークアーキテクチャーに対応するように、「[コントロールプレーンのネットワークの設定](#)」の **ctlplane-assignments.yaml** 環境ファイルを変更したバージョンです。

```
resource_registry:
  OS::TripleO::DeployedServer::ControlPlanePort: /usr/share/openstack-tripleo-heat-templates/deployed-server/deployed-neutron-port.yaml
  OS::TripleO::Network::Ports::ControlPlaneVipPort: /usr/share/openstack-tripleo-heat-templates/deployed-server/deployed-neutron-port.yaml
  OS::TripleO::Network::Ports::RedisVipPort: /usr/share/openstack-tripleo-heat-templates/network/ports/noop.yaml 1

parameter_defaults:
  NeutronPublicInterface: eth1
  EC2MetadataIp: 192.168.100.1 2
  ControlPlaneDefaultRoute: 192.168.100.1
  DeployedServerPortMap:
    control_virtual_ip:
      fixed_ips:
        - ip_address: 192.168.100.1
```

```

subnets:
  - cidr: 24
controller0-ctlplane:
  fixed_ips:
    - ip_address: 192.168.100.2
  subnets:
    - cidr: 24
compute0-ctlplane:
  fixed_ips:
    - ip_address: 192.168.100.3
  subnets:
    - cidr: 24

```

- 1 **RedisVipPort** リソースは、**network/ports/noop.yaml** にマッピングされます。このマッピングは、デフォルトの Redis VIP アドレスがコントロールプレーンから割り当てられていることが理由です。このような場合には、**noop** を使用して、このコントロールプレーンマッピングを無効化します。
- 2 **EC2MetadataIp** と **ControlPlaneDefaultRoute** パラメーターは、コントロールプレーンの仮想 IP アドレスの値に設定されます。デフォルトの NIC 設定テンプレートでは、これらのパラメーターが必須で、デプロイメント中に実行される検証に合格するには、ping 可能な IP アドレスを使用するように設定する必要があります。または、これらのパラメーターが必要ないように NIC 設定をカスタマイズします。

## 8.7. 事前にプロビジョニングされたノードでのオーバークラウドの作成

オーバークラウドのデプロイメントには、「[CLI ツールを使用したオーバークラウドの作成](#)」に記載の標準の CLI の方法を使用します。事前にプロビジョニングされたノードの場合は、デプロイメントコマンドに追加のオプションと、コア Heat テンプレートコレクションからの環境ファイルが必要です。

- **--disable-validations:** 事前にプロビジョニングされたインフラストラクチャーで使用しないサービスに対する基本的な CLI 検証を無効化します。無効化しないと、デプロイメントに失敗します。
- **environments/depoyed-server-environment.yaml:** 事前にプロビジョニングされたインフラストラクチャーを作成、設定するための主要な環境ファイル。この環境ファイルは、**OS::Nova::Server** リソースを **OS::Heat::DeployedServer** リソースに置き換えます。
- **environments/depoyed-server-bootstrap-environment-rhel.yaml:** 事前にプロビジョニングされたサーバー上でブートストラップのスクリプトを実行する環境ファイル。このスクリプトは、追加パッケージをインストールして、オーバークラウドノードの基本設定を提供します。
- **environments/depoyed-server-pacemaker-environment.yaml:** 事前にプロビジョニングされたコントローラーノードで Pacemaker の設定を行う環境ファイル。このファイルに登録されるリソースの名前空間は、**depoyed-server/depoyed-server-roles-data.yaml** からのコントローラーのロール名を使用します。デフォルトでは、**ControllerDeployedServer** となっています。
- **depoyed-server/depoyed-server-roles-data.yaml:** カスタムロールのサンプルファイル。これは、デフォルトの **roles\_data.yaml** が複製されたファイルですが、各ロールの **disable\_constraints: True** パラメーターも含まれています。このパラメーターは、

生成されたロールテンプレートのオーケストレーションの制約を無効にします。これらの制約は、事前にプロビジョニングされたインフラストラクチャーで使用しないサービスが対象です。

独自のカスタムロールファイルを使用する場合には、各ロールに **disable\_constraints: True** パラメーターを追加するようにしてください。以下に例を示します。

```
- name: ControllerDeployedServer
  disable_constraints: True
  CountDefault: 1
  ServicesDefault:
    - OS::TripleO::Services::CACerts
    - OS::TripleO::Services::CephMon
    - OS::TripleO::Services::CephExternal
    - OS::TripleO::Services::CephRgw
    ...
```

以下は、事前にプロビジョニングされたアーキテクチャー固有の環境ファイルを使用したオーバークラウドデプロイメントのコマンド例です。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy \
  [other arguments] \
  --disable-validations \
  -e /usr/share/openstack-tripleo-heat-templates/environments/deployed-
server-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/deployed-
server-bootstrap-environment-rhel.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/deployed-
server-pacemaker-environment.yaml \
  -r /usr/share/openstack-tripleo-heat-templates/deployed-server/deployed-
server-roles-data.yaml
```

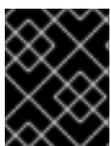
これにより、オーバークラウドの設定が開始されますが、デプロイメントのスタックは、オーバークラウドのノードリソースが **CREATE\_IN\_PROGRESS** の段階に入ると一時停止します。

```
2017-01-14 13:25:13Z [overcloud.Compute.0.Compute]: CREATE_IN_PROGRESS
state changed
2017-01-14 13:25:14Z [overcloud.Controller.0.Controller]:
CREATE_IN_PROGRESS state changed
```

このように一時停止されるのは、オーバークラウドノード上のオーケストレーションエージェントがメタデータサーバーをポーリングするのを **director** が待っているためです。次のセクションでは、メタデータサーバーのポーリングを開始するようにノードを設定する方法を説明します。

## 8.8. メタデータサーバーのポーリング

デプロイメントは進行中ですが、**CREATE\_IN\_PROGRESS** の段階で一時停止されます。次のステップでは、オーバークラウドノードのオーケストレーションエージェントが **director** 上のメタデータサーバーをポーリングするように設定します。この操作には、2つの方法があります。



### 重要

初期のデプロイメントの場合のみに自動設定を使用します。ノードをスケールアップする場合には自動設定を使用しないでください。

## 自動設定

`director` のコア Heat テンプレートコレクションには、オーバークラウドノード上で Heat エージェントの自動設定を行うスクリプトが含まれます。このスクリプトで、`director` との認証を行ってオーケストレーションサービスに対してクエリを実行するには、`stack` ユーザーとして `stackrc` ファイルを読み込む必要があります。

```
[stack@director ~]$ source ~/stackrc
```

また、このスクリプトでは、追加の環境変数でノードのロールやその IP アドレスを定義する必要があります。これらの環境変数は以下のとおりです。

### OVERCLOUD\_ROLES

設定するロールのスペース区切りの一覧。これらのロールは、ロールデータファイルで定義したロールに related します。

### [ROLE]\_hosts

ロールごとに、環境変数と、ロールに含まれるノードの IP アドレス (スペース区切りの一覧) が必要です。

以下のコマンドは、これらの環境変数の設定例です。

```
(undercloud) $ export OVERCLOUD_ROLES="ControllerDeployedServer
ComputeDeployedServer"
(undercloud) $ export ControllerDeployedServer_hosts="192.168.100.2"
(undercloud) $ export ComputeDeployedServer_hosts="192.168.100.3"
```

スクリプトを実行して、各オーバークラウドノード上にオーケストレーションエージェントを設定します。

```
(undercloud) $ /usr/share/openstack-tripleo-heat-templates/deployed-
server/scripts/get-occ-config.sh
```



### 注記

このスクリプトは、スクリプトを実行する同じユーザーを使用して SSH 経由で事前にプロビジョニングされたノードにアクセスします。今回の場合は、スクリプトは、`stack` ユーザーの認証を行います。

このスクリプトは、以下を行います。

- 各ノードのメタデータ URL を確認するために `director` のオーケストレーションサービスにクエリを実行します。
- ノードにアクセスして、固有のメタデータ URL で各ノードのエージェントを設定します。
- オーケストレーションエージェントサービスを再起動します。

スクリプトが完了したら、オーバークラウドノードは `director` 上でオーケストレーションサービスのポーリングを開始します。スタックのデプロイメントが継続されます。

## 手動による設定

事前にプロビジョニングされたノードでオーケストレーションエージェントを手動で設定する場合には、以下のコマンドを使用して、各ノードの URL に関して **director** 上のオーケストレーションサービスにクエリーを実行します。

```
[stack@director ~]$ source ~/stackrc
(undercloud) $ for STACK in $(openstack stack resource list -n5 --filter
name=deployed-server -c stack_name -f value overcloud) ; do STACKID=$(echo
$STACK | cut -d '-' -f2,4 --output-delimiter " "); echo "== Metadata URL
for $STACKID ==" ; openstack stack resource metadata $STACK deployed-
server | jq -r '["os-collect-config"].request.metadata_url' ; echo ; done
```

これにより、各ノードのスタック名やメタデータの URL が表示されます。

```
== Metadata URL for ControllerDeployedServer 0 ==
http://192.168.24.1:8080/v1/AUTH_6fce4e6019264a5b8283e7125f05b764/ov-
edServer-ts6lr4tm5p44-deployed-server-td42md2tap4g/43d302fa-d4c2-40df-
b3ac-624d6075ef27?
temp_url_sig=58313e577a93de8f8d2367f8ce92dd7be7aac3a1&temp_url_expires=214
7483586

== Metadata URL for ComputeDeployedServer 0 ==
http://192.168.24.1:8080/v1/AUTH_6fce4e6019264a5b8283e7125f05b764/ov-
edServer-wdpk7upmz3eh-deployed-server-ghv7ptfikz2j/0a43e94b-fe02-427b-
9bfe-71d2b7bb3126?
temp_url_sig=8a50d8ed6502969f0063e79bb32592f4203a136e&temp_url_expires=214
7483586
```

各オーバークラウドノード上で以下を行います。

1. 既存の **os-collect-config.conf** テンプレートを削除して、エージェントによって手動の変更が上書きされないようにします。

```
$ sudo /bin/rm -f /usr/libexec/os-apply-config/templates/etc/os-
collect-config.conf
```

2. **/etc/os-collect-config.conf** ファイルを対応するメタデータ URL を使用するよう設定します。たとえば、コントローラーノードは以下を使用します。

```
[DEFAULT]
collectors=request
command=os-refresh-config
polling_interval=30

[request]
metadata_url=http://192.168.24.1:8080/v1/AUTH_6fce4e6019264a5b8283e7
125f05b764/ov-edServer-ts6lr4tm5p44-deployed-server-
td42md2tap4g/43d302fa-d4c2-40df-b3ac-624d6075ef27?
temp_url_sig=58313e577a93de8f8d2367f8ce92dd7be7aac3a1&temp_url_expir
es=2147483586
```

3. ファイルを保存します。
4. **os-collect-config** サービスを再起動します。

```
[stack@controller ~]$ sudo systemctl restart os-collect-config
```

サービスを設定して再起動した後に、オーケストレーションエージェントは **director** のオーケストレーションサービスをポーリングしてオーバークラウドの設定を行います。デプロイメントスタックは作成を続行して、各ノードのスタックは最終的に **CREATE\_COMPLETE** に変わります。

## 8.9. オーバークラウド作成の監視

オーバークラウドの設定プロセスが開始されます。このプロセスは完了するまで多少時間がかかります。オーバークラウドの作成のステータスを確認するには、**stack** ユーザーとして別のターミナルを開き、以下のコマンドを実行します。

```
[stack@director ~]$ source ~/stackrc
(undercloud) $ heat stack-list --show-nested
```

**heat stack-list --show-nested** コマンドは、オーバークラウド作成の現在の段階を表示します。

## 8.10. オーバークラウドへのアクセス

**director** は、**director** ホストからオーバークラウドに対話するための設定を行い、認証をサポートするスクリプトを作成して、**stack** ユーザーのホームディレクトリーにこのファイル (**overcloudrc**) を保存します。このファイルを使用するには、以下のコマンドを実行します。

```
(undercloud) $ source ~/overcloudrc
```

これで、**director** のホストの CLI からオーバークラウドと対話するために必要な環境変数が読み込まれます。コマンドプロンプトが変わり、オーバークラウドと対話していることが示されます。

```
(overcloud) $
```

**director** のホストとの対話に戻るには、以下のコマンドを実行します。

```
(overcloud) $ source ~/stackrc
(undercloud) $
```

## 8.11. 事前にプロビジョニングされたノードのスケールアップ

事前にプロビジョニングされたノードをスケールアップするプロセスは、「[11章 オーバークラウドのスケールアップ](#)」に記載の標準のスケールアップの手順と似ていますが、事前にプロビジョニングされたノードを新たに追加するプロセスは異なります。これは、事前にプロビジョニングされたノードが **OpenStack Bare Metal (ironic)** および **OpenStack Compute (nova)** からの標準の登録および管理プロセスを使用しないためです。

### 事前にプロビジョニングされたノードのスケールアップ

事前にプロビジョニングされたノードでオーバークラウドをスケールアップする際には、各ノードで **director** のノード数に対応するようにオーケストレーションエージェントを設定する必要があります。

ノードのスケールアップの大まかなプロセスは以下のとおりです。

1. 「要件」の説明に従って、事前にプロビジョニングされたノードを準備します。

2. ノードをスケールアップします。手順については「[11章 オーバークラウドのスケールリング](#)」を参照してください。
3. デプロイメントコマンドを実行した後に、director が新しいノードリソースを作成するまで待ちます。「[メタデータサーバーのポーリング](#)」の手順に従って、事前にプロビジョニングされたノードが director のオーケストレーションサーバーのメタデータ URL をポーリングするように設定します。

### 事前にプロビジョニングされたノードのスケールダウン

事前にプロビジョニングされたノードでオーバークラウドをスケールダウンするには、「[11章 オーバークラウドのスケールリング](#)」に記載の通常のスケールダウンの手順に従います。

スタックからオーバークラウドノードを削除したら、それらのノードの電源をオフにします。標準のデプロイメントでは、director のベアメタルサービスがこの機能を制御しますが、事前にプロビジョニングされたノードでは、これらのノードを手動でシャットダウンするか、物理システムごとに電源管理制御を使用します。スタックからノードを削除した後にノードの電源をオフにしないと、稼動状態が続き、オーバークラウド環境の一部として再接続されてしまう可能性があります。

削除したノードの電源をオフにした後には、再プロビジョニングしてベースのオペレーティングシステムの設定に戻し、それらのノードが意図せずにオーバークラウドに加わってしまうことがないようにします。



#### 注記

オーバークラウドから以前に削除したノードは、再プロビジョニングしてベースオペレーティングシステムを新規インストールしてからでなければ、再利用しないようにしてください。スケールダウンのプロセスでは、オーバークラウドスタックからノードを削除するだけで、パッケージはアンインストールされません。

## 8.12. 事前にプロビジョニングされたオーバークラウドの削除

標準のオーバークラウドと同じ手順で、事前にプロビジョニングされたノードを使用するオーバークラウド全体を削除します。詳しい情報は、「[オーバークラウドの削除](#)」を参照してください。

オーバークラウドの削除後には、全ノードの電源をオフにしてから再プロビジョニングして、ベースオペレーティングシステムの設定に戻します。



#### 注記

オーバークラウドから削除したノードは、再プロビジョニングしてベースオペレーティングシステムを新規インストールしてからでなければ再利用しないでください。削除のプロセスでは、オーバークラウドスタックを削除するだけで、パッケージはアンインストールされません。

## 8.13. オーバークラウド作成の完了

これで、事前にプロビジョニングされたノードを使用したオーバークラウドの作成が完了しました。作成後の機能については、「[9章 オーバークラウド作成後のタスクの実行](#)」を参照してください。

## 第9章 オーバークラウド作成後のタスクの実行

本章では、任意のオーバークラウドを作成後に実行するタスクについて考察します。

### 9.1. コンテナ化されたサービスの管理

オーバークラウドでは、**OpenStack Platform** サービスの大半をコンテナ内で実行します。特定の状況では、1つのホスト上で個別のサービスを制御する必要がある場合があります。本項には、オーバークラウドノード上で、コンテナ化されたサービスを管理するために実行することのできる一般的な **docker** コマンドについて記載します。**docker** を使用したコンテナ管理に関する包括的な情報は、『**コンテナの使用ガイド**』の「**Docker フォーマットのコンテナイメージの使用方法**」を参照してください。



#### 注記

これらのコマンドを実行する前には、オーバークラウドノードにログイン済みであることを確認し、これらのコマンドをアンダークラウドで実行しないようにしてください。

#### コンテナとイメージの一覧表示

実行中のコンテナを一覧表示するには、以下のコマンドを実行します。

```
$ sudo docker ps
```

停止中またはエラーの発生したコンテナも一覧表示するには、コマンドに **--all** オプションを追加します。

```
$ sudo docker ps --all
```

コンテナイメージを一覧表示するには、以下のコマンドを実行します。

```
$ sudo docker images
```

#### コンテナのプロパティの確認

コンテナまたはコンテナイメージのプロパティを確認するには、**docker inspect** コマンドを使用します。たとえば、**keystone** コンテナを確認するには、以下のコマンドを実行します。

```
$ sudo docker inspect keystone
```

#### 基本的なコンテナ操作の管理

コンテナ化されたサービスを再起動するには、**docker restart** コマンドを使用します。たとえば、**keystone** コンテナを再起動するには、以下のコマンドを実行します。

```
$ sudo docker restart keystone
```

コンテナ化されたサービスを停止するには、**docker stop** コマンドを使用します。たとえば、**keystone** のコンテナを停止するには、以下のコマンドを実行します。

```
$ sudo docker stop keystone
```

停止されているコンテナ化されたサービスを起動するには、**docker start** コマンドを使用します。たとえば、**keystone** のコンテナを起動するには、以下のコマンドを実行します。

```
$ sudo docker start keystone
```



### 注記

コンテナ内のサービス設定ファイルに加えた変更は、コンテナの再起動後には元に戻ります。これは、コンテナがノードのローカルファイルシステム上の **/var/lib/config-data/puppet-generated/** にあるファイルに基づいてサービス設定を再生成するためです。たとえば、**keystone** コンテナ内の **/etc/keystone/keystone.conf** を編集してコンテナを再起動すると、そのコンテナはノードのローカルシステム上にある **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** を使用して設定を再生成します。再起動前にコンテナ内で加えられた変更は、この設定によって上書きされません。

### コンテナのモニター

コンテナ化されたサービスのログを確認するには、**docker logs** コマンドを使用します。たとえば、**keystone** のログを確認するには、以下のコマンドを実行します。

```
$ sudo docker logs keystone
```

### コンテナへのアクセス

コンテナ化されたサービスのシェルに入るには、**docker exec** コマンドを使用して **/bin/bash** を起動します。たとえば、**keystone** コンテナのシェルに入るには、以下のコマンドを実行します。

```
$ sudo docker exec -it keystone /bin/bash
```

**keystone** コンテナのシェルに **root** ユーザーとして入るには、以下のコマンドを実行します。

```
$ sudo docker exec --user 0 -it <NAME OR ID> /bin/bash
```

コンテナから出るには、以下のコマンドを実行します。

```
# exit
```

OpenStack Platform のコンテナ化されたサービスのトラブルシューティングに関する情報は、「[コンテナ化されたサービスのエラー](#)」を参照してください。

## 9.2. オーバークラウドのテナントネットワークの作成

オーバークラウドには、インスタンス用のテナントネットワークが必要です。**source** コマンドで **overcloud** を読み込んで、**Neutron** で初期テナントネットワークを作成します。以下に例を示します。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create default
(overcloud) $ openstack subnet create default --network default --gateway
172.20.1.1 --subnet-range 172.20.0.0/16
```

上記のステップにより、**default** という名前の基本的な **Neutron** ネットワークが作成されます。オーバークラウドは、内部 **DHCP** メカニズムを使用したこのネットワークから、**IP** アドレスを自動的に割り当てます。

作成したネットワークを確認します。

```
(overcloud) $ openstack network list
+-----+-----+-----+
| id                | name          | subnets    |
+-----+-----+-----+
| 95fadaa1-5dda-4777... | default      | 7e060813-35c5-462c-a56a-1c6f8f4f332f |
+-----+-----+-----+
```

### 9.3. オーバークラウドの外部ネットワークの作成

インスタンスに **Floating IP** を割り当てることができるように、オーバークラウドで外部ネットワークを作成する必要があります。

#### ネイティブ **VLAN** の使用

以下の手順では、外部ネットワーク向けの専用インターフェースまたはネイティブの **VLAN** が設定されていることが前提です。

**source** コマンドで **overcloud** を読み込み、**Neutron** で外部ネットワークを作成します。以下に例を示します。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create public --external --provider-network-type flat --provider-physical-network datacentre
(overcloud) $ openstack subnet create public --network public --dhcp --allocation-pool start=10.1.1.51,end=10.1.1.250 --gateway 10.1.1.1 --subnet-range 10.1.1.0/24
```

以下の例では、**public** という名前のネットワークを作成します。オーバークラウドには、デフォルトの **Floating IP** プールにこの特定の名前が必要です。このネットワークは、「[オーバークラウドの検証](#)」の検証テストでも重要となります。

このコマンドにより、ネットワークと **datacentre** の物理ネットワークのマッピングも行われます。デフォルトでは、**datacentre** は **br-ex** ブリッジにマッピングされます。オーバークラウドの作成時にカスタムの **Neutron** の設定を使用していない限りは、このオプションはデフォルトのままにしてください。

#### 非ネイティブ **VLAN** の使用

ネイティブ **VLAN** を使用しない場合には、以下のコマンドでネットワークを **VLAN** に割り当てます。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create public --external --provider-network-type vlan --provider-physical-network datacentre --provider-segment 104
```

```
(overcloud) $ openstack subnet create public --network public --dhcp --
allocation-pool start=10.1.1.51,end=10.1.1.250 --gateway 10.1.1.1 --
subnet-range 10.1.1.0/24
```

`provider:segmentation_id` の値は、使用する VLAN を定義します。この場合は、104 を使用します。

作成したネットワークを確認します。

```
(overcloud) $ openstack network list
+-----+-----+-----+
-----+
| id                | name          | subnets
|
+-----+-----+-----+
-----+
| d474fe1f-222d-4e32... | public        | 01c5f621-1e0f-4b9d-9c30-
7dc59592a52f |
+-----+-----+-----+
-----+
```

## 9.4. 追加の FLOATING IP ネットワークの作成

Floating IP ネットワークは、以下の条件を満たす限りは、`br-ex` だけでなく、どのブリッジにも使用することができます。

- ネットワーク環境ファイルで、`NeutronExternalNetworkBridge` が `''` に設定されている。
- デプロイ中に追加のブリッジをマッピングしている。たとえば、`br-floating` という新規ブリッジを `floating` という物理ネットワークにマッピングするには、環境ファイルで以下の設定を使用します。

```
parameter_defaults:
  NeutronBridgeMappings: "datacentre:br-ex,floating:br-floating"
```

オーバークラウドの作成後に Floating IP ネットワークを作成します。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create ext-net --external --provider-
physical-network floating --provider-network-type vlan --provider-segment
105
(overcloud) $ openstack subnet create ext-subnet --network ext-net --dhcp
--allocation-pool start=10.1.2.51,end=10.1.2.250 --gateway 10.1.2.1 --
subnet-range 10.1.2.0/24
```

## 9.5. オーバークラウドのプロバイダーネットワークの作成

プロバイダーネットワークは、デプロイしたオーバークラウドの外部に存在するネットワークに物理的に接続されたネットワークです。これは、既存のインフラストラクチャーネットワークや、Floating IP の代わりにルーティングによって直接インスタンスに外部アクセスを提供するネットワークを使用することができます。

プロバイダーネットワークを作成する際には、ブリッジマッピングを使用する物理ネットワークに関連付けます。これは、Floating IP ネットワークの作成と同様です。コンピュータノードは、仮想マシンの仮想ネットワークインターフェースをアタッチされているネットワークインターフェースに直接接続するため、プロバイダーネットワークはコントローラーとコンピュータの両ノードに追加します。

たとえば、使用するプロバイダーネットワークが **br-ex** ブリッジ上の **VLAN** の場合には、以下のコマンドを使用してプロバイダーネットワークを **VLAN 201** 上に追加します。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create provider_network --provider-physical-network datacentre --provider-network-type vlan --provider-segment 201 --share
```

このコマンドにより、共有ネットワークが作成されます。また、**--share** と指定する代わりにテナントを指定することも可能です。そのネットワークは、指定されたテナントに対してのみ提供されます。プロバイダーネットワークを外部としてマークした場合には、そのネットワークでポートを作成できるのはオペレーターのみとなります。

**Neutron** が **DHCP** サービスをテナントのインスタンスに提供するように設定するには、プロバイダーネットワークにサブネットを追加します。

```
(overcloud) $ openstack subnet create provider-subnet --network provider_network --dhcp --allocation-pool start=10.9.101.50,end=10.9.101.100 --gateway 10.9.101.254 --subnet-range 10.9.101.0/24
```

他のネットワークがプロバイダーネットワークを介して外部にアクセスする必要がある場合があります。このような場合には、新規ルーターを作成して、他のネットワークがプロバイダーネットワークを介してトラフィックをルーティングできるようにします。

```
(overcloud) $ openstack router create external
(overcloud) $ openstack router set --external-gateway provider_network external
```

このルーターに他のネットワークを接続します。たとえば、**subnet1** という名前のサブネットがある場合には、以下のコマンドを実行してルーターに接続することができます。

```
(overcloud) $ openstack router add subnet external subnet1
```

これにより、**subnet1** がルーティングテーブルに追加され、**subnet1** を使用するトラフィックをプロバイダーネットワークにルーティングできるようになります。

## 9.6. オーバークラウドの検証

オーバークラウドは、**OpenStack Integration Test Suite (tempest)** ツールセットを使用して、一連の統合テストを行います。本項には、統合テストの実行準備に関する情報を記載します。**OpenStack Integration Test Suite** の使用方法に関する詳しい説明は、『[OpenStack Integration Test Suite Guide](#)』を参照してください。

### Integration Test Suite の実行前

アンダークラウドからこのテストを実行する場合は、アンダークラウドのホストがオーバークラウドの内部 API ネットワークにアクセスできるようにします。たとえば、**172.16.0.201/24** のアドレスを使用して内部 API ネットワーク (**ID: 201**) にアクセスするにはアンダークラウドホストに一時的な **VLAN** を

追加します。

```
$ source ~/stackrc
(undercloud) $ sudo ovs-vsctl add-port br-ctlplane vlan201 tag=201 -- set
interface vlan201 type=internal
(undercloud) $ sudo ip l set dev vlan201 up; sudo ip addr add
172.16.0.201/24 dev vlan201
```

OpenStack Integration Test Suite を実行する前に、**heat\_stack\_owner** ロールがオーバークラウドに存在することを確認してください。

```
$ source ~/overcloudrc
(overcloud) $ openstack role list
+-----+-----+
| ID | Name |
+-----+-----+
| 6226a517204846d1a26d15aae1af208f | swiftoperator |
| 7c7eb03955e545dd86bbfeb73692738b | heat_stack_owner |
+-----+-----+
```

このロールが存在しない場合は、作成します。

```
(overcloud) $ openstack role create heat_stack_owner
```

## Integration Test Suite の実行後

検証が完了したら、オーバークラウドの内部 API への一時接続を削除します。この例では、以下のコマンドを使用して、以前にアンダークラウドで作成した VLAN を削除します。

```
$ source ~/stackrc
(undercloud) $ sudo ovs-vsctl del-port vlan201
```

## 9.7. オーバークラウド環境の変更

オーバークラウドを変更して、別機能を追加したり、操作の方法を変更したりする場合があります。オーバークラウドを変更するには、カスタムの環境ファイルと Heat テンプレートに変更を加えて、最初に作成したオーバークラウドから **openstack overcloud deploy** コマンドをもう 1 度実行します。たとえば、「[CLI ツールを使用したオーバークラウドの作成](#)」の方法を使用してオーバークラウドを作成した場合には、以下のコマンドを再度実行します。

```
$ source ~/stackrc
(undercloud) $ 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 ~/templates/storage-environment.yaml \
--ntp-server pool.ntp.org
```

**director** は Heat 内の **overcloud** スタックを確認してから、環境ファイルと Heat テンプレートのあるスタックで各アイテムを更新します。オーバークラウドは再度作成されずに、既存のオーバークラウドに変更が加えられます。

新規環境ファイルを追加する場合には、**openstack overcloud deploy** コマンドで **-e** オプションを使用してそのファイルを追加します。以下に例を示します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates \
  -e ~/templates/new-environment.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-
isolation.yaml \
  -e ~/templates/network-environment.yaml \
  -e ~/templates/storage-environment.yaml \
  -e ~/templates/node-info.yaml \
  --ntp-server pool.ntp.org
```

これにより、環境ファイルからの新規パラメーターやリソースがスタックに追加されます。



### 重要

**director** により後で上書きされてしまう可能性があるため、オーバークラウドの設定には手動で変更を加えないことを推奨します。

## 9.8. 動的インベントリースクリプトの実行

**director** を使用すると、**Ansible** ベースの自動化を **OpenStack Platform** 環境で実行することができます。**director** は、**tripleo-ansible-inventory** コマンドを使用して、環境内にノードの動的インベントリーを生成します。

### 手順

1. ノードの動的インベントリーを表示するには、**stackrc** を読み込んだ後に **tripleo-ansible-inventory** コマンドを実行します。

```
$ source ~/stackrc
(undercloud) $ tripleo-ansible-inventory --list
```

**--list** オプションを指定すると、全ホストの詳細が表示されます。これにより、動的インベントリーが **JSON** 形式で出力されます。

```
{"overcloud": {"children": ["controller", "compute"], "vars":
{"ansible_ssh_user": "heat-admin"}}, "controller": ["192.168.24.2"],
"undercloud": {"hosts": ["localhost"], "vars":
{"overcloud_horizon_url": "http://192.168.24.4:80/dashboard",
"overcloud_admin_password": "abcdefghijklm12345678",
"ansible_connection": "local"}}, "compute": ["192.168.24.3"]}
```

2. お使いの環境で **Ansible** のプレイブックを実行するには、**ansible** コマンドを実行し、**-i** オプションを使用して動的インベントリーツールの完全パスを追加します。以下に例を示します。

```
(undercloud) $ ansible [HOSTS] -i /bin/tripleo-ansible-inventory
[OTHER OPTIONS]
```

- **[HOSTS]** は使用するホストの種別に置き換えます。以下に例を示します。

- 全コントローラーノードの場合には **controller**
- 全コンピューターノードの場合には **compute**
- **controller** および **compute** など、全オーバークラウドの子ノードの場合には **overcloud**
- アンダークラウドの場合には **undercloud**
- 全ノードの場合には **"\*"**
- **[OTHER OPTIONS]** は追加の Ansible オプションに置き換えてください。役立つオプションには以下が含まれます。
  - **--ssh-extra-args='-o StrictHostKeyChecking=no'** は、ホストキーのチェックを省略します。
  - **-u [USER]** は、Ansible の自動化を実行する SSH ユーザーを変更します。オーバークラウドのデフォルトの SSH ユーザーは、動的インベントリーの **ansible\_ssh\_user** パラメーターで自動的に定義されます。**-u** オプションは、このパラメーターより優先されます。
  - **-m [MODULE]** は、特定の Ansible モジュールを使用します。デフォルトは **command** で Linux コマンドを実行します。
  - **-a [MODULE\_ARGS]** は選択したモジュールの引数を定義します。



### 重要

オーバークラウドの Ansible 自動化は、標準のオーバークラウドスタックとは異なります。つまり、この後に **openstack overcloud deploy** コマンドを実行すると、オーバークラウドノード上の OpenStack Platform サービスに対する Ansible ベースの設定を上書きする可能性があります。

## 9.9. オーバークラウドへの仮想マシンのインポート

既存の OpenStack 環境があり、仮想マシンを Red Hat OpenStack Platform 環境に移行する予定がある場合には、以下の手順を使用します。

実行中のサーバーのスナップショットを作成して新規イメージを作成し、そのイメージをダウンロードします。

```
$ source ~/overcloudrc
(overcloud) $ openstack server image create instance_name --name
image_name
(overcloud) $ openstack image save image_name --file exported_vm.qcow2
```

エクスポートしたイメージをオーバークラウドにアップロードして、新しいインスタンスを起動します。

```
(overcloud) $ openstack image create imported_image --file
exported_vm.qcow2 --disk-format qcow2 --container-format bare
(overcloud) $ openstack server create imported_instance --key-name
default --flavor m1.demo --image imported_image --nic net-id=net_id
```



## 重要

各仮想マシンのディスクは、既存の **OpenStack** 環境から新規の **Red Hat OpenStack Platform** にコピーする必要があります。QCOW を使用したスナップショットでは、元の階層化システムが失われます。

## 9.10. コンピュートノードからのインスタンスの移行

オーバークラウドのコンピュートノードでメンテナンスを行う場合があります。ダウンタイムを防ぐには、そのコンピュートノード上の仮想マシンを同じオーバークラウド内の別のコンピュートノードに移行します。

**director** は、すべてのコンピュートノードがセキュアな移行を提供するように設定します。全コンピュートノードには、各ホストの **nova** ユーザーが移行プロセス中に他のコンピュートノードにアクセスすることができるようにするための共有 **SSH** キーも必要です。**director** は、**OS::TripleO::Services::NovaCompute** コンポーザブルサービスを使用してこのキーを作成します。このコンポーザブルサービスは、全コンピュートロールにデフォルトで含まれているメインのサービスの1つです (『**Edit Role Parameters**』の「**Composable Services and Custom Roles**」を参照)。

### 手順

1. アンダークラウドから、コンピュートノードを選択し、そのノードを無効にします。

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set [hostname] nova-compute --disable
```

2. コンピュートノード上の全インスタンスを一覧表示します。

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

3. 以下のコマンドの1つを使用して、インスタンスを移行します。

- a. 選択した特定のホストにインスタンスを移行します。

```
(overcloud) $ openstack server migrate [instance-id] --live [target-host] --wait
```

- b. **nova-scheduler** により対象のホストが自動的に選択されるようにします。

```
(overcloud) $ nova live-migration [instance-id]
```

- c. 一度にすべてのインスタンスのライブマイグレーションを行います。

```
$ nova host-evacuate-live [hostname]
```



### 注記

**nova** コマンドで非推奨の警告が表示される可能性がありますが、安全に無視することができます。

4. 移行が完了するまで待ちます。
5. 正常に移行したことを確認します。

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

6. 選択したコンピューターノードのインスタンスがなくなるまで、移行を続けます。

これにより、コンピューターノードからすべてのインスタンスが移行されます。インスタンスのダウンタイムなしにノードでメンテナンスを実行できるようになります。コンピューターノードを有効な状態に戻すには、以下のコマンドを実行します。

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set [hostname] nova-compute --enable
```

## 9.11. オーバークラウドの削除防止

**heat stack-delete overcloud** コマンドで誤って削除されないように、Heat には特定のアクションを制限するポリシーセットが含まれます。/etc/heat/policy.json を開いて、以下のパラメーターを検索します。

```
"stacks:delete": "rule:deny_stack_user"
```

このパラメーターの設定を以下のように変更します。

```
"stacks:delete": "rule:deny_everybody"
```

ファイルを保存します。

これにより **heat** クライアントでオーバークラウドが削除されないように阻止されます。オーバークラウドを削除できるように設定するには、ポリシーを元の値に戻します。

## 9.12. オーバークラウドの削除

オーバークラウドはすべて、必要に応じて削除することができます。

既存のオーバークラウドを削除します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud delete overcloud
```

オーバークラウドが削除されていることを確認します。

```
(undercloud) $ openstack stack list
```

削除には、数分かかります。

削除が完了したら、デプロイメントシナリオの標準ステップに従って、オーバークラウドを再度作成します。

## 9.13. トークンのフラッシュ間隔の確認

Identity Service (keystone) は、他の OpenStack サービスに対するアクセス制御にトークンベースのシステムを使用します。ある程度の期間が過ぎた後には、データベースに未使用のトークンが多数蓄積されます。デフォルトの cronjob は毎日トークンをフラッシュします。環境をモニタリングして、必要に応じてトークンのフラッシュ間隔を調節することを推奨します。

オーバークラウドでは、**KeystoneCronToken** の値を使用して間隔を調整することができます。詳しい情報は、『[Overcloud Parameters](#)』ガイドを参照してください。

## 第10章 ANSIBLE を使用したオーバークラウドの設定



### 重要

この機能は、本リリースではテクノロジープレビューとして提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー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。

オーバークラウドの設定を適用する主要な方法として Ansible を使用することが可能です。本章では、オーバークラウドでこの機能を有効化する手順を説明します。

director は Ansible Playbook を自動生成しますが、Ansible の構文を十分に理解しておくが役立ちます。Ansible の使用方法については、<https://docs.ansible.com/> を参照してください。



### 注記

Ansible では、roles の概念も使用します。これは、OpenStack Platform director のロールとは異なります。

## 10.1. ANSIBLE ベースのオーバークラウド設定 (CONFIG-DOWNLOAD)

### config-download 機能

- Heat の代わりに Ansible を使用して、オーバークラウドの設定の適用を有効化します。
- オーバークラウドノード上の Heat と Heat エージェント (**os-collect-config**) の間の設定デプロイメントデータの通信と転送を置き換えます。

Heat は、**config-download** を有効化する場合またはしない場合も、標準の機能を維持します。

- director は環境ファイルとパラメーターを Heat に渡します。
- director は Heat を使用してスタックとすべての子リソースを作成します。
- ベアメタルノード、ネットワークなどの OpenStack サービスリソースはいずれも Heat が引き続き作成します。

Heat は **SoftwareDeployment** リソースから全デプロイメントデータを作成して、オーバークラウドのインストールと設定を行います。設定の適用は一切行いません。その代わりに、Heat は API からデータの提供のみを行います。スタックが作成されたら、Mistral ワークフローが Heat API に対してデプロイメントデータのクエリーを実行して、Ansible インベントリーファイルと生成された Playbook を使用して **ansible-playbook** を実行します。

## 10.2. オーバークラウドの設定メソッドを CONFIG-DOWNLOAD に切り替える手順

以下の手順では、オーバークラウドの設定メソッドを OpenStack Orchestration (heat) から Ansible ベース **config-download** のメソッドに切り替えます。このような状況では、アンダークラウドは Ansible の control node (**ansible-playbook** を実行するノード) としての機能を果たします。control node とアンダークラウドという用語は、アンダークラウドのインストールが実行されるのと同じノードを指します。

## 手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. **--config-download** オプションと **heat** ベースの設定を無効にする環境ファイルを指定してオーバークラウドのデプロイメントのコマンドを実行します。

```
$ openstack overcloud deploy --templates \
  --config-download \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/config-download-environment.yaml \
  --overcloud-ssh-user heat-admin \
  --overcloud-ssh-key ~/.ssh/id_rsa \
  [OTHER OPTIONS]
```

以下のオプションの用途に注意してください。

- **--config-download** により、追加の Mistral ワークフローが有効化され、Heat の代わりに **ansible-playbook** で設定が適用されるようになります。
- **-e /usr/share/openstack-tripleo-heat-templates/environments/config-download-environment.yaml** は、Heat のソフトウェアデプロイメント設定リソースを Ansible ベースの同等のリソースにマッピングするための必須の環境ファイルです。これにより、Heat が設定を適用するのではなく、Heat API を介して設定データが提供されます。
- **--overcloud-ssh-user** および **--overcloud-ssh-key** は、各オーバークラウドノードに SSH 接続して、初期 **tripleo-admin** ユーザーを作成し、SSH キーを **/home/tripleo-admin/.ssh/authorized\_keys** に挿入するのに使用します。SSH キーを挿入するには、初回の SSH 接続で **--overcloud-ssh-user** (**heat-admin** がデフォルト) と **--overcloud-ssh-key** (**~/.ssh/id\_rsa** がデフォルト) を使用して認証情報を指定します。**--overcloud-ssh-key** で指定した秘密鍵の公開を制限するために、**director** は Heat や Mistral などのどの API サービスにもこの鍵を渡さず、「**openstack overcloud deploy**」コマンドのみがこの鍵を使用して **tripleo-admin** ユーザーのアクセスを有効化します。

このコマンドを実行する際には、オーバークラウドに関連するその他のファイルも追加するようにしてください。以下に例を示します。

- **-e** で指定するカスタム設定の環境ファイル
  - **--roles-file** で指定するカスタムロール (**roles\_data**) ファイル
  - **--networks-file** で指定するコンポーザブルネットワーク (**network\_data**) ファイル
3. オーバークラウドのデプロイメントのコマンドは、標準のスタック操作を実行します。ただし、オーバークラウドのスタックが設定段階に達すると、スタックは **config-download** メソッドに切り替わり、オーバークラウドを設定します。

```
2018-05-08 02:48:38Z [overcloud-AllNodesDeploySteps-xzihzsekhwo6]:
UPDATE_COMPLETE Stack UPDATE completed successfully
2018-05-08 02:48:39Z [AllNodesDeploySteps]: UPDATE_COMPLETE state
changed
2018-05-08 02:48:45Z [overcloud]: UPDATE_COMPLETE Stack UPDATE
```

```

completed successfully

Stack overcloud UPDATE_COMPLETE

Deploying overcloud configuration

```

オーバークラウドの設定が完了するまで待ちます。

4. Ansible によるオーバークラウドの設定が完了した後は、**director** が成功および失敗したタスクと、オーバークラウドのアクセス URL のレポートが表示されます。

```

PLAY RECAP
*****
192.0.2.101      : ok=173  changed=42  unreachable=0    failed=0
192.0.2.102      : ok=133  changed=42  unreachable=0    failed=0
localhost        : ok=2    changed=0   unreachable=0    failed=0

Ansible passed.
Overcloud configuration completed.
Started Mistral Workflow tripleo.deployment.v1.get_horizon_url.
Execution ID: 0e4ca4f6-9d14-418a-9c46-27692649b584
Overcloud Endpoint: http://10.0.0.1:5000/
Overcloud Horizon Dashboard URL: http://10.0.0.1:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
Overcloud Deployed

```

事前にプロビジョニング済みのノードを使用する場合には、追加のステップを実行して、**config-download** を使用したデプロイメントが成功するようにします。

### 10.3. 事前にプロビジョニング済みのノードでの **CONFIG-DOWNLOAD** の有効化

事前にプロビジョニング済みのノードで **config-download** を使用する場合には、Heat ベースのホスト名をそれらの実際のホスト名にマッピングして、**ansible-playbook** が解決されたホストに到達できるようにする必要があります。それらの値は、**HostnameMap** を使用してマッピングします。

#### 手順

1. 環境ファイル (例: **hostname-map.yaml**) を作成して、**HostnameMap** パラメーターとホスト名のマッピングを指定します。以下の構文を使用してください。

```

parameter_defaults:
  HostnameMap:
    [HEAT HOSTNAME]: [ACTUAL HOSTNAME]
    [HEAT HOSTNAME]: [ACTUAL HOSTNAME]

```

**[HEAT HOSTNAME]** は通常 **[STACK NAME] - [ROLE] - [INDEX]** の表記法に従います。以下に例を示します。

```

parameter_defaults:
  HostnameMap:
    overcloud-controller-0: controller-00-rack01
    overcloud-controller-1: controller-01-rack02

```

```

overcloud-controller-2: controller-02-rack03
overcloud-compute-0: compute-00-rack01
overcloud-compute-1: compute-01-rack01
overcloud-compute-2: compute-02-rack01

```

2. `hostname-map.yaml` の内容を保存します。
3. `config-download` のデプロイメントを実行する際には、`-e` オプションで環境ファイルを指定します。以下に例を示します。

```

$ openstack overcloud deploy --templates \
  --config-download \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/config-download-environment.yaml \
  -e /home/stack/templates/hostname-map.yaml \
  --overcloud-ssh-user heat-admin \
  --overcloud-ssh-key ~/.ssh/id_rsa \
  [OTHER OPTIONS]

```

## 10.4. CONFIG-DOWNLOAD の作業ディレクトリーへのアクセスの有効化

Mistral は、`config-download` 機能の Ansible Playbook の `execution` を実行します。Mistral は Playbook、設定ファイル、ログを作業ディレクトリーに保存します。この作業ディレクトリーは、`/var/lib/mistral/` にあり、Mistral ワークフローの `execution` の UUID を使用して名前が付けられています。

これらの作業ディレクトリーにアクセスする前に、`stack` ユーザーに適切なアクセス権を設定する必要があります。

### 手順

1. `mistral` グループは、`/var/lib/mistral` 下にある全ファイルを読み取ることができます。アンダークラウド上の `stack` ユーザーに、これらのファイルに対する対話型の読み取り専用アクセス権を付与します。

```
$ sudo usermod -a -G mistral stack
```

2. 以下のコマンドで `stack` ユーザーのパーミッションを更新します。

```
[stack@director ~]$ exec su -l stack
```

このコマンドでは再度ログインを要求されます。`stack` ユーザーのパスワードを入力します。

3. `/var/lib/mistral` ディレクトリーへの読み取りアクセスをテストします。

```
$ ls /var/lib/mistral/
```

## 10.5. CONFIG-DOWNLOAD のログと作業ディレクトリーの確認

`config-download` の過程には、Ansible によってアンダークラウドの `/var/lib/mistral/<execution uuid>/ansible.log` にログファイルが作成されます。`<execution uuid>` は、`ansible-playbook` を実行する Mistral の `execution` に対応する UUID

です。

## 手順

1. **openstack workflow execution list** コマンドで全 **execution** を一覧表示して、**config-download** を実行した、選択された **Mistral** の **execution** のワークフロー ID を特定します。

```
$ openstack workflow execution list
$ less /var/lib/mistral/<execution uuid>/ansible.log
```

**<execution uuid>** は、**ansible-playbook** を実行した **Mistral** の **execution** の **UUID** です。

2. または、**/var/lib/mistral** 下で直近に変更されたディレクトリーを探して、最新のデプロイメントのログを迅速に特定します。

```
$ less /var/lib/mistral/$(ls -t /var/lib/mistral | head -
1)/ansible.log
```

## 10.6. CONFIG-DOWNLOAD の手動による実行

**/var/lib/mistral/** 内の各作業ディレクトリーには、**ansible-playbook** と直接対話するために必要な **Playbook** とスクリプトが含まれています。以下の手順では、これらのファイルとの対話方法について説明します。

### 手順

1. 選択した **Ansible Playbook** のディレクトリーに移動します。

```
$ cd /var/lib/mistral/<execution uuid>/
```

**<execution uuid>** は、**ansible-playbook** を実行した **Mistral** の **execution** の **UUID** です。

2. **Mistral** の作業ディレクトリーに移動したら、**ansible-playbook-command.sh** を実行して、デプロイメントを再現します。

```
$ ./ansible-playbook-command.sh
```

3. このスクリプトには、追加の **Ansible** 引数を渡すことができます。それらの引数は、**ansible-playbook** コマンドに未変更で渡されます。これにより、チェックモード (**-check**)、ホストの限定 (**--limit**)、変数のオーバーライド (**-e**) などの **Ansible** の機能を更に活用することが可能となります。以下に例を示します。

```
$ ./ansible-playbook-command.sh --limit Controller
```

4. 作業ディレクトリーには、オーバークラウドの設定を実行する **deploy\_steps\_playbook.yaml** という名前の **Playbook** が含まれています。この **Playbook** を表示するには、以下のコマンドを実行します。

```
$ less deploy_steps_playbook.yaml
```

Playbook は、作業ディレクトリーに含まれているさまざまなタスクファイルを使用します。タスクファイルには、OpenStack Platform の全ロールに共通するものと、特定の OpenStack Platform ロールおよびサーバー固有のものがあります。

- 作業ディレクトリーには、オーバークラウドの **roles\_data** ファイルで定義されている各ロールに対応するサブディレクトリーも含まれます。以下に例を示します。

```
$ ls Controller/
```

各 OpenStack Platform ロールにディレクトリーには、そのロール種別の個々のサーバー用のサブディレクトリーも含まれます。これらのディレクトリーには、コンポーザブルロールのホスト名の形式を使用します。以下に例を示します。

```
$ ls Controller/overcloud-controller-0
```

- Ansible のタスクはタグ付けられます。タグの全一覧を確認するには、**ansible-playbook** で CLI の引数 **--list-tags** を使用します。

```
$ ansible-playbook -i tripleo-ansible-inventory.yaml --list-tags
deploy_steps_playbook.yaml
```

次に、**ansible-playbook-command.sh** スクリプトで **--tags**、**--skip-tags**、**--start-at-task** のいずれかを使用して、タグ付けた設定を適用します。以下に例を示します。

```
$ ./ansible-playbook-command.sh --tags overcloud
```



### 警告

**--tags**、**--skip-tags**、**--start-at-task** などの **ansible-playbook** CLI 引数を使用する場合には、デプロイメントの設定は、間違った順序で実行したり適用したりしないでください。これらの CLI 引数は、以前に失敗したタスクを再度実行する場合や、初回のデプロイメントを繰り返す場合に便利な方法です。ただし、デプロイメントの一貫性を保証するには、**deploy\_steps\_playbook.yaml** の全タスクを順番通りに実行する必要があります。

## 10.7. CONFIG-DOWNLOAD の無効化

標準の Heat ベースの設定メソッドに戻るには、次回に **openstack overcloud deploy** を実行する際に、関連するオプションと環境ファイルを削除します。

### 手順

- stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. オーバークラウドのデプロイメントのコマンドを実行しますが、**--config-download** オプションまたは「**config-download-environment.yaml**」環境ファイルは含めないでください。

```
$ openstack overcloud deploy --templates \  
    [OTHER OPTIONS]
```

このコマンドを実行する際には、オーバークラウドに関連するその他のファイルも追加するようにしてください。以下に例を示します。

- **-e** で指定するカスタム設定の環境ファイル
  - **--roles-file** で指定するカスタムロール (**roles\_data**) ファイル
  - **--networks-file** で指定するコンポーザブルネットワーク (**network\_data**) ファイル
3. オーバークラウドのデプロイメントのコマンドは、標準のスタック操作を実行します。これには、**Heat** を使用した設定が含まれます。

## 10.8. 次のステップ

これで、通常のオーバークラウドの操作を続行できるようになりました。

## 第11章 オーバークラウドのスケーリング



### 警告

コンピュータインスタンスの高可用性 (またはインスタンス HA。『[High Availability for Compute Instances](#)』で説明) を使用している場合は、アップグレードとスケールアップはできません。操作を試みても失敗します。

HA を有効化しているインスタンスがある場合には、アップグレードまたはスケールアップを実行する前に無効にしてください。そのためには、「[Rollback](#)」に記載のロールバックの操作を実行してください。

オーバークラウドの作成後に、ノードを追加または削除する必要がある場合があります。たとえば、オーバークラウドのコンピュータノードを追加する場合などです。このような状況では、オーバークラウドの更新が必要です。

以下の表を使用して、各ノード種別のスケーリングに対するサポートを判断してください。

表11.1 各ノード種別のスケーリングサポート

ノード種別	スケールアップ	スケールダウン	備考
コントローラー	×	×	
Compute	Y	Y	
Ceph Storage ノード	Y	×	オーバークラウドを最初に作成する際に Ceph Storage ノードを1つ以上設定する必要があります。
Block Storage ノード	×	×	
Object Storage ノード	Y	Y	リングを手動で管理する必要があります (「 <a href="#">Object Storage ノードの置き換え</a> 」に説明を記載)。



### 重要

オーバークラウドをスケーリングする前には、空き領域が少なくとも 10 GB あることを確認してください。この空き領域は、イメージの変換やノードのプロビジョニングプロセスのキャッシュに使用されます。

### 11.1. ノードのさらなる追加

director のノードプールにさらにノードを追加するには、登録する新規ノードの詳細を記載した新しい JSON ファイル (例: `newnodes.json`) を作成します。

```
{
  "nodes": [
    {
      "mac": [
        "dd:dd:dd:dd:dd:dd"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "pxe_ipmitool",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.168.24.207"
    },
    {
      "mac": [
        "ee:ee:ee:ee:ee:ee"
      ],
      "cpu": "4",
      "memory": "6144",
      "disk": "40",
      "arch": "x86_64",
      "pm_type": "pxe_ipmitool",
      "pm_user": "admin",
      "pm_password": "p@55w0rd!",
      "pm_addr": "192.168.24.208"
    }
  ]
}
```

これらのパラメーターについての説明は、[「オーバークラウドへのノードの登録」](#)を参照してください。

以下のコマンドを実行して、これらのノードを登録します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud node import newnodes.json
```

新規ノードを追加した後は、それらのイントロスペクションプロセスを起動します。各新規ノードに以下のコマンドを使用します。

```
(undercloud) $ openstack baremetal node manage [NODE UUID]
(undercloud) $ openstack overcloud node introspect [NODE UUID] --provide
```

このコマンドは、ノードのハードウェアプロパティの検出とベンチマークを実行します。

イントロスペクションプロセスの完了後には、各新規ノードを任意のロールにタグ付けしてスケーリングします。たとえば、コンピュータノードの場合には、以下のコマンドを使用します。

```
(undercloud) $ openstack baremetal node set --property
capabilities='profile:compute,boot_option:local' [NODE UUID]
```

デプロイメント中に使用するブートイメージを設定します。**bm-deploy-kernel** および **bm-deploy-ramdisk** イメージの UUID を確認します。

```
(undercloud) $ openstack image list
+-----+-----+
| ID                | Name                |
+-----+-----+
| 09b40e3d-0382-4925-a356-3a4b4f36b514 | bm-deploy-kernel   |
| 765a46af-4417-4592-91e5-a300ead3faf6 | bm-deploy-ramdisk  |
| ef793cd0-e65c-456a-a675-63cd57610bd5 | overcloud-full     |
| 9a51a6cb-4670-40de-b64b-b70f4dd44152 | overcloud-full-initrd |
| 4f7e33f4-d617-47c1-b36f-cbe90f132e5d | overcloud-full-vmlinuz |
+-----+-----+
```

新規ノードの **deploy\_kernel** および **deploy\_ramdisk** 設定にこれらの UUID を設定します。

```
(undercloud) $ openstack baremetal node set --driver-info
deploy_kernel='09b40e3d-0382-4925-a356-3a4b4f36b514' [NODE UUID]
(undercloud) $ openstack baremetal node set --driver-info
deploy_ramdisk='765a46af-4417-4592-91e5-a300ead3faf6' [NODE UUID]
```

オーバークラウドをスケーリングするには、環境ファイルでロールに必要なノード数を指定して **openstack overcloud deploy** を再実行する必要があります。たとえば、コンピュータノード 5 台にスケーリングするには、以下のコマンドを実行します。

```
parameter_defaults:
  ...
  ComputeCount: 5
  ...
```

更新したファイルを使用して、デプロイメントのコマンドを再度実行します。このファイルは、以下の例では、**node-info.yaml** という名前です。

```
(undercloud) $ openstack overcloud deploy --templates -e
/home/stack/templates/node-info.yaml [OTHER_OPTIONS]
```

上記のコマンドにより、オーバークラウドのスタック全体が更新されます。このコマンドが更新するのは、スタックのみである点に注意してください。オーバークラウドの削除や、スタックの置き換えは行われません。



### 重要

コンピュータ以外のノードに対する同様のスケジューリングパラメーターなど、最初に作成したオーバークラウドからの環境ファイルおよびオプションをすべて追加するようにしてください。

## 11.2. コンピュータノードの削除

オーバークラウドからコンピュータノードを削除する必要がある状況が出てくる可能性があります。たとえば、問題のあるコンピュータノードを置き換える必要がある場合などです。



## 重要

オーバークラウドからコンピューターノードを削除する前に、インスタンスをそのノードから別のコンピューターノードに移行してください。詳しくは、「[コンピューターノードからのインスタンスの移行](#)」を参照してください。

次に、オーバークラウド上でノードの **Compute** サービスを無効化します。これにより、ノードで新規インスタンスがスケジューリングされないようになります。

```
$ source ~/stack/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set [hostname] nova-compute --
disable
```

アンダークラウドに戻ります。

```
(overcloud) $ source ~/stack/stackrc
```

オーバークラウドノードを削除するには、ローカルのテンプレートファイルを使用して **overcloud** スタックへの更新が必要です。最初に、オーバークラウドスタックの **UUID** を特定します。

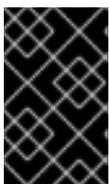
```
(undercloud) $ openstack stack list
```

削除するノードの **UUID** を特定します。

```
(undercloud) $ openstack server list
```

以下のコマンドを実行してスタックからノードを削除し、それに応じてプランを更新します。

```
(undercloud) $ openstack overcloud node delete --stack [STACK_UUID] --
templates -e [ENVIRONMENT_FILE] [NODE1_UUID] [NODE2_UUID] [NODE3_UUID]
```



## 重要

オーバークラウドの作成時に追加の環境ファイルを渡した場合には、オーバークラウドに、不要な変更が手動で加えられないように、ここで **-e** または **--environment-file** オプションを使用して環境ファイルを再度指定します。



## 重要

操作を続行する前に、**openstack overcloud node delete** コマンドが完全に終了したことを確認します。**openstack stack list** コマンドを使用して、**overcloud** スタックが **UPDATE\_COMPLETE** のステータスに切り替わっているかどうかをチェックしてください。

最後に、ノードの **Compute** サービスを削除します。

```
(undercloud) $ source ~/stack/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service delete [service-id]
```

ノードの **Open vSwitch** エージェントも削除します。

```
(overcloud) $ openstack network agent list
(overcloud) $ openstack network agent delete [openvswitch-agent-id]
```

オーバークラウドから自由にノードを削除して、別の目的でそのノードを再プロビジョニングすることができます。

### 11.3. コンピュートノードの置き換え

コンピュートノードに障害が発生した場合に、機能しているノードに置き換えることができます。コンピュートノードを置き換えるには、以下の手順を使用します。

- 既存のコンピュートノードからインスタンスを移行して、ノードをシャットダウンします。この手順は「[コンピュートノードからのインスタンスの移行](#)」を参照してください。
- オーバークラウドからコンピュートノードを削除します。この手順は「[コンピュートノードの削除](#)」を参照してください。
- 新しいコンピュートノードでオーバークラウドをスケールアウトします。その手順は、「[ノードのさらなる追加](#)」を参照してください。

このプロセスでは、インスタンスの可用性に影響を与えることなく、ノードを置き換えることができるようにします。

### 11.4. コントローラーノードの置き換え

特定の状況では、高可用性クラスター内のコントローラーノードに障害が発生することがあり、その場合は、そのコントローラーノードをクラスターから削除して新しいコントローラーノードに置き換える必要があります。このステップには、クラスター内の他のノードとの接続を確認する作業も含まれます。

本項では、コントローラーノードの置き換えの手順について説明します。このプロセスでは **openstack overcloud deploy** コマンドを実行してコントローラーノードの置き換えを要求し、オーバークラウドを更新します。このプロセスは、自動的に完了しない点に注意してください。オーバークラウドスタックの更新プロセスの途中で、**openstack overcloud deploy** コマンドによりエラーが報告されて、オーバークラウドスタックの更新が停止します。この時点で、プロセスに手動での介入が必要となり、その後に **openstack overcloud deploy** のプロセスを続行することができます。



#### 重要

以下の手順は、高可用性環境のみに適用します。コントローラーノード1台の場合には、この手順は使用しないでください。

#### 11.4.1. 事前のチェック

オーバークラウドコントローラーノードの置き換えを試みる前に、**Red Hat OpenStack Platform** 環境の現在の状態をチェックしておくことが重要です。このチェックしておくこと、コントローラーの置き換えプロセス中に複雑な事態が発生するのを防ぐことができます。以下の事前チェックリストを使用して、コントローラーノードの置き換えを実行しても安全かどうかを確認してください。チェックのためのコマンドはすべてアンダークラウドで実行します。

1. アンダークラウドで、**overcloud** スタックの現在の状態をチェックします。

```
$ source stackrc
(undercloud) $ openstack stack list --nested
```

**overcloud** スタックと後続の子スタックは、**CREATE\_COMPLETE** または **UPDATE\_COMPLETE** のステータスである必要があります。

- アンダークラウドデータベースのバックアップを実行します。

```
(undercloud) $ mkdir /home/stack/backup
(undercloud) $ sudo mysqldump --all-databases --quick --single-transaction | gzip > /home/stack/backup/dump_db_undercloud.sql.gz
```

- アンダークラウドで、新規ノードのプロビジョニング時にイメージのキャッシュと変換に対応できる **10 GB** の空きストレージ領域があるかどうかをチェックします。
- コントローラーノードで実行中の **Pacemaker** の状態をチェックします。たとえば、実行中のコントローラーノードの IP アドレスが **192.168.0.47** の場合には、以下のコマンドで **Pacemaker** のステータス情報を取得します。

```
(undercloud) $ ssh heat-admin@192.168.0.47 'sudo pcs status'
```

出力には、既存のノードで実行中のサービスと、障害が発生しているノードで停止中のサービスがすべて表示されるはずです。

- オーバークラウドの **MariaDB** クラスターの各ノードで以下のパラメーターをチェックします。

- **wsrep\_local\_state\_comment: Synced**

- **wsrep\_cluster\_size: 2**

実行中のコントローラーノードで以下のコマンドを使用して、パラメーターをチェックします (IP アドレスにはそれぞれ **192.168.0.47** と **192.168.0.46** を使用します)。

```
(undercloud) $ for i in 192.168.0.47 192.168.0.46 ; do echo "****
$i ****" ; ssh heat-admin@$i "sudo mysql -p\$(sudo hiera -c
/etc/puppet/hiera.yaml mysql::server::root_password) --
execute=\"SHOW STATUS LIKE 'wsrep_local_state_comment'; SHOW
STATUS LIKE 'wsrep_cluster_size';\""; done
```

- RabbitMQ** のステータスをチェックします。たとえば、実行中のコントローラーノードの IP アドレスが **192.168.0.47** の場合には、以下のコマンドを実行してステータスを取得します。

```
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo docker exec \$(sudo
docker ps -f name=rabbitmq-bundle -q) rabbitmqctl cluster_status"
```

**running\_nodes** キーには、障害が発生しているノードは表示されず、稼働中のノード 2 台のみが表示されるはずです。

- フェンシングが有効化されている場合には無効にします。たとえば、実行中のコントローラーノードの IP アドレスが **192.168.0.47** の場合には、以下のコマンドを実行してフェンシングを無効にします。

```
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo pcs property set
stonith-enabled=false"
```

以下のコマンドを実行してフェンシングのステータスを確認します。

```
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo pcs property show stonith-enabled"
```

8. **director** ノードで **nova-compute** サービスをチェックします。

```
(undercloud) $ sudo systemctl status openstack-nova-compute
(undercloud) $ openstack hypervisor list
```

出力では、メンテナンスモードに入っていないすべてのノードが **up** のステータスで表示されるはずですが。

9. アンダークラウドサービスがすべて実行中であることを確認します。

```
(undercloud) $ sudo systemctl -t service
```

### 11.4.2. Ceph monitor デーモンの削除

本手順では、ストレージクラスターから **ceph-mon** デーモンを削除します。コントローラーノードが **Ceph monitor** サービスを実行している場合には、以下のステップを完了して、**ceph-mon** デーモンを削除してください。この手順は、コントローラーが到達可能であることを前提としています。



#### 注記

新しい **Ceph monitor** デーモンは、クラスターに新しいコントローラーが追加された後に追加されます。

1. 置き換えるコントローラーに接続して、**root** になります。

```
# ssh heat-admin@192.168.0.47
# sudo su -
```



#### 注記

コントローラーが到達不可能な場合には、ステップ1と2をスキップして、稼働している任意のコントローラーノードでステップ3から手順を続行してください。

2. **root** として **monitor** を停止します。

```
# systemctl stop ceph-mon@<monitor_hostname>
```

例:

```
# systemctl stop ceph-mon@overcloud-controller-2
```

3. クラスターから **monitor** を削除します。

```
# ceph mon remove <mon_id>
```

4. Ceph monitor ノード上で、`/etc/ceph/ceph.conf` から `monitor` のエントリーを削除します。たとえば、`controller-2` を削除した場合には、`controller-2` の IP アドレスとホスト名を削除します。

編集前:

```
mon host = 172.18.0.21,172.18.0.22,172.18.0.24
mon initial members = overcloud-controller-2,overcloud-controller-1,overcloud-controller-0
```

編集後:

```
mon host = 172.18.0.22,172.18.0.24
mon initial members = overcloud-controller-1,overcloud-controller-0
```

5. オーバークラウドノードの `/etc/ceph/ceph.conf` に同じ変更を適用します。



### 注記

置き換え用のコントローラーノードが追加されると、`director` によって関連するノード上の `ceph.conf` ファイルが更新されます。通常、設定ファイルは `director` によってのみ管理され、手動で編集する必要はありませんが、このステップでは、新規ノードが追加される前に他のノードが再起動してしまった場合に一貫性を保つために、ファイルを編集しています。

6. オプションとして、`monitor` データをアーカイブして、別のサーバーに保存します。

```
# mv /var/lib/ceph/mon/<cluster>-<daemon_id>
/var/lib/ceph/mon/removed-<cluster>-<daemon_id>
```

### 11.4.3. ノードの置き換え

削除するノードのインデックスを特定します。ノードのインデックスは、`nova list` の出力に表示されるインスタンス名のサフィックスです。

```
(undercloud) $ openstack server list
+-----+-----+
| ID                | Name                |
+-----+-----+
| 861408be-4027-4f53-87a6-cd3cf206ba7a | overcloud-compute-0 |
| 0966e9ae-f553-447a-9929-c4232432f718 | overcloud-compute-1 |
| 9c08fa65-b38c-4b2e-bd47-33870bfff06c7 | overcloud-compute-2 |
| a7f0f5e1-e7ce-4513-ad2b-81146bc8c5af | overcloud-controller-0 |
| cfefaf60-8311-4bc3-9416-6a824a40a9ae | overcloud-controller-1 |
| 97a055d4-aeed-481c-82b7-4a5f384036d2 | overcloud-controller-2 |
+-----+-----+
```

この例では、`overcloud-controller-1` ノードを削除して、`overcloud-controller-3` に置き換えます。初めにノードをメンテナンスモードに切り替えて、`director` が障害の発生したノードを再プロビジョニングしないようにします。`nova list` で表示されるインスタンスの ID を、`openstack baremetal node list` で表示されるノード ID と関連させます。

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

```

+-----+-----+-----+
| UUID | Name | Instance UUID |
+-----+-----+-----+
| 36404147-7c8a-41e6-8c72-a6e90afc7584 | None | 7bee57cf-4a58-4eaf-b851-2a8bf6620e48 |
| 91eb9ac5-7d52-453c-a017-c0e3d823efd0 | None | None |
| 75b25e9a-948d-424a-9b3b-f0ef70a6eacf | None | None |
| 038727da-6a5c-425f-bd45-fda2f4bd145b | None | 763bfec2-9354-466a-ae65-2401c13e07e5 |
| dc2292e6-4056-46e0-8848-d6e96df1f55d | None | 2017b481-706f-44e1-852a-2ee857c303c4 |
| c7eadcea-e377-4392-9fc3-cf2b02b7ec29 | None | 5f73c7d7-4826-49a5-b6be-8bfd558f3b41 |
| da3a8d19-8a59-4e9d-923a-6a336fe10284 | None | cfefaf60-8311-4bc3-9416-6a824a40a9ae |
| 807cb6ce-6b94-4cd1-9969-5c47560c2eee | None | c07c13e6-a845-4791-9628-260110829c3a |
+-----+-----+-----+

```

ノードをメンテナンスモードに切り替えます。

```
(undercloud) $ openstack baremetal node maintenance set da3a8d19-8a59-4e9d-923a-6a336fe10284
```

新規ノードを **control** プロファイルでタグ付けします。

```
(undercloud) $ openstack baremetal node set --property capabilities='profile:control,boot_option:local' 75b25e9a-948d-424a-9b3b-f0ef70a6eacf
```

オーバークラウドのデータベースは、置き換え手順の実行中に稼働し続ける必要があります。この手順の実行中に **Pacemaker** が **Galera** を停止しないようにするには、実行中のコントローラーノードを選択して、そのコントローラーノードの IP アドレスを使用して、アンダークラウドで以下のコマンドを実行します。

```
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo pcs resource unmanage galera"
```

削除するノードインデックスを定義する YAML ファイルを作成します (**~/templates/remove-controller.yaml**)。

```
parameters:
  ControllerRemovalPolicies:
    [{'resource_list': ['1']}]
```



## 注記

Corosync 内での `settle` の試行回数を減らすことによって、置き換えプロセスをスピードアップすることができます。~/`templates/remove-controller.yaml` 環境ファイルで `CorosyncSettleTries` パラメーターを指定します。

```
parameter_defaults:
  CorosyncSettleTries: 5
```

ノードインデックスを特定した後は、オーバークラウドを再デプロイして、`remove-controller.yaml` 環境ファイルを追加します。

```
(undercloud) $ openstack overcloud deploy --templates --control-scale 3 -e
~/templates/remove-controller.yaml [OTHER OPTIONS]
```

オーバークラウドの作成時に追加の環境ファイルまたはオプションを渡した場合には、予定外の変更がオーバークラウドに加えられないように、その環境ファイルまたはオプションをここで再度渡してください。

ただし、`-e ~/templates/remove-controller.yaml` が必要なのは、この場合には1回のみである点に注意してください。

`director` は古いノードを削除して、新しいノードを作成してから、オーバークラウドスタックを更新します。以下のコマンドを使用すると、オーバークラウドスタックのステータスをチェックすることができます。

```
(undercloud) $ openstack stack list --nested
```

### 11.4.4. 手動での介入

`ControllerNodesPostDeployment` の段階中には、オーバークラウドスタックの更新が `ControllerDeployment_Step1` で `UPDATE_FAILED` エラーにより停止します。これは、一部の Puppet モジュールがノードの置き換えをサポートしていないためです。処理のこの時点で手動による介入が必要です。以下に記載する設定ステップに従ってください。

1. コントローラーノードの IP アドレスの一覧を取得します。以下に例を示します。

```
(undercloud) $ openstack server list -c Name -c Networks
+-----+-----+
| Name          | Networks          |
+-----+-----+
| overcloud-compute-0 | ctlplane=192.168.0.44 |
| overcloud-controller-0 | ctlplane=192.168.0.47 |
| overcloud-controller-2 | ctlplane=192.168.0.46 |
| overcloud-controller-3 | ctlplane=192.168.0.48 |
+-----+-----+
```

2. 各ノードの Corosync 設定から障害の発生したノードを削除して、Corosync を再起動します。この例では、`overcloud-controller-0` と `overcloud-controller-2` にログインして以下のコマンドを実行します。

```
(undercloud) $ for NAME in overcloud-controller-0 overcloud-
controller-2; do IP=$(openstack server list -c Networks -f value --
```

```
name $NAME | cut -d "=" -f 2) ; ssh heat-admin@$IP "sudo pcs cluster
localnode remove overcloud-controller-1; sudo pcs cluster reload
corosync"; done
```

3. 残りのノードの中の1台にログインして、**crm\_node** コマンドで対象のノードをクラスターから削除します。

```
(undercloud) $ ssh heat-admin@192.168.0.47
[heat-admin@overcloud-controller-0 ~]$ sudo crm_node -R overcloud-
controller-1 --force
```

このノードにログインした状態を維持します。

4. 障害が発生したノードを **RabbitMQ** クラスターから削除します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo docker exec -it $(sudo
docker ps -f name=rabbitmq-bundle -q) rabbitmqctl
forget_cluster_node rabbit@overcloud-controller-1
```

5. **Galera** クラスター内のノードの一覧を更新し、クラスターをリフレッシュします。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource update
galera cluster_host_map="overcloud-controller-0:overcloud-
controller-0.internalapi.localdomain;overcloud-controller-
3:overcloud-controller-3.internalapi.localdomain;overcloud-
controller-2:overcloud-controller-2.internalapi.localdomain"
wsrep_cluster_address="gcomm://overcloud-controller-
0.internalapi.localdomain,overcloud-controller-
3.internalapi.localdomain,overcloud-controller-
2.internalapi.localdomain"
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource refresh
galera
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource manage
galera
```

6. 新規ノードをクラスターに追加します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster node add
overcloud-controller-3
```

7. 新規コントローラーノードを起動します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster start
overcloud-controller-3
```

手動の設定が完了しました。コントローラーにログインした状態を維持します。

別のターミナルを開き、オーバークラウドのデプロイメントコマンドを再度実行して、スタックの更新を継続します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates --control-scale 3
[OTHER OPTIONS]
```



## 重要

オーバークラウドの作成時に追加の環境ファイルまたはオプションを渡した場合には、予定外の変更がオーバークラウドに加えられないように、その環境ファイルまたはオプションをここで再度渡してください。ただし、**remove-controller.yaml** ファイルは必要なくなった点に注意してください。

### 11.4.5. オーバークラウドサービスの最終処理

オーバークラウドのスタックの更新が完了したら、新たに追加されたコントローラーノード上で **Pacemaker** がコントローラーサービスを実行できるように、適切なクラスターノードのプロパティを設定します。既存のコントローラーノードのどれかで (たとえば、**overcloud-controller-0**)、以下のコマンドを実行します。

```
[heat-admin@overcloud-controller-0 ~]$ for i in $(sudo pcs property | grep
overcloud-controller-0: | cut -d' ' -f 3- | tr ' ' '\n' | grep role); do
sudo pcs property set --node overcloud-controller-3 $i; done
```

これ以降、新たに追加されたコントローラーノードで、**Pacemaker** の管理するサービスが実行されます。

最終のステータスチェックを実行して、サービスが正しく実行されていることを確認します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs status
```



## 注記

エラーが発生したサービスがある場合には、**pcs resource refresh** コマンドを使用して、問題の解決後にそのサービスを再起動します。

**director** を終了します。

```
[heat-admin@overcloud-controller-0 ~]$ exit
```

### 11.4.6. L3 エージェントのルーターホスティングの最終処理

オーバークラウドと対話できるようにするために、**source** コマンドで **overcloudrc** ファイルを読み込みます。ルーターをチェックして、L3 エージェントがオーバークラウド環境内のルーターを適切にホストしていることを確認します。以下の例では、**r1** という名前のルーターを使用します。

```
$ source ~/overcloudrc
(overcloud) $ neutron l3-agent-list-hosting-router r1
```

このリストには、新しいノードの代わりに、依然として古いノードが表示される場合があります。これを置き換えるには、環境内の L3 ネットワークエージェントを一覧表示します。

```
(overcloud) $ neutron agent-list | grep "neutron-l3-agent"
```

新しいノードと古いノード上でエージェントの **UUID** を特定します。新しいノードのエージェントにルーターを追加し、古いノードからそのルーターを削除します。以下に例を示します。

```
(overcloud) $ neutron l3-agent-router-add fd6b3d6e-7d8c-4e1a-831a-
```

```
4ec1c9ebb965 r1
(overcloud) $ neutron l3-agent-router-remove b40020af-c6dd-4f7a-b426-
eba7bac9dbc2 r1
```

ルーターに対して最終チェックを実行し、すべてがアクティブであることを確認します。

```
(overcloud) $ neutron l3-agent-list-hosting-router r1
```

古いコントローラーノードをポイントしている既存の **Neutron** エージェントを削除します。以下に例を示します。

```
(overcloud) $ neutron agent-list -F id -F host | grep overcloud-
controller-1
| ddae8e46-3e8e-4a1b-a8b3-c87f13c294eb | overcloud-controller-
1.localdomain |
(overcloud) $ neutron agent-delete ddae8e46-3e8e-4a1b-a8b3-c87f13c294eb
```

### 11.4.7. Compute サービスの最終処理

削除されたノードの **Compute** サービスはオーバークラウドにまだ存在しているので、削除する必要があります。 **source** コマンドで **overcloudrc** ファイルを読み込み、オーバークラウドと対話できるようにします。削除したノードの **Compute** サービスをチェックします。

```
[stack@director ~]$ source ~/overcloudrc
(overcloud) $ openstack compute service list --host overcloud-controller-
1.localdomain
```

削除したノードのコンピュートサービスを削除します。

```
(overcloud) $ for SERVICE in $(openstack compute service list --host
overcloud-controller-1.localdomain -f value -c ID) ; do openstack compute
service delete $SERVICE ; done
```

### 11.4.8. 結果

障害が発生したコントローラーノードと、関連サービスが新しいノードに置き換えられました。



#### 重要

「[Object Storage ノードの置き換え](#)」のように **Object Storage** でリングファイルの自動構築を無効にした場合には、新規ノード用に **Object Storage** リングファイルを手動で構築する必要があります。リングファイルの手動構築についての詳しい情報は、「[Object Storage ノードの置き換え](#)」を参照してください。

## 11.5. CEPH STORAGE ノードの置き換え

**director** では、**director** で作成したクラスター内の **Ceph Storage** ノードを置き換えることができます。手順については、「[Deploying an Overcloud with Containerized Red Hat Ceph](#)」ガイドを参照してください。

## 11.6. OBJECT STORAGE ノードの置き換え

本項では、クラスターの整合性を保ちながら **Object Storage** ノードを置き換える方法を説明します。以下の例では、2 台のノードで構成される **Object Storage** クラスターで、**overcloud-objectstorage-1** を置き換える必要があります。この手順は、ノードを 1 台追加して、**overcloud-objectstorage-1** を削除することを目的とします (実際には置き換えます)。

1. `~/templates/swift-upscale.yaml` という名前の環境ファイルを作成して、以下の内容を記載します。

```
parameter_defaults:
  ObjectStorageCount: 3
```

**ObjectStorageCount** は、環境内で **Object Storage** ノードをいくつ指定するかを定義します。今回の例では、ノードを 2 つから 3 つにスケールリングします。

2. **openstack overcloud deploy** の一部として、オーバークラウドの残りの環境ファイル (**ENVIRONMENT\_FILES**) と合わせて **swift-upscale.yaml** を追加します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates
ENVIRONMENT_FILES -e swift-upscale.yaml
```



### 注記

**swift-upscale.yaml** ファイルのパラメーターが以前の環境ファイルのパラメーターよりも優先されるように、このファイルを環境ファイルの一覧の最後に追加します。

デプロイメントが完了したら、オーバークラウドには別の **Object Storage** ノードが追加されています。

3. データは新しいノード用に複製する必要があります。ノードを削除する前に (この場合は **overcloud-objectstorage-1**)、**replication pass** が新規ノードで完了するのを待つ必要があります。 `/var/log/swift/swift.log` で複製パスの進捗を確認することができます。パスが完了すると、**Object Storage** サービスは以下のようなエントリーをログに残します。

```
Mar 29 08:49:05 localhost object-server: Object replication
complete.
Mar 29 08:49:11 localhost container-server: Replication run OVER
Mar 29 08:49:13 localhost account-server: Replication run OVER
```

4. リングから以前のノードを削除するには、**swift-upscale.yaml** の **ObjectStorageCount** の数を減らして以前のリングを省略します。今回は 3 から 2 に減らします。

```
parameter_defaults:
  ObjectStorageCount: 2
```

5. 新規環境ファイル (**remove-object-node.yaml**) を作成します。このファイルは、以前に指定した **Object Storage** ノードを特定し、削除します。以下の内容では **overcloud-objectstorage-1** の削除を指定します。

```
parameter_defaults:
  ObjectStorageRemovalPolicies:
    [{'resource_list': ['1']}]
```

6. デプロイメントのコマンドで両環境ファイルを指定します。

```
(undercloud) $ openstack overcloud deploy --templates
ENVIRONMENT_FILES -e swift-upscale.yaml -e remove-object-node.yaml
...
```

**director** は、オーバークラウドから **Object Storage** ノードを削除して、オーバークラウド上の残りのノードを更新し、ノードの削除に対応します。

## 11.7. ノードのブラックリスト登録

オーバークラウドノードがデプロイメントの更新を受け取らないように除外することができます。これは、既存のノードがコア **Heat** テンプレートコレクションから更新されたパラメーターセットやリソースを受け取らないように除外した状態で、新規ノードをスケールリングする場合に役立ちます。つまり、ブラックリストに登録されているノードは、スタック操作の影響を受けなくなります。

ブラックリストを作成するには、環境ファイルの **DeploymentServerBlacklist** パラメーターを使います。

### ブラックリストの設定

**DeploymentServerBlacklist** パラメーターは、サーバー名のリストです。新たな環境ファイルを作成するか、既存のカスタム環境ファイルにパラメーター値を追加して、ファイルをデプロイメントコマンドに渡します。

```
parameter_defaults:
  DeploymentServerBlacklist:
    - overcloud-compute-0
    - overcloud-compute-1
    - overcloud-compute-2
```



### 注記

パラメーター値のサーバー名には、実際のサーバーホスト名ではなく、**OpenStack Orchestration (Heat)** で定義されている名前を使用します。

**openstack overcloud deploy** コマンドで、この環境ファイルを指定します。以下に例を示します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates \
-e server-blacklist.yaml \
[OTHER OPTIONS]
```

**Heat** はリスト内のサーバーをすべてブラックリストし、**Heat** デプロイメントの更新を受け取らないようにします。スタック操作が完了した後は、ブラックリストに登録されたサーバーは以前の状態のままとなります。操作中に **os-collect-config** エージェントの電源をオフにしたり、停止したりすることもできます。



### 警告

- ノードをブラックリストに登録する場合には、注意が必要です。ブラックリストを有効にした状態で要求された変更を適用する方法を十分に理解していない限り、ブラックリストは使用しないでください。ブラックリスト機能を使うと、スタックがハングしたり、オーバークラウドが誤って設定されたりする場合があります。たとえば、クラスター設定の変更が Pacemaker クラスターの全メンバーに適用される場合には、この変更の間に Pacemaker クラスターのメンバーをブラックリストに登録すると、クラスターが機能しなくなる場合があります。
- 更新またはアップグレードの操作中にブラックリストを使わないでください。これらの操作には、特定のサーバーに対する変更を分離するための独自の方法があります。詳細は、『**Upgrading Red Hat OpenStack Platform**』のドキュメントを参照してください。
- サーバーをブラックリストに追加すると、そのサーバーをブラックリストから削除するまでは、それらのノードにはさらなる変更は適用されません。これには、更新、アップグレード、スケールアップ、スケールダウン、およびノードの置き換えが含まれます。

## ブラックリストのクリア

その後のスタック操作のためにブラックリストをクリアするには、**DeploymentServerBlacklist** を編集して空の配列を使用します。

```
parameter_defaults:
  DeploymentServerBlacklist: []
```



### 警告

**DeploymentServerBlacklist** パラメーターを単に削除しないでください。パラメーターを削除しただけの場合には、オーバークラウドデプロイメントには、前回保存された値が使用されます。

## 第12章 ノードの再起動

アンダークラウドおよびオーバークラウドでノードを再起動する必要がある場合があります。以下の手順では、異なるノード種別を再起動する方法を説明します。以下の点に注意してください。

- 1つのロールで全ノードを再起動する場合には、各ノードを個別に再起動することを推奨しています。この方法は、再起動中にそのロールのサービスを保持するのに役立ちます。
- OpenStack Platform 環境の全ノードを再起動する場合、再起動の順序は以下のリストを参考にしてください。

### 推奨されるノード再起動順

1. director の再起動
2. コントローラーとその他のコンポーザブルノードを再起動します。
3. Ceph Storage ノードの再起動
4. コンピュートノードの再起動

### 12.1. アンダークラウドノードの再起動

以下の手順では、アンダークラウドノードを再起動します。

#### 手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. アンダークラウドを再起動します。

```
$ sudo reboot
```

3. ノードが起動するまで待ちます。

### 12.2. コントローラーノードおよびコンポーザブルノードの再起動

以下の手順では、コントローラーノードと、コンポーザブルロールをベースとするスタンドアロンのノードを再起動します。これには、コンピュートノードと Ceph Storage ノードは含まれません。

#### 手順

1. ノードを選択してログインします。
2. ノードを再起動します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo reboot
```

3. ノードが起動するまで待ちます。
4. ノードにログインしてサービスをチェックします。以下に例を示します。
  - a. ノードが Pacemaker サービスを使用している場合には、ノードがクラスターに再度参加したかどうかを確認します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs status
```

- b. ノードが **Systemd** サービスを使用している場合には、全サービスが有効化されているかどうかを確認します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status
```

## 12.3. CEPH STORAGE (OSD) クラスターの再起動

以下の手順では、Ceph Storage (OSD) ノードのクラスターを再起動します。

### 手順

1. **Ceph MON** またはコントローラーノードにログインして、**Ceph Storage** クラスターのリバランスを一時的に無効にします。

```
$ sudo ceph osd set noout  
$ sudo ceph osd set norebalance
```

2. 再起動する最初の **Ceph Storage** ノードを選択して、ログインします。
3. ノードを再起動します。

```
$ sudo reboot
```

4. ノードが起動するまで待ちます。
5. ノードにログインして、クラスターのステータスを確認します。

```
$ sudo ceph -s
```

**pgs** が **pgmap** により通常通りに報告されていることを確認します (**active+clean**)。

6. ノードからログアウトして、次のノードを再起動し、ステータスを確認します。全 **Ceph Storage** ノードが再起動されるまで、このプロセスを繰り返します。
7. 完了したら、**Ceph MON** またはコントローラーノードにログインして、クラスターのリバランスを再度有効にします。

```
$ sudo ceph osd unset noout  
$ sudo ceph osd unset norebalance
```

8. 最終のステータスチェックを実行して、クラスターが **HEALTH\_OK** を報告していることを確認します。

```
$ sudo ceph status
```

## 12.4. コンピュートノードの再起動

以下の手順では、コンピュートノードを再起動します。OpenStack Platform 環境内のインスタンスのダウンタイムを最小限に抑えるために、この手順には、選択したコンピュートノードからインスタンスを移行するステップも含まれています。これは、以下のワークフローを伴います。

- 再起動するコンピューターノードを選択して無効にし、新規インスタンスをプロビジョニングしないようにします。
- インスタンスを別のコンピューターノードに移行します。
- 空のコンピューターノードを再起動して有効化します。

## 手順

1. アンダークラウドに **stack** ユーザーとしてログインします。

2. 全コンピューターノードとその UUID を一覧表示します。

```
$ source ~/stackrc
(undercloud) $ openstack server list --name compute
```

再起動するコンピューターノードのUUID を特定します。

3. アンダークラウドから、コンピューターノードを選択し、そのノードを無効にします。

```
$ source ~/overcloudrc
(overcloud) $ openstack compute service list
(overcloud) $ openstack compute service set [hostname] nova-compute
--disable
```

4. コンピューターノード上の全インスタンスを一覧表示します。

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

5. 以下のコマンドの1つを使用して、インスタンスを移行します。

- a. 選択した特定のホストにインスタンスを移行します。

```
(overcloud) $ openstack server migrate [instance-id] --live
[target-host]--wait
```

- b. **nova-scheduler** により対象のホストが自動的に選択されるようにします。

```
(overcloud) $ nova live-migration [instance-id]
```

- c. 一度にすべてのインスタンスのライブマイグレーションを行います。

```
$ nova host-evacuate-live [hostname]
```



### 注記

**nova** コマンドで非推奨の警告が表示される可能性があります。安全に無視することができます。

6. 移行が完了するまで待ちます。
7. 正常に移行したことを確認します。

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

8. 選択したコンピュートノードのインスタンスがなくなるまで、移行を続けます。
9. コンピュートノードにログインして、再起動します。

```
[heat-admin@overcloud-compute-0 ~]$ sudo reboot
```

10. ノードが起動するまで待ちます。
11. コンピュートノードを再度有効化します。

```
$ source ~/overcloudrc  
(overcloud) $ openstack compute service set [hostname] nova-compute  
--enable
```

12. コンピュートノードが有効化されているかどうかを確認します。

```
(overcloud) $ openstack compute service list
```

## 第13章 DIRECTOR の問題のトラブルシューティング

**director** プロセスの特定の段階で、エラーが発生する可能性があります。本項では、一般的な問題の診断に関する情報を提供します。

**director** のコンポーネントの共通ログを確認してください。

- `/var/log` ディレクトリーには、多数の **OpenStack Platform** の共通コンポーネントのログや、標準の **Red Hat Enterprise Linux** アプリケーションのログが含まれています。
- **journal** サービスは、さまざまなコンポーネントのログを提供します。**Ironic** は **openstack-ironic-api** と **openstack-ironic-conductor** の2つのユニットを使用する点に注意してください。同様に、**ironic-inspector** は **openstack-ironic-inspector** と **openstack-ironic-inspector-dnsmasq** の2つのユニットを使用します。該当するコンポーネントごとに両ユニットを使用します。以下に例を示します。

```
$ source ~/stackrc
(undercloud) $ sudo journalctl -u openstack-ironic-inspector -u
openstack-ironic-inspector-dnsmasq
```

- **ironic-inspector** は、`/var/log/ironic-inspector/ramdisk/` に **ramdisk** ログを **gz** 圧縮の **tar** ファイルとして保存します。ファイル名には、日付、時間、ノードの **IPMI** アドレスが含まれます。イントロスペクションの問題を診断するには、これらのログを使用します。

### 13.1. ノード登録のトラブルシューティング

ノード登録における問題は通常、ノードの情報が間違っていることが原因で発生します。このような場合には、**ironic** を使用して、登録したノードデータの問題を修正します。以下にいくつか例を示します。

割り当てられたポートの **UUID** を確認します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal port list --node [NODE UUID]
```

**MAC** アドレスを更新します。

```
(undercloud) $ openstack baremetal port set --address=[NEW MAC] [PORT
UUID]
```

以下のコマンドを実行します。

```
(undercloud) $ openstack baremetal node set --driver-info ipmi_address=[NEW
IPMI ADDRESS] [NODE UUID]
```

### 13.2. ハードウェアイントロスペクションのトラブルシューティング

イントロスペクションのプロセスは最後まで実行する必要があります。ただし、**Ironic** の **Discovery Daemon (ironic-inspector)** は、検出する **ramdisk** が応答しない場合にはデフォルトの1時間が経過するとタイムアウトします。検出する **ramdisk** のバグが原因とされる場合もありますが、通常は特に **BIOS** の起動設定などの環境の誤設定が原因で発生します。

以下には、環境設定が間違っている場合の一般的なシナリオと、診断/解決方法に関するアドバイスを示します。

## ノードのイントロスペクション開始におけるエラー

通常、イントロスペクションプロセスは、**openstack overcloud node introspect** コマンドを使用します。ただし、**ironic-inspector** で直接イントロスペクションを実行している場合には、「AVAILABLE」の状態のノードの検出に失敗する可能性があります。このコマンドは、デプロイメント用であり、検出用ではないためです。検出前に、ノードのステータスを「MANAGEABLE」に変更します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal node manage [NODE UUID]
```

検出が完了したら、状態を「AVAILABLE」に戻してからプロビジョニングを行います。

```
(undercloud) $ openstack baremetal node provide [NODE UUID]
```

## 検出プロセスの停止

イントロスペクションのプロセスを停止します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal introspection abort [NODE UUID]
```

プロセスがタイムアウトするまで待つことも可能です。必要であれば、**/etc/ironic-inspector/inspector.conf** の **timeout** 設定を別の時間 (分単位) に変更します。

## イントロスペクション ramdisk へのアクセス

イントロスペクションの **ramdisk** は、動的なログイン要素を使用します。これは、イントロスペクションのデバッグ中にノードにアクセスするための一時パスワードまたは SSH キーのいずれかを提供できることを意味します。以下のプロセスを使用して、**ramdisk** アクセスを設定します。

1. 以下のように **openssl passwd -1** コマンドに一時パスワードを指定して MD5 ハッシュを生成します。

```
$ openssl passwd -1 mytestpassword
$1$enjRSyIw$/fYUpJwr6abFy/d.koRgQ/
```

2. **/httpboot/inspector.ipxe** ファイルを編集して、**kernel** で開始する行を特定し、**rootpwd** パラメーターと MD5 ハッシュを追記します。以下に例を示します。

```
kernel http://192.2.0.1:8088/agent.kernel ipa-inspection-callback-
url=http://192.168.0.1:5050/v1/continue ipa-inspection-
collectors=default,extra-hardware,logs
systemd.journald.forward_to_console=yes BOOTIF=${mac} ipa-debug=1
ipa-inspection-benchmarks=cpu,mem,disk
rootpwd="$1$enjRSyIw$/fYUpJwr6abFy/d.koRgQ/" selinux=0
```

または、**sshkey** パラメーターに SSH 公開キーを追記します。



## 注記

**rootpwd** および **sshkey** のパラメーターにはいずれも引用符が必要です。

3. イントロスペクションを開始し、**arp** コマンドまたは **DHCP** のログから **IP** アドレスを特定します。

```
$ arp
$ sudo journalctl -u openstack-ironic-inspector-dnsmasq
```

4. 一時パスワードまたは **SSH** キーを使用して、**root** ユーザーとして **SSH** 接続します。

```
$ ssh root@192.168.24.105
```

### イントロスペクションストレージのチェック

**director** は **OpenStack Object Storage (swift)** を使用して、イントロスペクションプロセス中に取得したハードウェアデータを保存します。このサービスが稼働していない場合には、イントロスペクションは失敗する場合があります。以下のコマンドを実行して、**OpenStack Object Storage** に関連したサービスをすべてチェックし、このサービスが稼働中であることを確認します。

```
$ sudo systemctl list-units openstack-swift*
```

## 13.3. WORKFLOW および EXECUTION のトラブルシューティング

**OpenStack Workflow (mistral)** サービスは、複数の **OpenStack** タスクをワークフローにグループ化します。**Red Hat OpenStack Platform** は、これらのワークフローセットを使用して、**CLI** と **Web UI** で共通の機能を実行します。これには、ベアメタルノードの制御、検証、プラン管理、オーバークラウドのデプロイメントが含まれます。

たとえば **openstack overcloud deploy** コマンドを実行すると、**OpenStack Workflow** サービスは2つのワークフローを実行します。最初はデプロイメントプランのアップロードです。

```
Removing the current plan files
Uploading new plan files
Started Mistral Workflow. Execution ID: aef1e8c6-a862-42de-8bce-
073744ed5e6b
Plan updated
```

2つ目は、オーバークラウドのデプロイメントを開始します。

```
Deploying templates in the directory /tmp/tripleoclient-LhRlHX/tripleo-
heat-templates
Started Mistral Workflow. Execution ID: 97b64abe-d8fc-414a-837a-
1380631c764d
2016-11-28 06:29:26Z [overcloud]: CREATE_IN_PROGRESS Stack CREATE started
2016-11-28 06:29:26Z [overcloud.Networks]: CREATE_IN_PROGRESS state
changed
2016-11-28 06:29:26Z [overcloud.HeatAuthEncryptionKey]: CREATE_IN_PROGRESS
state changed
2016-11-28 06:29:26Z [overcloud.ServiceNetMap]: CREATE_IN_PROGRESS state
changed
...
```

## Workflow オブジェクト

OpenStack Workflow では、以下のオブジェクトを使用して、ワークフローを記録します。

### Actions

Shell スクリプトの実行や HTTP 要求の実行など、関連タスクが実行された場合に OpenStack が実行する特定の指示。OpenStack のコンポーネントには、OpenStack Workflow が使用する Actions が含まれます。

### Tasks

実行するアクションと、アクションの実行後の結果を定義します。これらのタスクには通常、アクションまたはアクションに関連付けられたワークフローが含まれます。タスクが完了したら、ワークフローは、タスクが成功したか否かによって、別のタスクに指示を出します。

### Workflows

グループ化されて、特定の順番で実行されるタスクセット

### Executions

実行する特定のアクション、タスク、またはワークフローを定義します。

## Workflow のエラー診断

OpenStack Workflow では、実行に関して着実にログを取ることもできるので、特定のコマンドが失敗した場合に問題を特定しやすくなります。たとえば、ワークフローの実行に失敗した場合には、どの部分で失敗したかを特定することができます。ワークフローの実行に失敗した状態 (**ERROR**) のものを表示します。

```
$ source ~/stackrc
(undercloud) $ openstack workflow execution list | grep "ERROR"
```

失敗したワークフローの実行の UUID を取得して (例: `dffa96b0-f679-4cd2-a490-4769a3825262`)、実行とその出力を表示します。

```
(undercloud) $ openstack workflow execution show dffa96b0-f679-4cd2-a490-4769a3825262
(undercloud) $ openstack workflow execution output show dffa96b0-f679-4cd2-a490-4769a3825262
```

これにより、実行で失敗したタスクに関する情報を取得できます。**openstack workflow execution show** は、実行に使用したワークフローも表示します (例: **tripleo.plan\_management.v1.publish\_ui\_logs\_to\_swift**)。以下のコマンドを使用して完全なワークフロー定義を表示することができます。

```
(undercloud) $ openstack workflow definition show
tripleo.plan_management.v1.publish_ui_logs_to_swift
```

これは、特定のタスクがワークフローのどの部分で発生するかを特定する際に便利です。

同様のコマンド構文を使用して、アクションの実行と、その結果を表示することもできます。

```
(undercloud) $ openstack action execution list
(undercloud) $ openstack action execution show 8a68eba3-0fec-4b2a-adc9-5561b007e886
(undercloud) $ openstack action execution output show 8a68eba3-0fec-4b2a-adc9-5561b007e886
```

これは、問題を引き起こす固有のアクションを特定する際に便利です。

## 13.4. オーバークラウドの作成のトラブルシューティング

デプロイメントが失敗する可能性のあるレイヤーは 3 つあります。

- **Orchestration (Heat および Nova サービス)**
- **Bare Metal Provisioning (Ironic サービス)**
- **デプロイメント後の設定 (Puppet)**

オーバークラウドのデプロイメントがこれらのレベルで失敗した場合には、**OpenStack** クライアントおよびサービスログファイルを使用して、失敗したデプロイメントの診断を行います。

### 13.4.1. Orchestration

多くの場合は、オーバークラウドの作成に失敗した後に、**Heat** により失敗したオーバークラウドスタックが表示されます。

```
$ source ~/stackrc
(undercloud) $ openstack stack list --nested --property status=FAILED
+-----+-----+-----+-----+
| id                | stack_name | stack_status      | creation_time |
+-----+-----+-----+-----+
| 7e88af95-535c-4a55... | overcloud  | CREATE_FAILED     | 2015-04-06T17:57:16Z |
+-----+-----+-----+-----+
```

スタック一覧が空の場合には、初期の **Heat** 設定に問題があることが分かります。**Heat** テンプレートと設定オプションをチェックし、さらに **openstack overcloud deploy** を実行後のエラーメッセージを確認してください。

### 13.4.2. Bare Metal Provisioning

**ironic** をチェックして、全登録ノードと現在の状態を表示します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal node list
+-----+-----+-----+-----+-----+-----+
| UUID      | Name | Instance UUID | Power State | Provision State | Maintenance |
+-----+-----+-----+-----+-----+-----+
| f1e261... | None | None          | power off   | available        | False       |
| f0b8c1... | None | None          | power off   | available        | False       |
```

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

プロビジョニングプロセスでよく発生する問題を以下に示します。

- 結果の表の「**Provision State**」および「**Maintenance**」の列を確認します。以下をチェックしてください。
  - 空の表または、必要なノード数よりも少ない
  - 「**Maintenance**」が **True** に設定されている
  - プロビジョニングの状態が **manageable** に設定されている。これにより、登録または検出プロセスに問題があることが分かります。たとえば、「**Maintenance**」が **True** に自動的に設定された場合は通常、ノードの電源管理の認証情報が間違っています。
- 「**Provision State**」が **available** の場合には、ベアメタルのデプロイメントが開始される前に問題が発生します。
- 「**Provision State**」が **active** で、「**Power State**」が **power on** の場合には、ベアメタルのデプロイメントは正常に完了しますが、問題は、デプロイメント後の設定ステップで発生することになります。
- ノードの「**Provision State**」が **wait call-back** の場合には、このノードではまだ **Bare Metal Provisioning** プロセスが完了していません。このステータスが変わるまで待ってください。ステータスがかわらない場合には、問題のあるノードの仮想コンソールに接続して、出力を確認します。
- 「**Provision State**」が **error** または **deploy failed** の場合には、このノードの **Bare Metal Provisioning** は失敗しています。ベアメタルノードの詳細を確認してください。

```
(undercloud) $ openstack baremetal node show [NODE UUID]
```

エラーの説明が含まれる **last\_error** フィールドがないか確認します。エラーメッセージは曖昧なため、ログを使用して解明します。

```
(undercloud) $ sudo journalctl -u openstack-ironic-conductor -u
openstack-ironic-api
```

- **wait timeout error** が表示されており、「**Power State**」が **power on** の場合には、問題のあるノードの仮想コンソールに接続して、出力を確認します。

### 13.4.3. デプロイメント後の設定

設定ステージでは多くのことが発生する可能性があります。たとえば、設定に問題があるために、特定の **Puppet** モジュールの完了に失敗する可能性があります。本項では、これらの問題を診断するプロセスを説明します。

オーバークラウドスタックからのリソースをすべて表示して、どのスタックに問題があるのかを確認します。

```
$ source ~/stackrc
(undercloud) $ openstack stack resource list overcloud --filter
status=FAILED
```

このコマンドにより、問題のあるリソースの全リストが表示されます。

問題のあるリソースを表示します。

```
(undercloud) $ openstack stack resource show overcloud [FAILED RESOURCE]
```

**resource\_status\_reason** のフィールドの情報で診断に役立つ可能性のあるものを確認します。

**nova** コマンドを使用して、オーバークラウドノードの IP アドレスを表示します。

```
(undercloud) $ openstack server list
```

デプロイされたノードの1つに **heat-admin** ユーザーとしてログインします。たとえば、スタックのリソース一覧から、コントローラーノード上にエラーが発生していることが判明した場合には、コントローラーノードにログインします。**heat-admin** ユーザーには、**sudo** アクセスが設定されています。

```
(undercloud) $ ssh heat-admin@192.168.24.14
```

**os-collect-config** ログを確認して、考えられる失敗の原因をチェックします。

```
[heat-admin@overcloud-controller-0 ~]$ sudo journalctl -u os-collect-config
```

場合によっては、Nova によるノードへのデプロイメントが完全に失敗する可能性があります。このような場合にはオーバークラウドのロール種別の1つの **OS::Heat::ResourceGroup** が失敗していることが示されるため、**nova** を使用して問題を確認します。

```
(undercloud) $ openstack server list
(undercloud) $ openstack server show [SERVER ID]
```

最もよく表示されるエラーは、**No valid host was found** のエラーメッセージです。このエラーのトラブルシューティングについては、「["No Valid Host Found" エラーのトラブルシューティング](#)」を参照してください。その他の場合は、以下のログファイルを参照してトラブルシューティングを実施してください。

- **/var/log/nova/\***
- **/var/log/heat/\***
- **/var/log/ironic/\***

コントローラーノードのデプロイ後のプロセスは、5つの主なステップで構成されます。以下のステップが含まれます。

表13.1 コントローラーノードの設定ステップ

手順	説明
<b>ControllerDeployment_Step1</b>	Pacemaker、RabbitMQ、Memcached、Redis、および Galera を含むロードバランシング用のソフトウェアの初期設定

<b>ControllerDeployment_Step2</b>	Pacemaker の設定、HAProxy、MongoDB、Galera、Ceph Monitor、OpenStack Platform の各種サービス用のデータベースの初期化を含む、クラスターの初期設定
<b>ControllerDeployment_Step3</b>	OpenStack Object Storage (swift) の最初のリング構築。OpenStack Platform の全サービス (nova、neutron、cinder、sahara、ceilometer、heat、horizon、aodh、gnocchi) の設定
<b>ControllerDeployment_Step4</b>	サービス起動順序やサービス起動パラメーターを決定するための制約事項を含む、Pacemaker でのサービスの起動設定値の設定
<b>ControllerDeployment_Step5</b>	OpenStack Identity (keystone) のプロジェクト、ロール、ユーザーの初期設定

### 13.5. プロビジョニングネットワークでの IP アドレスの競合に対するトラブルシューティング

宛先のホストに、すでに使用中の IP アドレスが割り当てられている場合には、検出およびデプロイメントのタスクは失敗します。この問題を回避するには、プロビジョニングネットワークのポートスキャンを実行して、検出の IP アドレス範囲とホストの IP アドレス範囲が解放されているかどうかを確認することができます。

アンダークラウドホストで以下のステップを実行します。

**nmap** をインストールします。

```
$ sudo yum install nmap
```

**nmap** を使用して、アクティブなアドレスの IP アドレス範囲をスキャンします。この例では、**192.168.24.0/24** の範囲をスキャンします。この値は、プロビジョニングネットワークの IP サブネットに置き換えてください (CIDR 表記のビットマスク)。

```
$ sudo nmap -sn 192.168.24.0/24
```

**nmap** スキャンの出力を確認します。

たとえば、アンダークラウドおよびサブネット上に存在するその他のホストの IP アドレスを確認する必要があります。アクティブな IP アドレスが **undercloud.conf** の IP アドレス範囲と競合している場合には、オーバークラウドノードのイントロスペクションまたはデプロイを実行する前に、IP アドレスの範囲を変更するか、IP アドレスを解放するかのいずれかを行う必要があります。

```
$ sudo nmap -sn 192.168.24.0/24
```

```
Starting Nmap 6.40 ( http://nmap.org ) at 2015-10-02 15:14 EDT
Nmap scan report for 192.168.24.1
Host is up (0.00057s latency).
```

```
Nmap scan report for 192.168.24.2
Host is up (0.00048s latency).
Nmap scan report for 192.168.24.3
Host is up (0.00045s latency).
Nmap scan report for 192.168.24.5
Host is up (0.00040s latency).
Nmap scan report for 192.168.24.9
Host is up (0.00019s latency).
Nmap done: 256 IP addresses (5 hosts up) scanned in 2.45 seconds
```

### 13.6. "NO VALID HOST FOUND" エラーのトラブルシューティング

`/var/log/nova/nova-conductor.log` に、以下のエラーが含まれる場合があります。

```
NoValidHost: No valid host was found. There are not enough hosts
available.
```

これは、Nova Scheduler が新規インスタンスを起動するのに適したベアメタルノードを検出できなかったことを意味し、そうなると通常は、Nova が検出を想定しているリソースと、Ironic が Nova に通知するリソースが一致しなくなります。その場合には、以下の点をチェックしてください。

1. イントロスペクションが正常に完了することを確認してください。または、各ノードに必要な Ironic ノードのプロパティが含まれていることをチェックしてください。各ノードに以下のコマンドを実行します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal node show [NODE UUID]
```

`properties` JSON フィールドの `cpus`、`cpu_arch`、`memory_mb`、`local_gb` キーに有効な値が指定されていることを確認してください。

2. 使用する Nova フレーバーが、必要なノード数において、上記の Ironic ノードプロパティを超えていないかどうかを確認します。

```
(undercloud) $ openstack flavor show [FLAVOR NAME]
```

3. `openstack baremetal node list` の出力で、`available` の状態のノードの数が十分かどうかを確認します。ノードの状態が `manageable` の場合は通常、イントロスペクションに失敗しています。
4. また、ノードがメンテナンスモードではないことを確認します。`openstack baremetal node list` を使用してチェックしてください。通常、自動でメンテナンスモードに切り替わるノードは、電源の認証情報が間違っています。認証情報を確認して、メンテナンスモードをオフにします。

```
(undercloud) $ openstack baremetal node maintenance unset [NODE
UUID]
```

5. Automated Health Check (AHC) ツールを使用して、自動でノードのタグ付けを行う場合には、各フレーバー/フレーバーに対応するノードが十分に存在することを確認します。`properties` フィールドの `capabilities` キーに `openstack baremetal node show` がいないか確認します。たとえば、コンピュータロールのタグを付けられたノードには、`profile:compute` が含まれているはずで

6. イントросペクションの後に **Ironic** から **Nova** にノードの情報が反映されるには若干時間がかかります。これは通常、**director** のツールが原因です。ただし、一部のステップを手動で実行した場合には、短時間ですが、**Nova** でノードが利用できなくなる場合があります。以下のコマンドを使用して、システム内の合計リソースをチェックします。

```
(undercloud) $ openstack hypervisor stats show
```

## 13.7. オーバークラウド作成後のトラブルシューティング

オーバークラウドを作成した後は、将来そのオーバークラウドで特定の操作を行うようにすることができます。たとえば、利用可能なノードをスケーリングしたり、障害の発生したノードを置き換えたりすることができます。このような操作を行うと、特定の問題が発生する場合があります。本項には、オーバークラウド作成後の操作が失敗した場合の診断とトラブルシューティングに関するアドバイスを記載します。

### 13.7.1. オーバークラウドスタックの変更

**director** を使用して **overcloud** スタックを変更する際に問題が発生する場合があります。スタックの変更例には、以下のような操作が含まれます。

- ノードのスケーリング
- ノードの削除
- ノードの置き換え

スタックの変更は、スタックの作成と似ており、**director** は要求されたノード数が利用可能かどうかをチェックして追加のノードをプロビジョニングしたり、既存のノードを削除したりしてから、**Puppet** の設定を適用します。**overcloud** スタックを変更する場合に従うべきガイドラインを以下に記載します。

第一段階として、「[デプロイメント後の設定](#)」に記載したアドバイスに従います。この手順と同じステップを、**overcloud Heat** スタック更新の問題の診断に役立てることができます。特に、以下のコマンドを使用して問題のあるリソースを特定します。

#### **openstack stack list --show-nested**

全スタックを一覧表示します。**--show-nested** はすべての子スタックとそれぞれの親スタックを表示します。このコマンドは、スタックでエラーが発生した時点を特定するのに役立ちます。

#### **openstack stack resource list overcloud**

**overcloud** スタック内の全リソースとそれらの状態を一覧表示します。このコマンドは、スタック内でどのリソースが原因でエラーが発生しているかを特定するのに役立ちます。このリソースのエラーの原因となっている **Heat** テンプレートコレクションと **Puppet** モジュール内のパラメーターと設定を特定することができます。

#### **openstack stack event list overcloud**

**overcloud** スタックに関連するすべてのイベントを時系列で一覧表示します。これには、スタック内の全リソースのイベント開始/完了時間とエラーが含まれます。この情報は、リソースの障害点を特定するのに役立ちます。

以下のセクションには、特定の種別のノード上の問題診断に関するアドバイスを記載します。

### 13.7.2. コントローラーサービスのエラー

オーバークラウドコントローラーノードには、Red Hat OpenStack Platform のサービスの大部分が含ま

れます。同様に、高可用性のクラスターで複数のコントローラーノードを使用することができます。ノード上の特定のサービスに障害が発生すると、高可用性のクラスターは一定レベルのフェイルオーバーを提供します。ただし、オーバークラウドをフル稼働させるには、障害のあるサービスの診断が必要となります。

コントローラーノードは、**Pacemaker** を使用して高可用性クラスター内のリソースとサービスを管理します。**Pacemaker Configuration System (pcs)** コマンドは、**Pacemaker** クラスターを管理するツールです。クラスター内のコントローラーノードでこのコマンドを実行して、設定およびモニタリングの機能を実行します。高可用性クラスター上のオーバークラウドサービスのトラブルシューティングに役立つコマンドを以下にいくつか記載します。

### **pcs status**

有効なリソース、エラーが発生したリソース、オンラインのノードなど、クラスター全体のステータス概要を提供します。

### **pcs resource show**

リソースの一覧をそれぞれのノードで表示します。

### **pcs resource disable [resource]**

特定のリソースを停止します。

### **pcs resource enable [resource]**

特定のリソースを起動します。

### **pcs cluster standby [node]**

ノードをスタンバイモードに切り替えます。そのノードはクラスターで利用できなくなります。このコマンドは、クラスターに影響を及ぼさずに特定のノードメンテナンスを実行するのに役立ちます。

### **pcs cluster unstandby [node]**

ノードをスタンバイモードから解除します。ノードはクラスター内で再度利用可能となります。

これらの **Pacemaker** コマンドを使用して障害のあるコンポーネントおよびノードを特定します。コンポーネントを特定した後は、**/var/log/** でそれぞれのコンポーネントのログファイルを確認します。

## 13.7.3. コンテナ化されたサービスのエラー

オーバークラウドのデプロイメント中またはデプロイメント後にコンテナ化されたサービスでエラーが発生した場合には、以下の推奨事項に従って、エラーの根本的な原因を特定してください。



### 注記

これらのコマンドを実行する前には、オーバークラウドノードにログイン済みであることを確認し、これらのコマンドをアンダークラウドで実行しないようにしてください。

### コンテナログの確認

各コンテナは、主要プロセスからの標準出力を保持します。この出力はログとして機能し、コンテナ実行時に実際に何が発生したのかを特定するのに役立ちます。たとえば、**keystone** コンテナのログを確認するには、以下のコマンドを使用します。

```
$ sudo docker logs keystone
```

大半の場合は、このログにコンテナのエラーの原因が記載されています。

## コンテナの検査

状況によっては、コンテナに関する情報を検証する必要がある場合があります。たとえば、以下のコマンドを使用して **keystone** コンテナのデータを確認します。

```
$ sudo docker inspect keystone
```

これにより、ローレベルの設定データが含まれた **JSON** オブジェクトが提供されます。その出力を **jq** コマンドにパイプして、特定のデータを解析することが可能です。たとえば、**keystone** コンテナのマウントを確認するには、以下のコマンドを実行します。

```
$ sudo docker inspect keystone | jq .[0].Mounts
```

**--format** オプションを使用して、データを単一行に解析することができます。これは、コンテナデータのセットに対してコマンドを実行する場合に役立ちます。たとえば、**keystone** コンテナを実行するのに使用するオプションを再生成するには、以下のように **inspect** コマンドに **--format** オプションを指定して実行します。

```
$ sudo docker inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}}
{{range .Mounts}} -v {{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}
{{end}}{{end}} -ti {{.Config.Image}}' keystone
```



### 注記

**--format** オプションは、**Go** 構文を使用してクエリを作成します。

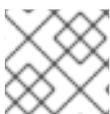
これらのオプションを **docker run** コマンドとともに使用して、トラブルシューティング目的のコンテナを再度作成します。

```
$ OPTIONS=$( sudo docker inspect --format='{{range .Config.Env}} -e "
{{.}}" {{end}} {{range .Mounts}} -v {{.Source}}:{{.Destination}}{{if
.Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}' keystone )
$ sudo docker run --rm $OPTIONS /bin/bash
```

## コンテナ内でのコマンドの実行

状況によっては、特定の **Bash** コマンドでコンテナ内の情報を取得する必要がある場合があります。このような場合には、以下の **docker** コマンドを使用して、稼働中のコンテナ内でコマンドを実行します。たとえば、**keystone** コンテナで次のコマンドを実行します。

```
$ sudo docker exec -ti keystone <COMMAND>
```



### 注記

**-ti** オプションを指定すると、コマンドは対話式の擬似ターミナルで実行されます。

**<COMMAND>** は必要なコマンドに置き換えます。たとえば、各コンテナには、サービスの接続を確認するためのヘルスチェックスクリプトがあります。**keystone** にヘルスチェックスクリプトを実行するには、以下のコマンドを実行します。

```
$ sudo docker exec -ti keystone /openstack/healthcheck
```

コンテナのシェルにアクセスするには、`/bin/bash` をコマンドとして使用する `docker exec` を実行します。

```
$ sudo docker exec -ti keystone /bin/bash
```

### コンテナのエクスポート

コンテナが失敗した場合には、ファイルの詳細を調べる必要があります。この場合は、コンテナの全ファイルシステムを `tar` アーカイブとしてエクスポートすることができます。たとえば、`keystone` コンテナのファイルシステムをエクスポートするには、以下のコマンドを実行します。

```
$ sudo docker export keystone -o keystone.tar
```

このコマンドにより `keystone.tar` アーカイブが作成されます。これを抽出して、調べることができます。

### 13.7.4. Compute サービスのエラー

Compute ノードは、Compute サービスを使用して、ハイパーバイザーベースの操作を実行します。これは、このサービスを中心にコンピュートノードのメインの診断が行われていることを意味します。以下に例を示します。

- コンテナのステータスを表示します。

```
$ sudo docker ps -f name=nova_compute
```

- コンピュートノードの主なログファイルは `/var/log/containers/nova/nova-compute.log` です。コンピュートノードの通信で問題が発生した場合に、通常はこのログが診断を開始するのに適した場所です。
- コンピュートノードでメンテナンスを実行する場合には、既存のインスタンスをホストから稼働中のコンピュートノードに移行し、ノードを無効にします。ノードの移行についての詳しい情報は、「[コンピュートノードからのインスタンスの移行](#)」を参照してください。

### 13.7.5. Ceph Storage サービスのエラー

Red Hat Ceph Storage クラスターで発生した問題については、『[Red Hat Ceph Storage Configuration Guide](#)』の「[Logging Configuration Reference](#)」を参照してください。この項には、全 Ceph Storage サービスのログ診断についての情報が記載されています。

## 13.8. アンダークラウドの調整

このセクションでのアドバイスは、アンダークラウドのパフォーマンスを向上に役立たせることが目的です。必要に応じて、推奨事項を実行してください。

- **Identity Service (keystone)** は、他の OpenStack サービスに対するアクセス制御にトークンベースのシステムを使用します。ある程度の期間が過ぎた後には、データベースに未使用のトークンが多数蓄積されます。デフォルトの `cronjob` は毎日トークンをフラッシュします。環境をモニタリングして、トークンのフラッシュ間隔を調節することを推奨します。アンダークラウドの場合は、`crontab -u keystone -e` のコマンドで間隔を調整することができます。この変更は一時的で、`openstack undercloud update` を実行すると、この `cronjob` の設定はデフォルトに戻ってしまう点に注意してください。
- **Heat** は、`openstack overcloud deploy` を実行するたびにデータベースの

**raw\_template** テーブルにある全一時ファイルのコピーを保存します。**raw\_template** テーブルは、過去のテンプレートをすべて保持し、サイズが増加します。**raw\_templates** テーブルにある未使用のテンプレートを削除するには、以下のように、日次の **cron** ジョブを作成して、未使用のまま 1 日以上データベースに存在するテンプレートを消去してください。

```
0 04 * * * /bin/heat-manage purge_deleted -g days 1
```

- **openstack-heat-engine** および **openstack-heat-api** サービスは、一度に過剰なリソースを消費する可能性があります。そのような場合は **/etc/heat/heat.conf** で **max\_resources\_per\_stack=-1** を設定して、Heat サービスを再起動します。

```
$ sudo systemctl restart openstack-heat-engine openstack-heat-api
```

- **director** には、同時にノードをプロビジョニングするリソースが十分でない場合があります。同時に提供できるノード数はデフォルトで 10 個となっています。同時にプロビジョニングするノード数を減らすには、**/etc/nova/nova.conf** の **max\_concurrent\_builds** パラメーターを 10 未満に設定して Nova サービスを再起動します。

```
$ sudo systemctl restart openstack-nova-api openstack-nova-scheduler
```

- **/etc/my.cnf.d/server.cnf** ファイルを編集します。調整が推奨される値は、以下のとおりです。

#### **max\_connections**

データベースに同時接続できる数。推奨の値は 4096 です。

#### **innodb\_additional\_mem\_pool\_size**

データベースがデータのディクショナリーの情報や他の内部データ構造を保存するのに使用するメモリープールサイズ (バイト単位)。デフォルトは通常 8 M ですが、アンダークラウドの理想の値は 20 M です。

#### **innodb\_buffer\_pool\_size**

データベースがテーブルやインデックスデータをキャッシュするメモリー領域つまり、バッファプールのサイズ (バイト単位)。通常デフォルトは 128 M で、アンダークラウドの理想の値は 1000 M です。

#### **innodb\_flush\_log\_at\_trx\_commit**

コミット操作の厳密な ACID 準拠と、コミット関連の I/O 操作を再編成してバッチで実行することによって実現可能なパフォーマンス向上の間のバランスを制御します。1 に設定します。

#### **innodb\_lock\_wait\_timeout**

データベースのトランザクションが、行のロック待ちを中断するまでの時間 (秒単位)。

#### **innodb\_max\_purge\_lag**

この変数は、解析操作が遅れている場合に INSERT、UPDATE、DELETE 操作を遅延させる方法を制御します。10000 に設定します。

#### **innodb\_thread\_concurrency**

同時に実行するオペレーティングシステムのスレッド数の上限。理想的には、各 CPU およびディスクリソースに対して少なくとも 2 つのスレッドを提供します。たとえば、クワッドコア CPU と単一のディスクを使用する場合は、スレッドを 10 個使用します。

- オーバークラウドを作成する際には、Heat に十分なワーカーが配置されているようにします。通常、アンダークラウドに CPU がいくつあるかにより左右されます。ワーカーの数を手動で設定するには、**/etc/heat/heat.conf** ファイルを編集して **num\_engine\_workers** パラメー

ターを必要なワーカー数 (理想は 4) に設定し、Heat エンジンを再起動します。

```
$ sudo systemctl restart openstack-heat-engine
```

## 13.9. SOS レポートの作成

Red Hat に連絡して OpenStack Platform に関するサポートを受ける必要がある場合は、SOS レポートを生成する必要がある場合があります。SOS レポートの作成方法についての詳しい説明は、以下のナレッジベースの記事を参照してください。

- [「How to collect all required logs for Red Hat Support to investigate an OpenStack issue」](#)

## 13.10. アンダークラウドとオーバークラウドの重要なログ

以下のログを使用して、トラブルシューティングの際にアンダークラウドとオーバークラウドの情報を割り出します。

表13.2 アンダークラウドの重要なログ

情報	ログの場所
OpenStack Compute のログ	<code>/var/log/nova/nova-compute.log</code>
OpenStack Compute API の対話	<code>/var/log/nova/nova-api.log</code>
OpenStack Compute コンダクターのログ	<code>/var/log/nova/nova-conductor.log</code>
OpenStack Orchestration ログ	<code>heat-engine.log</code>
OpenStack Orchestration API の対話	<code>heat-api.log</code>
OpenStack Orchestration CloudFormations ログ	<code>/var/log/heat/heat-api-cfn.log</code>
OpenStack Bare Metal コンダクターのログ	<code>ironic-conductor.log</code>
OpenStack Bare Metal API の対話	<code>ironic-api.log</code>
イントロスペクション	<code>/var/log/ironic-inspector/ironic-inspector.log</code>
OpenStack Workflow Engine ログ	<code>/var/log/mistral/engine.log</code>
OpenStack Workflow Executor のログ	<code>/var/log/mistral/executor.log</code>
OpenStack Workflow API の対話	<code>/var/log/mistral/api.log</code>

表13.3 オーバークラウドの重要なログ

情報	ログの場所
cloud-init ログ	<code>/var/log/cloud-init.log</code>
オーバークラウドの設定 (最後に実行した Puppet のサマリー)	<code>/var/lib/puppet/state/last_run_summary.yaml</code>
オーバークラウドの設定 (最後に実行した Puppet からのレポート)	<code>/var/lib/puppet/state/last_run_report.yaml</code>
オーバークラウドの設定 (全 Puppet レポート)	<code>/var/lib/puppet/reports/overcloud-*/*</code>
オーバークラウドの設定 (実行した Puppet 毎の標準出力)	<code>/var/run/heat-config/deployed/*-stdout.log</code>
オーバークラウドの設定 (実行した Puppet 毎の標準エラー出力)	<code>/var/run/heat-config/deployed/*-stderr.log</code>
高可用性ログ	<code>/var/log/pacemaker.log</code>

## 付録A SSL/TLS 証明書の設定

アンダークラウドがパブリックエンドポイントの通信に SSL/TLS を使用するように設定できます。ただし、独自の認証局で発行した SSL 証明書を使用する場合には、その証明書には以下の項に記載する設定のステップが必要です。



### 注記

オーバークラウドの SSL/TLS 証明書作成については、『**Advanced Overcloud Customization**』ガイドの「[Enabling SSL/TLS on Overcloud Public Endpoints](#)」を参照してください。

### A.1. 署名ホストの初期化

署名ホストとは、新規証明書を生成し、認証局を使用して署名するホストです。選択した署名ホスト上で SSL 証明書を作成したことがない場合には、ホストを初期化して新規証明書に署名できるようにする必要があります。

`/etc/pki/CA/index.txt` ファイルは、すべての署名済み証明書の記録を保管します。このファイルが存在しているかどうかを確認してください。存在していない場合には、空のファイルを作成します。

```
$ sudo touch /etc/pki/CA/index.txt
```

`/etc/pki/CA/serial` ファイルは、次に署名する証明書に使用する次のシリアル番号を特定します。このファイルが存在するかどうかを確認し、存在しない場合には、新規ファイルを作成して新しい開始値を指定します。

```
$ sudo echo '1000' | sudo tee /etc/pki/CA/serial
```

### A.2. 認証局の作成

通常、SSL/TLS 証明書の署名には、外部の認証局を使用します。場合によっては、独自の認証局を使用する場合があります。たとえば、内部のみの認証局を使用するように設定する場合などです。

たとえば、キーと証明書のペアを生成して、認証局として機能するようにします。

```
$ openssl genrsa -out ca.key.pem 4096
$ openssl req -key ca.key.pem -new -x509 -days 7300 -extensions v3_ca -
out ca.crt.pem
```

`openssl req` コマンドは、認証局に関する特定の情報を要求します。それらの情報を指定してください。

これで、`ca.crt.pem` という名前の認証局ファイルが作成されます。

### A.3. クライアントへの認証局の追加

SSL/TLS を使用して通信することを目的としている外部のクライアントの場合は、Red Hat OpenStack Platform 環境にアクセスする必要のある各クライアントに認証局ファイルをコピーします。クライアントへのコピーが完了したら、そのクライアントで以下のコマンドを実行して、認証局のトラストバンドルに追加します。

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/  
$ sudo update-ca-trust extract
```

## A.4. SSL/TLS キーの作成

以下のコマンドを実行して、SSL/TLS キー (**server.key.pem**) を生成します。このキーは、さまざまな段階で、アンダークラウドとオーバークラウドの証明書を作成するのに使用します。

```
$ openssl genrsa -out server.key.pem 2048
```

## A.5. SSL/TLS 証明書署名要求の作成

次の手順では、アンダークラウドおよびオーバークラウドのいずれかの証明書署名要求を作成します。

カスタマイズするデフォルトの **OpenSSL** 設定ファイルをコピーします。

```
$ cp /etc/pki/tls/openssl.cnf .
```

カスタムの **openssl.cnf** ファイルを編集して、**director** に使用する SSL パラメーターを設定します。変更するパラメーターの種別には以下のような例が含まれます。

```
[req]  
distinguished_name = req_distinguished_name  
req_extensions = v3_req  
  
[req_distinguished_name]  
countryName = Country Name (2 letter code)  
countryName_default = AU  
stateOrProvinceName = State or Province Name (full name)  
stateOrProvinceName_default = Queensland  
localityName = Locality Name (eg, city)  
localityName_default = Brisbane  
organizationalUnitName = Organizational Unit Name (eg, section)  
organizationalUnitName_default = Red Hat  
commonName = Common Name  
commonName_default = 192.168.0.1  
commonName_max = 64  
  
[ v3_req ]  
# Extensions to add to a certificate request  
basicConstraints = CA:FALSE  
keyUsage = nonRepudiation, digitalSignature, keyEncipherment  
subjectAltName = @alt_names  
  
[alt_names]  
IP.1 = 192.168.0.1  
DNS.1 = instack.localdomain  
DNS.2 = vip.localdomain  
DNS.3 = 192.168.0.1
```

**commonName\_default** は以下のいずれか1つに設定します。

- IP アドレスを使用して SSL/TLS 経由でアクセスする場合には、**undercloud.conf** で **undercloud\_public\_vip** パラメーターを使用します。
- 完全修飾ドメイン名を使用して SSL/TLS でアクセスする場合には、代わりにドメイン名を使用します。

**alt\_names** セクションを編集して、以下のエントリを追加します。

- **IP:** SSL 経由で **director** にアクセスするためのクライアントの IP アドレス一覧
- **DNS:** SSL 経由で **director** にアクセスするためのクライアントのドメイン名一覧。**alt\_names** セクションの最後に DNS エントリとしてパブリック API の IP アドレスも追加します。

**openssl.cnf** に関する詳しい情報については、**man openssl.cnf** を実行します。

次のコマンドを実行し、手順1で作成したキーストアより公開鍵を使用して証明書署名要求を生成します (**server.csr.pem**)。

```
$ openssl req -config openssl.cnf -key server.key.pem -new -out
server.csr.pem
```

「[SSL/TLS キーの作成](#)」で作成した SSL/TLS キーを **-key** オプションで必ず指定してください。

次の項では、この **server.csr.pem** ファイルを使用して SSL/TLS 証明書を作成します。

## A.6. SSL/TLS 証明書の作成

以下のコマンドで、アンダークラウドまたはオーバークラウドの証明書を作成します。

```
$ sudo openssl ca -config openssl.cnf -extensions v3_req -days 3650 -in
server.csr.pem -out server.crt.pem -cert ca.crt.pem -keyfile ca.key.pem
```

上記のコマンドでは、以下のオプションを使用しています。

- **v3** 拡張機能を指定する設定ファイル。この値は、「**-config**」オプションとして追加します。
- 認証局を介して証明書を生成し、署名するために、「[SSL/TLS 証明書署名要求の作成](#)」で設定した証明書署名要求。この値は、「**-in**」オプションで設定します。
- 証明書への署名を行う、「[認証局の作成](#)」で作成した認証局。この値は **-cert** オプションとして追加します。
- 「[認証局の作成](#)」で作成した認証局の秘密鍵。この値は **-keyfile** オプションとして追加します。

このコマンドを実行すると、**server.crt.pem** という名前の証明書が作成されます。この証明書は、「[SSL/TLS キーの作成](#)」で作成した SSL/TLS キーとともに使用して SSL/TLS を有効にします。

## A.7. アンダークラウドで証明書を使用する場合

以下のコマンドを実行して、証明書とキーを統合します。

```
$ cat server.crt.pem server.key.pem > undercloud.pem
```

-

このコマンドにより、**undercloud.pem**が作成されます。**undercloud.conf** ファイルの **undercloud\_service\_certificate** オプションにこのファイルの場所を指定します。このファイルには、HAProxy ツールが読み取ることができるように、特別な SELinux コンテキストも必要です。以下の例を目安にしてください。

```
$ sudo mkdir /etc/pki/instack-certs
$ sudo cp ~/undercloud.pem /etc/pki/instack-certs/
$ sudo semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
$ sudo restorecon -R /etc/pki/instack-certs
```

**undercloud.conf** ファイルの **undercloud\_service\_certificate** オプションに **undercloud.pem**の場所を追記します。以下に例を示します。

```
undercloud_service_certificate = /etc/pki/instack-certs/undercloud.pem
```

また、「[認証局の作成](#)」で作成した認証局をアンダークラウドの信頼済み認証局の一覧に認証局を追加して、アンダークラウド内の異なるサービスが認証局にアクセスできるようにします。

```
$ sudo cp ca.crt.pem /etc/pki/ca-trust/source/anchors/
$ sudo update-ca-trust extract
```

「[director の設定](#)」の手順に従ってアンダークラウドのインストールを続行します。

## 付録B 電源管理ドライバー

IPMI は、`director` が電源管理制御に使用する主要な手法ですが、`director` は他の電源管理タイプもサポートします。この付録では、サポートされる電源管理機能の一覧を提供します。「[オーバークラウドへのノードの登録](#)」には、以下の電源管理設定を使用します。

### B.1. REDFISH

Distributed Management Task Force (DMTF) によって開発された IT インフラストラクチャー向けの標準 RESTful API

#### `pm_type`

このオプションを `redfish` に設定します。

#### `pm_user; pm_password`

Redfish のユーザー名およびパスワード

#### `pm_addr`

Redfish コントローラーの IP アドレス

#### `pm_system_id`

システムリソースの正規のパス。このパスには、そのシステムの root サービス、バージョン、パス/一意 ID が含まれる必要があります (例: `/redfish/v1/Systems/CX34R87`)。

### B.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。

#### `pm_type`

このオプションを `idrac` に設定します。

#### `pm_user; pm_password`

DRAC のユーザー名およびパスワード

#### `pm_addr`

DRAC ホストの IP アドレス

### B.3. 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。

#### `pm_type`

このオプションを `ilo` に設定します。

#### `pm_user; pm_password`

iLO のユーザー名およびパスワード

#### `pm_addr`

iLO インターフェースの IP アドレス

- このドライバーを有効化するには、`undercloud.conf` の `enabled_hardware_types` オプションに `ilo` を追加してから、`openstack undercloud install` を再実行します。

- また **director** では、iLO 向けに追加のユーティリティーセットが必要です。 **python-proliantutils** パッケージをインストールして **openstack-ironic-conductor** サービスを再起動します。

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

- HP ノードは、正常にイントロスペクションするには 2015 年のファームウェアバージョンが必要です。ファームウェアバージョン 1.85 (2015 年 5 月 13 日) を使用したノードで、**director** は正常にテストされています。
- 共有 iLO ポートの使用はサポートされません。

## B.4. CISCO UNIFIED COMPUTING SYSTEM (UCS)

Cisco の UCS は、コンピュータ、ネットワーク、ストレージのアクセス、仮想化リソースを統合するデータセンタープラットフォームです。このドライバーは、UCS に接続されたベアメタルシステムの電源管理を重視しています。

### pm\_type

このオプションを **cisco-ucs-managed** に設定します。

### pm\_user; pm\_password

UCS のユーザー名およびパスワード

### pm\_addr

UCS インターフェースの IP アドレス

### pm\_service\_profile

使用する UCS サービスプロファイル。通常 **org-root/ls-[service\_profile\_name]** の形式を取ります。以下に例を示します。

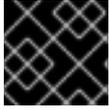
```
"pm_service_profile": "org-root/ls-Nova-1"
```

- このドライバーを有効化するには、**undercloud.conf** の **enabled\_hardware\_types** オプションに **cisco-ucs-managed** を追加してから、**openstack undercloud install** を再実行します。
- また **director** では、iLO 向けに追加のユーティリティーセットが必要です。 **python-UcsSdk** パッケージをインストールして **openstack-ironic-conductor** サービスを再起動します。

```
$ sudo yum install python-UcsSdk
$ sudo systemctl restart openstack-ironic-conductor.service
```

## B.5. FUJITSU INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)

Fujitsu の iRMC は、LAN 接続と拡張された機能を統合した Baseboard Management Controller (BMC) です。このドライバーは、iRMC に接続されたベアメタルシステムの電源管理にフォーカスしています。

**重要**

iRMC S4 以降のバージョンが必要です。

**pm\_type**

このオプションを **irmc** に設定します。

**pm\_user; pm\_password**

iRMC インターフェースのユーザー名とパスワード

**pm\_addr**

iRMC インターフェースの IP アドレス

**pm\_port (オプション)**

iRMC の操作に使用するポート。デフォルトは **443** です。

**pm\_auth\_method (オプション)**

iRMC 操作の認証方法。 **basic** または **digest** を使用します。デフォルトは **basic** です。

**pm\_client\_timeout (オプション)**

iRMC 操作のタイムアウト (秒単位)。デフォルトは **60** 秒です。

**pm\_sensor\_method (オプション)**

センサーデータの取得方法。 **ipmitool** または **scci** です。デフォルトは **ipmitool** です。

- このドライバーを有効化するには、 **undercloud.conf** の **enabled\_hardware\_types** オプションに **irmc** を追加してから、 **openstack undercloud install** を再実行します。
- センサーの方法として **SCCI** を有効にした場合には、 **director** には、追加のユーティリティセットも必要です。 **python-scciclient** パッケージをインストールして、 **openstack-ironic-conductor** サービスを再起動します。

```
$ yum install python-scciclient
$ sudo systemctl restart openstack-ironic-conductor.service
```

## B.6. VIRTUAL BASEBOARD MANAGEMENT CONTROLLER (VBMC)

**director** は仮想マシンを KVM ホスト上のノードとして使用することができます。エミュレーションされた IPMI デバイスを使用して電源管理を制御します。これにより、「[オーバークラウドへのノードの登録](#)」からの標準の IPMI パラメーターを使用することができますが、仮想ノードに対して使用することになります。

**重要**

このオプションでは、ベアメタルノードの代わりに仮想マシンを使用するので、テストおよび評価の目的でのみ利用することができます。Red Hat OpenStack Platform のエンタープライズ環境には推奨しません。

### KVM ホストの設定

KVM ホスト上で、OpenStack Platform リポジトリを有効化して **python-virtualbmc** パッケージをインストールします。

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
$ sudo yum install -y python-virtualbmc
```

**vbmc** コマンドを使用して、各仮想マシン用に仮想 Baseboard Management Controller (BMC) を作成します。たとえば、**Node01** および **Node02** という名前の仮想マシンに BMC を作成する場合は、以下のコマンドを実行します。

```
$ vbmc add Node01 --port 6230 --username admin --password p455w0rd!
$ vbmc add Node02 --port 6231 --username admin --password p455w0rd!
```

これにより、各 BMC にアクセスするポートが定義され、各 BMC の認証情報が設定されます。



### 注記

仮想マシンごとに異なるポートを使用してください。1025 未満のポート番号には、システムの root 権限が必要です。

以下のコマンドで各 BMC を起動します。

```
$ vbmc start Node01
$ vbmc start Node02
```



### 注記

KVM ホストの再起動後には、このステップを繰り返す必要があります。

## ノードの登録

ノードの登録ファイル (`/home/stack/instackenv.json`) で以下のパラメーターを使用します。

### **pm\_type**

このオプションを **ipmi** に設定します。

### **pm\_user; pm\_password**

ノードの仮想 BMC デバイスの IPMI ユーザー名とパスワード

### **pm\_addr**

ノードが含まれている KVM ホストの IP アドレス

### **pm\_port**

KVM ホスト上の特定のノードにアクセスするポート

### **mac**

ノード上のネットワークインターフェースの MAC アドレス一覧。各システムのプロビジョニング NIC の MAC アドレスのみを使用します。

例:

```
{
  "nodes": [
    {
      "pm_type": "pxe_ipmitool",
      "mac": [
        "aa:aa:aa:aa:aa:aa"
      ]
    }
  ]
}
```

```

    ],
    "pm_user": "admin",
    "pm_password": "p455w0rd!",
    "pm_addr": "192.168.0.1",
    "pm_port": "6230",
    "name": "Node01"
  },
  {
    "pm_type": "pxe_ipmitool",
    "mac": [
      "bb:bb:bb:bb:bb:bb"
    ],
    "pm_user": "admin",
    "pm_password": "p455w0rd!",
    "pm_addr": "192.168.0.1",
    "pm_port": "6231",
    "name": "Node02"
  }
]
}

```

### 既存のノードの移行

既存のノードは、非推奨の `pxe_ssh` を使用する設定から新しい仮想 BMC の方法を使用するように移行することができます。以下のコマンドは、ノードが `pxe_ipmitool` ドライバーを使用するようにし、そのパラメーターを以下のように設定します。

```

openstack baremetal node set Node01 \
  --driver pxe_ipmitool \
  --driver-info ipmi_address=192.168.0.1 \
  --driver-info ipmi_port=6230 \
  --driver-info ipmi_username="admin" \
  --driver-info ipmi_password="p455w0rd!"

```

## B.7. RED HAT VIRTUALIZATION

このドライバーにより、RESTful API を介して、Red Hat Virtualization 内の仮想マシンを制御することができますようになります。

### `pm_type`

このオプションを `staging-ovirt` に設定します。

### `pm_user; pm_password`

Red Hat Virtualization 環境のユーザー名とパスワード。ユーザー名には認証プロバイダーも含まれます (例: `admin@internal`)。

### `pm_addr`

Red Hat Virtualization REST API の IP アドレス

### `pm_vm_name`

制御する仮想マシンの名前

### `mac`

ノード上のネットワークインターフェースの MAC アドレス一覧。各システムのプロビジョニング NIC の MAC アドレスのみを使用します。

- このドライバーを有効化するには、`undercloud.conf` の `enabled_hardware_types` オプションに `staging-ovirt` を追加してから、`openstack undercloud install` を再実行します。

## B.8. フェイクドライバー

このドライバーは、電源管理なしにベアメタルデバイスを使用する方法を提供します。これは、`director` が登録されたベアメタルデバイスを制御しないので、イントロスペクションとデプロイの特定の時点で手動で電源をコントロールする必要があることを意味します。



### 重要

このオプションは、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。

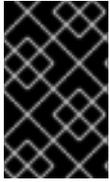
### pm\_type

このオプションを `fake` に設定します。

- このドライバーは、電源管理を制御しないので、認証情報は使用しません。
- このドライバーを有効化するには、`undercloud.conf` の `enabled_hardware_types` オプションに `staging-ovirt` を追加してから、`openstack undercloud install` を再実行します。
- ノードのイントロスペクションを実行する際には、`openstack overcloud node introspect` コマンドを実行した後にノードの電源を手動でオフにします。
- オーバークラウドのデプロイ実行時には、`ironic node-list` コマンドでノードのステータスを確認します。ノードのステータスが `deploying` から `deploy wait-callback` に変わるまで待ってから、手動でノードの電源をオンにします。
- オーバークラウドのプロビジョニングプロセスが完了したら、ノードを再起動します。プロビジョニングが完了したかどうかをチェックするには、`ironic node-list` コマンドでノードのステータスをチェックし、`active` に変わるのを待ってから、すべてのオーバークラウドノードを手動で再起動します。

## 付録C 完全なディスクイメージ

メインのオーバークラウドイメージは、フラットパーティションイメージです。これは、パーティション情報またはブートローダーがイメージ自体に含まれていないことを意味します。**director** は、起動時に別のカーネルと **ramdisk** を使用し、オーバークラウドイメージをディスクに書き込む際には基本的なパーティションレイアウトを作成しますが、パーティションレイアウト、ブートローダー、および強化されたセキュリティー機能が含まれる完全なディスクイメージを作成することが可能です。



### 重要

以下のプロセスでは、**director** のイメージ構築機能を使用します。**Red Hat** では、本項に記載の指針に従ったイメージ構築のみをサポートしています。これらとは異なる仕様でのカスタムのイメージ構築はサポートされていません。

セキュリティーが強化されたイメージには、セキュリティーが重要な機能となる **Red Hat OpenStack Platform** のデプロイメントに必要な追加のセキュリティー対策が含まれます。イメージをセキュアにするためには、以下のような推奨事項があります。

- **/tmp** ディレクトリーは、別のボリュームまたはパーティションにマウントされ、**rw**、**nosuid**、**nodev**、**noexec**、**relatime** のフラグが付きます。
- **/var**、**/var/log**、**/var/log/audit** のディレクトリーは、別のボリュームまたはパーティションにマウントされ、**rw** および **relatime** のフラグが付きます。
- **/home** ディレクトリーは、別のパーティションまたはボリュームにマウントされ、**rw**、**nodev**、**relatime** のフラグが付きます。
- **GRUB\_CMDLINE\_LINUX** の設定に以下の変更を加えます。
  - 監査を有効にするには、**audit=1** を追加して、追加のカーネルブートフラグを付けます。
  - **nousb** を追加して、ブートローダー設定を使用した **USB** のカーネルサポートを無効にします。
  - **crashkernel=auto** を設定して、セキュアでないブートフラグを削除します。
- セキュアでないモジュール (**usb-storage**、**cramfs**、**freevxfs**、**jffs2**、**hfs**、**hfsplus**、**squashfs**、**udf**、**vfat**) をブラックリストに登録して、読み込まれないようにします。
- イメージから、セキュアでないパッケージ (**kexec-tools** によりインストールされた **kdump** と **telnet**) を削除します。
- セキュリティーに必要な新しい **screen** パッケージを追加します。

セキュリティー強化されたイメージを構築するには、以下の手順を実行する必要があります。

1. ベースの **Red Hat Enterprise Linux 7** イメージをダウンロードします。
2. 登録固有の環境変数を設定します。
3. パーティションスキーマとサイズを変更してイメージをカスタマイズします。
4. イメージを作成します。

5. そのイメージをデプロイメントにアップロードします。

以下の項では、これらのタスクを実行する手順について詳しく説明します。

## C.1. ベースのクラウドイメージのダウンロード

完全なディスクイメージを構築する前に、ベースとして使用する Red Hat Enterprise Linux の既存のクラウドイメージをダウンロードする必要があります。Red Hat カスタマーポータルにナビゲートして、ダウンロードする KVM ゲストイメージを選択します。たとえば、最新の Red Hat Enterprise Linux の KVM ゲストイメージは以下のページにあります。

- [「Red Hat Enterprise Linux Server のインストーラーおよびイメージ」](#)

## C.2. 環境変数の設定

完全なディスクイメージの構築プロセスとして、**director** にはベースイメージと、新規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

イメージの構築プロセスにより、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構築プロセスを完了すると登録を解除します。

セキュリティ強化された完全なイメージを構築するには、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サブスクリプション管理のデフォルトのプロセスでは <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

1つの組織内の1つ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** で有効化されます。セキュリティ強化された完全なディスクイメージには以下のリポジトリを使用します。

- **rhel-7-server-rpms**
- **rhel-7-server-extras-rpms**
- **rhel-ha-for-rhel-7-server-rpms**
- **rhel-7-server-optional-rpms**
- **rhel-7-server-openstack-13-rpms**

### 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。

Red Hat カスタマーポータルにローカルの QCOW2 をイメージ一時的に登録するための環境変数のセットをエクスポートする一連のコマンドの例を以下に示します。

```
$ export DIB_LOCAL_IMAGE=./rhel-server-7.5-x86_64-kvm.qcow2
$ export REG_METHOD=portal
$ export REG_USER="[your username]"
$ export REG_PASSWORD="[your password]"
$ export REG_REPOS="rhel-7-server-rpms \
rhel-7-server-extras-rpms \
rhel-ha-for-rhel-7-server-rpms \
rhel-7-server-optional-rpms \
rhel-7-server-openstack-13-rpms"
```

## C.3. ディスクレイアウトのカスタマイズ

デフォルトでは、セキュリティが強化されたイメージのサイズは **20G** で、事前定義されたパーティショニングサイズを使用します。ただし、オーバークラウドのコンテナイメージを収容するには、パーティショニングレイアウトを若干変更する必要があります。

パーティション	以前のサイズ	新しいサイズ
/	6G	8G
/tmp	1G	1G
/var	7G	10G

パーティション	以前のサイズ	新しいサイズ
/var/log	5G	5G
/var/log/audit	900M	900M
/home	100M	100M
<b>Total</b>	<b>20G</b>	<b>25G</b>

これにより、イメージサイズが **25G** に拡張されます。また、必要に応じて、パーティショニングレイアウトとディスクサイズをさらに変更することも可能です。

パーティショニングレイアウトとディスクサイズを変更するには、以下の手順に従ってください。

- **DIB\_BLOCK\_DEVICE\_CONFIG** 環境変数を使用してパーティショニングスキーマを変更します。
- **DIB\_IMAGE\_SIZE** 環境変数を更新して、イメージのグローバルサイズを変更します。

### C.3.1. パーティショニングスキーマの変更

パーティショニングスキーマを編集して、パーティショニングサイズを変更したり、新規パーティションの作成や既存のパーティションの削除を行うことができます。新規パーティショニングスキーマは以下の環境変数で定義することができます。

```
$ export DIB_BLOCK_DEVICE_CONFIG='<yaml_schema_with_partitions>'
```

以下の YAML 構成は、オーバークラウドのコンテナイメージをプルするのに十分なスペースを提供する、変更後のパーティショニングレイアウトを示しています。

```
export DIB_BLOCK_DEVICE_CONFIG=''
- local_loop:
  name: image0
- partitioning:
  base: image0
  label: mbr
  partitions:
    - name: root
      flags: [ boot,primary ]
      size: 8G
      mkfs:
        type: xfs
        label: "img-rootfs"
        mount:
          mount_point: /
          fstab:
            options: "rw,relatime"
            fck-passno: 1
    - name: tmp
      size: 1G
      mkfs:
```

```

    type: xfs
    mount:
      mount_point: /tmp
      fstab:
        options: "rw,nosuid,nodev,noexec,relatime"
- name: var
  size: 10G
  mkfs:
    type: xfs
    mount:
      mount_point: /var
      fstab:
        options: "rw,relatime"
- name: log
  size: 5G
  mkfs:
    type: xfs
    mount:
      mount_point: /var/log
      fstab:
        options: "rw,relatime"
- name: audit
  size: 900M
  mkfs:
    type: xfs
    mount:
      mount_point: /var/log/audit
      fstab:
        options: "rw,relatime"
- name: home
  size: 100M
  mkfs:
    type: xfs
    mount:
      mount_point: /home
      fstab:
        options: "rw,nodev,relatime"
...

```

サンプルの YAML コンテンツをイメージのパーティションスキーマのベースとして使用します。パーティションサイズとレイアウトを必要に応じて変更します。



### 注記

パーティションサイズはデプロイメント後には**変更できない**ので、正しいパーティションサイズを定義してください。

### C.3.2. イメージサイズの変更

変更後のパーティショニングスキーマの合計は、デフォルトのディスクサイズ (20G) を超える可能性があります。そのような場合には、イメージサイズを変更する必要がある場合があります。イメージサイズを変更するには、イメージの作成に使用した設定ファイルを編集します。

`/usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-images.yaml` のコピーを作成します。

```
# cp /usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-
images.yaml \
/home/stack/overcloud-hardened-images-custom.yaml
```

設定ファイルで **DIB\_IMAGE\_SIZE** を編集して、必要に応じて値を調整します。

```
...

environment:
DIB_PYTHON_VERSION: '2'
DIB_MODPROBE_BLACKLIST: 'usb-storage cramfs freevxfs jffs2 hfs hfsplus
squashfs udf vfat bluetooth'
DIB_BOOTLOADER_DEFAULT_CMDLINE: 'nofb nomodeset vga=normal console=tty0
console=ttyS0,115200 audit=1 nusb'
DIB_IMAGE_SIZE: '25' ❶
COMPRESS_IMAGE: '1'
```

❶ この値は、新しいディスクサイズの合計に応じて調整してください。

このファイルを保存します。



### 重要

**director** がオーバークラウドをデプロイする際には、オーバークラウドイメージの RAW バージョンを作成します。これは、アンダークラウドに、その RAW イメージを収容するのに必要な空き容量がなければならないことを意味します。たとえば、セキュリティー強化されたイメージのサイズを **40G** に増やした場合には、アンダークラウドのハードディスクに **40G** の空き容量が必要となります。

## C.4. セキュリティー強化された完全なディスクイメージの作成

環境変数を設定してイメージをカスタマイズした後は、**openstack overcloud image build** コマンドを使用してイメージを作成します。

```
# openstack overcloud image build \
--image-name overcloud-hardened-full \
--config-file /home/stack/overcloud-hardened-images-custom.yaml \ ❶
--config-file /usr/share/openstack-tripleo-common/image-yaml/overcloud-
hardened-images-rhel7.yaml
```

❶ これは、「**イメージサイズの変更**」の新規ディスクサイズが含まれたカスタム設定ファイルです。異なるカスタムディスクサイズを使用していない場合には、代わりに元の **/usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-images.yaml** ファイルを使用してください。

これにより、**overcloud-hardened-full.qcow2** という名前のイメージが作成され、必要なセキュリティー機能がすべて含まれます。

## C.5. セキュリティー強化された完全なディスクイメージのアップロード

OpenStack Image (glance) サービスにイメージをアップロードして、Red Hat OpenStack Platform director から使用を開始します。セキュリティ強化されたイメージをアップロードするには、以下の手順を実行してください。

1. 新規作成されたイメージの名前を変更し、イメージディレクトリーに移動します。

```
# mv overcloud-hardened-full.qcow2 ~/images/overcloud-full.qcow2
```

2. オーバークラウドの古いイメージをすべて削除します。

```
# openstack image delete overcloud-full
# openstack image delete overcloud-full-initrd
# openstack image delete overcloud-full-vmlinuz
```

3. 新規オーバークラウドイメージをアップロードします。

```
# openstack overcloud image upload --image-path /home/stack/images -
-whole-disk
```

既存のイメージをセキュリティ強化されたイメージに置き換える場合は、**--update-existing** フラグを使用します。これにより、元の **overcloud-full** イメージは、自分で作成した、セキュリティ強化された新しいイメージに置き換えられます。

## 付録D 代替ブートモード

ノードのデフォルトブートモードは、iPXE を使用した BIOS です。以下の項では、ノードのプロビジョニングおよび検査の際に **director** が使用する代替ブートモードについて説明します。

### D.1. 標準の PXE

iPXE ブートプロセスは、HTTP を使用してイントロスペクションおよびデプロイメントのイメージをブートします。旧システムは、TFTP を介してブートする標準の PXE ブートのみをサポートしている場合があります。

iPXE から PXE に変更するには、**director** ホスト上の **undercloud.conf** ファイルを編集して、**ipxe\_enabled** を **False** に設定します。

```
ipxe_enabled = False
```

このファイルを保存して、アンダークラウドのインストールを実行します。

```
$ openstack undercloud install
```

このプロセスに関する詳しい情報は、「[Changing from iPXE to PXE in Red Hat OpenStack Platform director](#)」の記事を参照してください。

### D.2. UEFI ブートモード

デフォルトのブートモードは、レガシー BIOS モードです。新しいシステムでは、レガシー BIOS モードの代わりに UEFI ブートモードが必要な可能性があります。その場合には、**undercloud.conf** ファイルで以下のように設定します。

```
ipxe_enabled = True
inspection_enable_uefi = True
```

このファイルを保存して、アンダークラウドのインストールを実行します。

```
$ openstack undercloud install
```

登録済みの各ノードのブートモードを **uefi** に設定します。たとえば、**capabilities** プロパティに **boot\_mode** パラメーターを追加する場合や既存のパラメーターを置き換える場合には、以下のコマンドを実行します。

```
$ NODE=<NODE NAME OR ID> ; openstack baremetal node set --property
capabilities="boot_mode:uefi,$(openstack baremetal node show $NODE -f json
-c properties | jq -r .properties.capabilities | sed "s/boot_mode:
[^\,]*, //g")" $NODE
```



#### 注記

このコマンドで **profile** および **boot\_option** のキータブが保持されていることを確認してください

また、各フレーバーのブートモードを **uefi** に設定します。以下に例を示します。

```
$ openstack flavor set --property capabilities:boot_mode='uefi' control
```

## 付録E プロファイルの自動タグ付け

イントロスペクションプロセスでは、一連のベンチマークテストを実行します。**director** は、これらのテストからデータを保存します。このデータをさまざまな方法で使用するポリシーセットを作成することができます。以下に例を示します。

- ポリシーにより、パフォーマンスの低いノードまたは不安定なノードを特定して、オーバークラウドで使用されないように隔離することができます。
- ポリシーにより、ノードを自動的に特定のプロファイルにタグ付けするかどうかを定義することができます。

### E.1. ポリシーファイルの構文

ポリシーファイルは、JSON 形式で、ルールセットが記載されます。各ルールは、**description**、**condition**、**action** を定義します。

#### 説明

これは、プレーンテキストで記述されたルールの説明です。

#### 例:

```
"description": "A new rule for my node tagging policy"
```

#### conditions

**condition** は、以下のキー/値のパターンを使用して評価を定義します。

#### field

評価するフィールドを定義します。フィールドの種別については、「[プロファイルの自動タグ付けのプロパティ](#)」を参照してください。

#### op

評価に使用する演算を定義します。これには、以下が含まれます。

- **eq**: 等しい
- **ne**: 等しくない
- **lt**: より小さい
- **gt**: より大きい
- **le**: より小さいか等しい
- **ge**: より大きいか等しい
- **in-net**: IP アドレスが指定のネットワーク内にあることをチェックします。
- **matches**: 指定の正規表現と完全に一致する必要があります。
- **contains**: 値には、指定の正規表現が含まれる必要があります。
- **is-empty**: フィールドが空欄であることをチェックします。

## invert

評価の結果をインバージョン (反転) するかどうかを定義するブール値

## multiple

複数の結果が存在する場合に、使用する評価を定義します。これには、以下が含まれます。

- **any:** いずれかの結果が一致する必要があります。
- **all:** すべての結果が一致する必要があります。
- **first:** 最初の結果が一致する必要があります。

## value

評価内の値を定義します。フィールドと演算の結果でその値となった場合には、条件は **true** の結果を返します。そうでない場合には、条件は **false** の結果を返します。

例:

```
"conditions": [
  {
    "field": "local_gb",
    "op": "ge",
    "value": 1024
  }
],
```

## Actions

**action** は、**condition** が **true** として返された場合に実行されます。これには、**action** キーと、**action** の値に応じて追加のキーが使用されます。

- **fail:** イントロスペクションが失敗します。失敗のメッセージには、**message** パラメーターが必要です。
- **set-attribute:** Ironic ノードで属性を設定します。Ironic の属性へのパス (例: `/driver_info/ipmi_address`) である **path** フィールドと、設定する **value** が必要です。
- **set-capability:** Ironic ノードでケイパビリティを設定します。新しいケイパビリティに応じた名前と値を指定する **name** および **value** のフィールドが必要です。同じケイパビリティの既存の値は置き換えられます。たとえば、これを使用してノードのプロファイルを定義します。
- **extend-attribute:** **set-attribute** と同じですが、既存の値を一覧として扱い、その一覧に値を追記します。オプションの **unique** パラメーターが **True** に設定すると、対象の値がすでに一覧に含まれている場合には何も追加されません。

例:

```
"actions": [
  {
    "action": "set-capability",
    "name": "profile",
    "value": "swift-storage"
  }
]
```

## E.2. ポリシーファイルの例

以下は、適用するイントロスペクションルールを記載した JSON ファイル (**rules.json**) の一例です。

```
[
  {
    "description": "Fail introspection for unexpected nodes",
    "conditions": [
      {
        "op": "lt",
        "field": "memory_mb",
        "value": 4096
      }
    ],
    "actions": [
      {
        "action": "fail",
        "message": "Memory too low, expected at least 4 GiB"
      }
    ]
  },
  {
    "description": "Assign profile for object storage",
    "conditions": [
      {
        "op": "ge",
        "field": "local_gb",
        "value": 1024
      }
    ],
    "actions": [
      {
        "action": "set-capability",
        "name": "profile",
        "value": "swift-storage"
      }
    ]
  },
  {
    "description": "Assign possible profiles for compute and controller",
    "conditions": [
      {
        "op": "lt",
        "field": "local_gb",
        "value": 1024
      },
      {
        "op": "ge",
        "field": "local_gb",
        "value": 40
      }
    ],
    "actions": [
      {
        "action": "set-capability",
```

```

    "name": "compute_profile",
    "value": "1"
  },
  {
    "action": "set-capability",
    "name": "control_profile",
    "value": "1"
  },
  {
    "action": "set-capability",
    "name": "profile",
    "value": null
  }
]
}
]

```

上記の例は、3つのルールで構成されています。

- メモリーが 4096 MiB 未満の場合には、イントロスペクションが失敗します。このようなルールを適用して、クラウドに含まれるべきではないノードを除外することができます。
- ハードドライブのサイズが 1 TiB 以上のノードの場合は **swift-storage** プロファイルが無条件で割り当てられます。
- ハードドライブが 1 TiB 未満だが 40 GiB を超えているノードは、コンピューターノードまたはコントロールノードのいずれかに割り当てることができます。 **openstack overcloud profiles match** コマンドを使用して、後で最終選択できるように 2つのケイパビリティ (**compute\_profile** と **control\_profile**) を割り当てています。この設定が機能するように、既存のプロファイルのケイパビリティは削除しています。削除しなかった場合には、そのケイパビリティが優先されてしまいます。

他のノードは変更されません。



### 注記

イントロスペクションルールを使用して **profile** 機能を割り当てる場合は常に、既存の値よりこの割り当てた値が優先されます。ただし、既存のプロファイル機能があるノードについては、**[PROFILE]\_profile** 機能は無視されます。

## E.3. ポリシーファイルのインポート

以下のコマンドで、ポリシーファイルを **director** にインポートします。

```
$ openstack baremetal introspection rule import rules.json
```

次にイントロスペクションプロセスを実行します。

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

イントロスペクションが完了したら、ノードとノードに割り当てられたプロファイルを確認します。

```
$ openstack overcloud profiles list
```

イントロスペクションルールで間違いがあった場合には、すべてを削除できます。

```
$ openstack baremetal introspection rule purge
```

#### E.4. プロファイルの自動タグ付けのプロパティ

プロファイルの自動タグ付けは、各条件の **field** の属性に対する以下のノードプロパティを評価します。

プロパティ	説明
memory_mb	ノードのメモリーサイズ (MB)
cpus	ノードの CPU の合計コア数
cpu_arch	ノードの CPU のアーキテクチャー
local_gb	ノードのルートディスクのストレージの合計容量。 ノードのルートディスクの設定についての詳しい説明は、「 <a href="#">ノードのルートディスクの定義</a> 」を参照してください。

## 付録F セキュリティーの強化

以下の項では、アンダークラウドのセキュリティーを強化するための推奨事項について説明します。

### F.1. HAPROXY の SSL/TLS の暗号およびルールの変更

アンダークラウドで SSL/TLS を有効化した場合には (「[director の設定パラメーター](#)」を参照)、HAProxy 設定を使用する SSL/TLS の暗号とルールを強化することをお勧めします。これにより、[POODLE TLS 脆弱性](#) などの SSL/TLS の脆弱性を回避することができます。

`hieradata_override` のアンダークラウド設定オプションを使用して、以下の `hieradata` を設定します。

#### `tripleo::haproxy::ssl_cipher_suite`

HAProxy で使用する暗号スイート

#### `tripleo::haproxy::ssl_options`

HAProxy で使用する SSL/TLS ルール

たとえば、以下のような暗号およびルールを使用することができます。

- 暗号: `ECDHE-ECDSA-CHACHA20-POLY1305: ECDHE-RSA-CHACHA20-POLY1305: ECDHE-ECDSA-AES128-GCM-SHA256: ECDHE-RSA-AES128-GCM-SHA256: ECDHE-ECDSA-AES256-GCM-SHA384: ECDHE-RSA-AES256-GCM-SHA384: DHE-RSA-AES128-GCM-SHA256: DHE-RSA-AES256-GCM-SHA384: ECDHE-ECDSA-AES128-SHA256: ECDHE-RSA-AES128-SHA256: ECDHE-ECDSA-AES128-SHA: ECDHE-RSA-AES256-SHA384: ECDHE-RSA-AES128-SHA: ECDHE-ECDSA-AES256-SHA384: ECDHE-ECDSA-AES256-SHA: ECDHE-RSA-AES256-SHA: DHE-RSA-AES128-SHA256: DHE-RSA-AES128-SHA: DHE-RSA-AES256-SHA256: DHE-RSA-AES256-SHA: ECDHE-ECDSA-DES-CBC3-SHA: ECDHE-RSA-DES-CBC3-SHA: EDH-RSA-DES-CBC3-SHA: AES128-GCM-SHA256: AES256-GCM-SHA384: AES128-SHA256: AES256-SHA256: AES128-SHA: AES256-SHA: DES-CBC3-SHA: !DSS`
- ルール: `no-ssl3 no-tls-tickets`

`hieradata` オーバーライドファイル (`haproxy-hiera-overrides.yaml`) を作成して、以下の内容を記載します。

```
tripleo::haproxy::ssl_cipher_suite: ECDHE-ECDSA-CHACHA20-POLY1305: ECDHE-RSA-CHACHA20-POLY1305: ECDHE-ECDSA-AES128-GCM-SHA256: ECDHE-RSA-AES128-GCM-SHA256: ECDHE-ECDSA-AES256-GCM-SHA384: ECDHE-RSA-AES256-GCM-SHA384: DHE-RSA-AES128-GCM-SHA256: DHE-RSA-AES256-GCM-SHA384: ECDHE-ECDSA-AES128-SHA256: ECDHE-RSA-AES128-SHA256: ECDHE-ECDSA-AES128-SHA: ECDHE-RSA-AES256-SHA384: ECDHE-RSA-AES128-SHA: ECDHE-ECDSA-AES256-SHA384: ECDHE-ECDSA-AES256-SHA: ECDHE-RSA-AES256-SHA: DHE-RSA-AES128-SHA256: DHE-RSA-AES128-SHA: DHE-RSA-AES256-SHA256: DHE-RSA-AES256-SHA: ECDHE-ECDSA-DES-CBC3-SHA: ECDHE-RSA-DES-CBC3-SHA: EDH-RSA-DES-CBC3-SHA: AES128-GCM-SHA256: AES256-GCM-SHA384: AES128-SHA256: AES256-SHA256: AES128-SHA: AES256-SHA: DES-CBC3-SHA: !DSS
tripleo::haproxy::ssl_options: no-ssl3 no-tls-tickets
```



#### 注記

暗号のコレクションは、改行なしで1行に記述します。

**undercloud.conf** ファイルで先程作成した **hierradata** オーバーライドファイルを使用するように **hieradata\_override** パラメーターを設定してから **openstack undercloud install** を実行します。

```
[DEFAULT]
...
hieradata_override = haproxy-hiera-overrides.yaml
...
```

## 付録G POWER 版 RED HAT OPENSTACK PLATFORM (テクノロジープレビュー)



### 重要

この機能は、本リリースではテクノロジープレビューとして提供しているため、Red Hat では全面的にはサポートしていません。これは、テスト目的のみでご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー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。

Red Hat OpenStack Platform を新規インストールする場合は、オーバークラウドのコンピュータノードを POWER (ppc64le) ハードウェアにデプロイする必要があります。コンピュータノードのクラスターには、同じアーキテクチャーのシステムを使用するか、x86\_64 と ppc64le のシステムを混在させることができます。アンダークラウド、コントローラーノード、Ceph Storage ノード、およびその他のシステムはすべて x86\_64 ハードウェアでのみサポートされています。

### 概要:

- アンダークラウドを x86\_64 ノードにデプロイする。
- x86\_64 ノードをオーバークラウドコントローラーノードとして使用する準備を行い、それらのノードをプロビジョニングできる状態にする。
- 事前にプロビジョニングされた ppc64le ノードをオーバークラウドコンピュータノードとして使用する準備を行う。
- カスタム roles\_data.yaml ファイルを生成し、ppc64le ノード用の ComputeAllt ロールを追加する。
- オーバークラウドをデプロイする。
- アンダークラウドのメタデータサーバーをポーリングする。
- オーバークラウドのデプロイメントが正常に完了したことを確認する。

### IBM POWER ベースのコンピュータノードを使用する Red Hat OpenStack Platform のデプロイ:

1. アンダークラウドを x86\_64 ノードにデプロイします。「[1章はじめに](#)」から「[5章コンテナイメージのソースの設定](#)」に記載の手順に従います。
2. x86\_64 ノードをオーバークラウドコントローラーノードとして使用する準備を行い、それらのノードをプロビジョニングできる状態にします。コントローラーノード用に、少なくとも1つのノードが必要です。高可用性用の追加のコントローラーノードや追加の x86\_64 コンピュータノードを必要に応じて準備します。「[6章CLI ツールを使用した基本的なオーバークラウドの設定](#)」から「[環境ファイルを使用したオーバークラウドのカスタマイズ](#)」に記載の手順に従います。
3. 事前にプロビジョニングされた ppc64le ノードをオーバークラウドコンピュータノードとして使用する準備を行います。コンピュータノード用に、少なくとも1つのノードが必要です。必要に応じて、高可用性のために別のコンピュータノードを準備します。「[8章事前にプロビジョニングされたノードを使用した基本的なオーバークラウドの設定](#)」から「[コントロールプレーンのネットワークの設定](#)」に記載の手順に従います。
4. director ノードでカスタム roles\_data.yaml ファイルを生成し、ppc64le ノード用の ComputeAllt ロールを追加します。以下に例を示します。

```
(undercloud) [stack@director ~]$ openstack overcloud roles generate \
--roles-path /usr/share/openstack-tripleo-heat-templates/roles/ \
-o /home/stack/roles_data.yaml \
Controller Compute ComputeAlt BlockStorage ObjectStorage CephStorage
```

5. オーバークラウドをデプロイします。お使いの環境で必要となる標準の環境ファイルに加えて、カスタム **roles\_data.yaml** ファイルおよび **computealt.yaml** 環境ファイルを指定します。以下に例を示します。

```
(undercloud) [stack@director ~]$ openstack overcloud deploy \
--templates /usr/share/openstack-tripleo-heat-templates \
-r /home/stack/roles_data.yaml \
--disable-validations \
--ntp-server pool.ntp.org \
-e /home/stack/templates/ctlplane-assignments.yaml \
-e /home/stack/templates/node-info.yaml \
-e /home/stack/templates/overcloud_images.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/computealt.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/deployed-server-bootstrap-environment-rhel.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/storage-environment.yaml
```

このコマンドでは、以下のオプションが使われています。

#### **--templates**

**/usr/share/openstack-tripleo-heat-templates** の Heat テンプレートコレクションをベースとして使用し、オーバークラウドを作成します。

#### **-r /home/stack/roles\_data.yaml**

デプロイメントに必要なロールのマッピング情報には、カスタム **roles\_data.yaml** ファイルを使用します。

#### **--disable-validations**

事前にプロビジョニングされたインフラストラクチャーで使用しないサービスに対する基本的な CLI 検証を無効化します。無効化しないと、デプロイメントに失敗します。

#### **--ntp-server pool.ntp.org**

時刻の同期に NTP サーバーを使用します。オーバークラウドノードクラスターの同期を保つためには、このオプションが必要です。

#### **-e /home/stack/templates/ctlplane-assignments.yaml**

コントロールプレーンのネットワークを設定する環境ファイルを追加します。詳細は、「[コントロールプレーンのネットワークの設定](#)」を参照してください。

#### **-e /home/stack/templates/node-info.yaml**

各ロールに使用するノード数とフレーバーを定義する環境ファイルを追加します。

#### **-e /home/stack/templates/overcloud\_images.yaml**

コンテナイメージのソースが記載された環境ファイルを追加します。詳しくは、「[5章 コンテナイメージのソースの設定](#)」を参照してください。

**-e /usr/share/openstack-tripleo-heat-templates/environments/computealt.yaml**

ppc64le ノードを定義する環境ファイルを追加します。

**-e /usr/share/openstack-tripleo-heat-templates/environments/deployed-server-bootstrap-environment-rhel.yaml**

事前にプロビジョニングされたサーバー上でブートストラップのスクリプトを実行する環境ファイルを追加します。このスクリプトは、追加パッケージをインストールして、オーバークラウドノードの基本設定を提供します。

**-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml**

オーバークラウドデプロイメントのネットワーク分離を初期化する環境ファイルを追加します。

**-e /home/stack/templates/network-environment.yaml**

ネットワーク分離をカスタマイズする環境ファイルを追加します。

**-e /home/stack/templates/storage-environment.yaml**

ストレージの設定を初期化する環境ファイルを追加します。



### 注記

デプロイメントのスタックは、オーバークラウドノードのリソースが **CREATE\_IN\_PROGRESS** の段階に入ると一時停止します。この一時停止は、**director** がオーバークラウドノード上のオーケストレーションエージェントがメタデータサーバーをポーリングするのを待っているためです。次のステップに進み、メタデータサーバーのポーリングを開始します。

6. アンダークラウドのメタデータサーバーをポーリングします。「[メタデータサーバーのポーリング](#)」を参照してください。
7. オーバークラウドのデプロイメントが正常に完了したことを確認します。「[オーバークラウド作成の監視](#)」と「[オーバークラウドへのアクセス](#)」を参照してください。事前にプロビジョニングされた **ppc64le** ノードと、**director** によってプロビジョニングされた **x86\_64** ノードを含む全コンピュータノードを一覧表示するには、**openstack hypervisor list** を実行します。