



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 © 2023 Red Hat, Inc.

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

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

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

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

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

Linux[®] 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 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. コンテナ化	10
1.5. CEPH STORAGE	10
第2章 要件	12
2.1. 環境要件	12
2.2. アンダークラウドの要件	13
2.3. ネットワーク要件	16
2.4. オーバークラウドの要件	18
2.5. リポジトリの要件	22
第3章 オーバークラウドのプランニング	25
3.1. ノードのデプロイメントロールのプランニング	25
3.2. ネットワークのプランニング	26
3.3. ストレージのプランニング	31
3.4. 高可用性のプランニング	32
第4章 アンダークラウドのインストール	34
4.1. プロキシを使用してアンダークラウドを実行する際の考慮事項	34
4.2. STACK ユーザーの作成	35
4.3. テンプレートとイメージ用のディレクトリーの作成	36
4.4. アンダークラウドのホスト名の設定	36
4.5. アンダークラウドの登録と更新	37
4.6. DIRECTOR パッケージのインストール	38
4.7. CEPH-ANSIBLE のインストール	38
4.8. DIRECTOR の設定	39
4.9. DIRECTOR の設定パラメーター	39
4.10. アンダークラウドへの HIERADATA の設定	44
4.11. DIRECTOR のインストール	45
4.12. オーバークラウドノードのイメージの取得	46
4.13. コントロールプレーン用のネームサーバーの設定	50
4.14. 次のステップ	51
第5章 コンテナイメージのソースの設定	52
5.1. レジストリーメソッド	52
5.2. コンテナイメージの準備コマンドの使用法	53
5.3. 追加のサービス用のコンテナイメージ	54
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	57
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	58
5.6. SATELLITE サーバーをレジストリーとして使用する手順	60
5.7. 次のステップ	63
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	64
6.1. オーバークラウドノードの登録	65
6.2. ノードのハードウェアの検査	66
6.3. ベアメタルノードの自動検出	73
6.4. アーキテクチャーに固有なロールの生成	76
6.5. プロファイルへのノードのタグ付け	76
6.6. ルートディスクの定義	77

6.7. OVERCLOUD-MINIMAL イメージの使用による RED HAT サブスクリプションエンタイトルメントの使用回避	79
6.8. ノード数とフレーバーを定義する環境ファイルの作成	80
6.9. アンダークラウド CA を信頼するオーバークラウドノードの設定	80
6.10. 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	82
6.11. CLI ツールを使用したオーバークラウドの作成	83
6.12. オーバークラウド作成時の環境ファイルの追加	88
6.13. オーバークラウドプランの管理	92
6.14. オーバークラウドのテンプレートおよびプランの検証	92
6.15. オーバークラウド作成の監視	93
6.16. オーバークラウドデプロイメント出力の表示	93
6.17. オーバークラウドへのアクセス	94
6.18. オーバークラウド作成の完了	94
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	95
7.1. WEB UI へのアクセス	95
7.2. WEB UI のナビゲーション	96
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	99
7.4. WEB UI を使用したノードの登録	100
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	102
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	103
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	104
7.8. WEB UI でのロールの追加	105
7.9. WEB UI を使用したロールへのノードの割り当て	106
7.10. WEB UI を使用したロールパラメーターの編集	106
7.11. WEB UI を使用したオーバークラウドの作成開始	108
7.12. オーバークラウド作成の完了	110
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	111
8.1. ノード設定のためのユーザーの作成	112
8.2. ノードのオペレーティングシステムの登録	113
8.3. ノードへのユーザーエージェントのインストール	114
8.4. DIRECTOR への SSL/TLS アクセスの設定	114
8.5. コントロールプレーンのネットワークの設定	114
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	116
8.7. 事前にプロビジョニングされたノード向けの CEPH STORAGE の設定	118
8.8. 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	119
8.9. メタデータサーバーのポーリング	120
8.10. オーバークラウド作成の監視	122
8.11. オーバークラウドへのアクセス	122
8.12. 事前にプロビジョニングされたノードのスケーリング	123
8.13. 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	124
8.14. オーバークラウド作成の完了	124
第9章 オーバークラウド作成後のタスクの実行	125
9.1. コンテナ化されたサービスの管理	125
9.2. オーバークラウドのテナントネットワークの作成	126
9.3. オーバークラウドの外部ネットワークの作成	127
9.4. 追加の FLOATING IP ネットワークの作成	128
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	128
9.6. 基本的なオーバークラウドフレーバーの作成	129
9.7. オーバークラウドの検証	129
9.8. オーバークラウド環境の変更	130
9.9. 動的インベントリースクリプトの実行	131

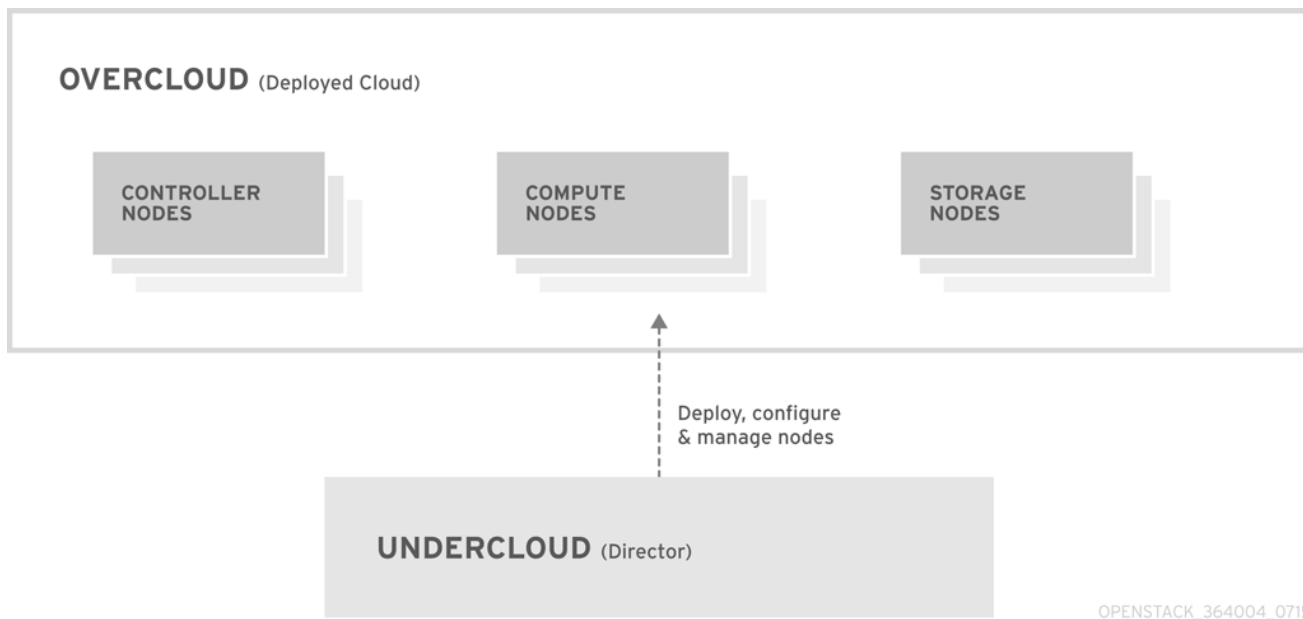
9.10. オーバークラウドへの仮想マシンのインポート	132
9.11. オーバークラウドの削除防止	133
9.12. オーバークラウドの削除	133
第10章 ANSIBLE を使用したオーバークラウドの設定	134
10.1. ANSIBLE ベースのオーバークラウド設定 (CONFIG-DOWNLOAD)	134
10.2. オーバークラウドの設定メソッドを CONFIG-DOWNLOAD に切り替える手順	134
10.3. 事前にプロビジョニング済みのノードでの CONFIG-DOWNLOAD の有効化	136
10.4. CONFIG-DOWNLOAD の作業ディレクトリへのアクセスの有効化	137
10.5. CONFIG-DOWNLOAD のログと作業ディレクトリの確認	137
10.6. 手動での CONFIG-DOWNLOAD の実行	138
10.7. CONFIG-DOWNLOAD の無効化	139
10.8. 次のステップ	140
第11章 仮想コントロールプレーンの作成	141
11.1. 仮想コントロールプレーンのアーキテクチャー	141
11.2. RHOSP オーバークラウドのコントロールプレーンを仮想化する際の利点と制限	141
11.3. RED HAT VIRTUALIZATION ドライバーを使用した仮想コントローラーのプロビジョニング	142
第12章 オーバークラウドノードのスケーリング	146
12.1. オーバークラウドへのノード追加	146
12.2. ロールのノード数の追加	148
12.3. コンピュートノードの削除または交換	148
12.4. CEPH STORAGE ノードの置き換え	155
12.5. オブジェクトストレージノードの置き換え	155
12.6. ノードのブラックリスト登録	156
第13章 コントローラーノードの置き換え	159
13.1. コントローラー置き換えの準備	159
13.2. バックアップまたはスナップショットからのコントローラーノードの復元	160
13.3. CEPH MONITOR デーモンの削除	161
13.4. コントローラーを置き換えるためのクラスター準備	162
13.5. コントローラーノードの再利用	163
13.6. BMC IP アドレスの再利用	164
13.7. コントローラーノード置き換えのトリガー	165
13.8. コントローラーノード置き換え後のクリーンアップ	166
第14章 ノードのリブート	168
14.1. アンダークラウドノードのリブート	168
14.2. コントローラーノードおよびコンポーザブルノードのリブート	168
14.3. スタンドアロンの CEPH MON ノードのリブート	169
14.4. CEPH STORAGE (OSD) クラスターのリブート	169
14.5. OBJECT STORAGE サービス (SWIFT) ノードの再起動	170
14.6. コンピュートノードのリブート	170
第15章 DIRECTOR の問題のトラブルシューティング	172
15.1. ノード登録のトラブルシューティング	172
15.2. 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	172
15.3. ワークフローおよび実行に関するトラブルシューティング	174
15.4. オーバークラウドの作成に関するトラブルシューティング	175
15.5. 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	179
15.6. "NO VALID HOST FOUND" エラーのトラブルシューティング	179
15.7. オーバークラウド作成後のトラブルシューティング	180
15.8. アンダークラウドの調整	184
15.9. SOS レポートの作成	185

15.10. アンダークラウドとオーバークラウドの重要なログ	185
付録A SSL/TLS 証明書の設定	187
A.1. 署名ホストの初期化	187
A.2. 認証局の作成	187
A.3. クライアントへの認証局の追加	187
A.4. SSL/TLS 鍵の作成	188
A.5. SSL/TLS 証明書署名要求の作成	188
A.6. SSL/TLS 証明書の作成	189
A.7. アンダークラウドで証明書を使用する場合	189
付録B 電源管理ドライバー	191
B.1. REDFISH	191
B.2. DELL REMOTE ACCESS CONTROLLER (DRAC)	191
B.3. INTEGRATED LIGHTS-OUT (ILO)	191
B.4. CISCO UNIFIED COMPUTING SYSTEM (UCS)	192
B.5. FUJITSU INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)	193
B.6. VIRTUAL BASEBOARD MANAGEMENT CONTROLLER (VBMC)	193
B.7. RED HAT VIRTUALIZATION	196
B.8. フェイクドライバー	196
付録C 完全なディスクイメージ	198
C.1. ベースのクラウドイメージのダウンロード	199
C.2. ディスクイメージの環境変数	199
C.3. ディスクレイアウトのカスタマイズ	200
C.4. セキュリティーが強化された完全なディスクイメージの作成	204
C.5. セキュリティーが強化された完全なディスクイメージのアップロード	204
付録D 代替ブートモード	205
D.1. 標準の PXE	205
D.2. UEFI ブートモード	205
付録E プロファイルの自動タグ付け	207
E.1. ポリシーファイルの構文	207
E.2. ポリシーファイルの例	209
E.3. ポリシーファイルのインポート	210
E.4. プロファイルの自動タグ付けのプロパティー	211
付録F セキュリティーの強化	212
F.1. HAPROXY の SSL/TLS の暗号およびルールの変更	212
付録G RED HAT OPENSTACK PLATFORM FOR POWER	214
G.1. CEPH STORAGE	214
G.2. コンポーザブルサービス	214

第1章 はじめに

Red Hat OpenStack Platform (RHOSP) director は、完全な RHOSP 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のプロビジョニングや制御を行う新たな OpenStack のコンポーネントが含まれます。

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。まずアンダークラウドをインストールし、続いてアンダークラウドをツールとして使用してオーバークラウドをインストールおよび設定します。



OPENSTACK_364004_0715

1.1. アンダークラウド

アンダークラウドは、Red Hat OpenStack Platform director ツールセットが含まれる主要管理ノードです。OpenStack をインストールした単一システムで、OpenStack 環境 (オーバークラウド) を設定する OpenStack ノードをプロビジョニング/管理する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設定するコンポーネントは、さまざまな機能を持ちます。

環境のプランニング

アンダークラウドには、特定のノードロールを作成して割り当てるのに使用できるプランニング機能が含まれます。アンダークラウドには、Compute、Controller、さまざまな Storage ロールなど、特定のノードに割り当てることができるデフォルトのノードロールセットが含まれます。また、カスタムロールを設定することもできます。さらに、各ノードロールにどの Red Hat 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、各ノードの帯域外管理インターフェイス (通常 Intelligent Platform Management Interface (IPMI)) 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。電源管理ドライバーの全一覧については、[付録B 電源管理ドライバー](#)を参照してください。

オーケストレーション

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む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イスまたは Web ベースのユーザーインターフェイスで、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): モニターリング向けのアラームコンポーネント
 - OpenStack Telemetry Event Storage (panko): モニター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ノードロールが含まれます。アンダークラウドには、以下に示すオーバークラウドのデフォルトノードロールセットが含まれます。

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設定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ロールは、以下のコンポーネントをサポートします。これらのサービスがすべてデフォルトで有効化されている訳ではありません。これらのコンポーネントの中には、有効にするのにカスタムの環境ファイルまたは事前にパッケージ化された環境ファイルを必要とするものがあります。

- 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

コンピュータ

これらのノードは OpenStack 環境にコンピュータリソースを提供します。コンピュータ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タノードには、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。

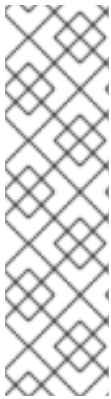
- Ceph Storage ノード: ストレージクラスターを設定するために使用します。それぞれのノードには、Ceph Object Storage Daemon (OSD) が含まれます。また、Ceph Storage ノードをデプロイする場合には、director により Ceph Monitor がコントローラーノードにインストールされます。
- Block storage (Cinder): HA コントローラーノードの外部ブロックストレージとして使用します。このノードには、以下のコンポーネントが含まれます。
 - OpenStack Block Storage (cinder) ボリューム
 - 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設定では、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、クラスターのリソースを管理します。Pacemaker は、クラスター内の全ノードにわたって OpenStack コンポーネントの可用性を管理および監視します。
- 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。
- バージョン 13 から、director を使用してコンピュートインスタンスの高可用性 (インスタンス HA) をデプロイできるようになりました。インスタンス HA により、コンピュートノードで障害が発生した際にそのノードからインスタンスを自動的に退避することができます。

1.4. コンテナ化

オーバークラウド上の各 OpenStack Platform サービスは、対応するノード上の個別の Linux コンテナ内で実行されます。これにより、それぞれのサービスを分離し、OpenStack Platform を簡単に維持およびアップグレードすることができます。以下に示すように、Red Hat では、オーバークラウド用コンテナイメージを取得するためのさまざまな方法をサポートしています。

- Red Hat Container Catalog から直接イメージをプルする
- アンダークラウド上でイメージをホストする
- Satellite 6 サーバー上でイメージをホストする

本ガイドでは、レジストリー情報の設定および基本的なコンテナ操作の実施方法について説明します。コンテナ化されたサービスに関する詳しい情報は、[コンテナ化されたサービスへの移行](#) を参照してください。

1.5. CEPH STORAGE

通常、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。ブロックストレージリソースの消費に関して、それぞれの OpenStack クライアントは固有のニーズを持つのが一般的です。glance (イメージ)、cinder (ボリューム)、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。



注記

マルチアーキテクチャクラウドでは、事前にインストール済みの Ceph または外部の Ceph **しか**サポートされません。詳細は、[オーバークラウドの既存 Red Hat Ceph クラスターとの統合](#) および [付録G Red Hat OpenStack Platform for POWER](#) 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](#) のアート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
- クラスター内に Red Hat OpenStack Platform コントローラーノード用のホストマシン 3 台
- クラスター内に 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。少なくとも、コンピュートノードおよび Ceph Storage ノードにはベアメタルシステムが必要です。
- すべてのオーバークラウドベアメタルシステムには、Intelligent Platform Management Interface (IPMI) が必要です。これは、director が電源管理を制御するためです。
- 各ノードの内部 BIOS クロックを UTC に設定します。これにより、タイムゾーンオフセットを適用する前に **hwclock** が BIOS クロックを同期するとファイルのタイムスタンプに未来の日時を設定される問題を防ぐことができます。
- Red Hat OpenStack Platform には、ロケール設定の一部として特殊文字のエンコーディングに関する要件があります。
 - すべてのノードで UTF-8 エンコーディングを使用します。すべてのノードで **LANG** 環境変数を **en_US.UTF-8** に設定するようにします。
 - Red Hat OpenStack Platform リソース作成の自動化に Red Hat Ansible Tower を使用している場合は、非 ASCII 文字を使用しないでください。
- 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 Red Hat OpenStack Platform for POWER](#) 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
 - **ceph-ansible** Playbook は、アンダークラウドによりデプロイされたホスト 10 台につき 1 GB の常駐セットサイズ (RSS) を消費します。デプロイされたオーバークラウドが既存の Ceph クラスタを使用する、または新たな Ceph クラスタをデプロイする場合には、それに応じてアンダークラウド用 RAM をプロビジョニングしてください。
- ルートディスク上に最小 100 GB の空きディスク領域。その内訳を以下に示します。
 - 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イスカード。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 インターフェイスを使用することを推奨します。
- ホストオペレーティングシステムとして、最新バージョンの Red Hat Enterprise Linux 7 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	説明
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 7 でホストされていること
Red Hat 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 Virtualization 4.x でホストされていること
Microsoft Hyper-V	Red Hat Customer Portal Certification Catalogue に記載の Hyper-V のバージョンでホストされている。
VMware ESX および ESXi	Red Hat Customer Portal Certification Catalogue 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 には、ホストオペレーティングシステムとして、最新バージョンの Red Hat Enterprise Linux 7 がインストールされている必要があります。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することを推奨します。



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。これらの設定を適用したら、director 仮想マシンの電源を一旦オフにしてから再投入する必要があります。仮想マシンをリブートするだけでは不十分です。

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イ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 経由でブートします。

これらのブリッジを作成および設定する方法の詳細は、Red Hat Enterprise Linux 7 ネットワークガイドの [ネットワークブリッジの設定](#) を参照してください。

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 virt-viewer
```

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** ドメインが起動します。**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、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:** [director のアンダークラウドのバックアップとリストア](#) 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クアップ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イスでネイティブ 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: オーバークラウドおよびアンダーグラウンドへの外部アクセスに使用する別のネットワーク。このネットワークに接続するこのインターフェイスには、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イ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設定には、以下が含まれます。
 - シングル NIC 設定: ネイティブ VLAN 上のプロビジョニングネットワークと、オーバークラウドネットワークの種別ごとのサブネットを使用するタグ付けされた VLAN 用に NIC を 1 つ。
 - デュアル NIC 設定: 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 設定: 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 設定: 各 NIC は、オーバークラウドネットワークの種別ごと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イスの作成、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1IP 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。詳細は、「[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、[ネットワークガイド](#)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、ブート順序の最上位に指定するようにします。
- すべてのオーバークラウドベアメタルシステムには、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イスが必要です。このインターフェイスにより、director は各ノードの電源管理機能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、単一のテナントでのみ使用するように確保され、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。この設定では、お使いの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ます。したがって、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しなければならない場合があります。
- 1つのブリッジには単一のインターフェイ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イ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。
- コントローラーノードのネットワークカードまたはネットワークスイッチの異常がオーバークラウドサービスの可用性を阻害するのを防ぐには、keystone 管理エンドポイントがボンディングされたネットワークカードまたはネットワークハードウェアの冗長性を使用するネットワークに配置されるようにしてください。keystone エンドポイントを **internal_api** などの別のネットワークに移動する場合は、アンダークラウドが VLAN またはサブネットに到達できるようにします。詳細は、Red Hat ナレッジベースのソリューション [How to migrate Keystone Admin Endpoint to internal_api network ?](#) を参照してください。

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. コンピュートノードの要件

コンピュートノードは、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ロールを果たします。コンピュートノードは、ハードウェアの仮想化をサポートしている必要があります。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。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、この最低要求に追加の RAM を加算します。

ディスク領域

最小 50 GB の空きディスク領域

ネットワークインターフェイスカード

最小 1 枚の 1 Gbps ネットワークインターフェイスカード (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イス向けの場合には、追加のネットワークインターフェイスを使用します。

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イ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) をホストするロールを果たします。

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。

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 Hardware Requirements for Highly Available Controllers](#) のアートを参照してください。

ディスクストレージとレイアウト

Object Storage サービス (swift) がコントローラーノード上で実行されていない場合には、最小で 50 GB のストレージが必要です。ただし、Telemetry (**gnocchi**) および Object Storage 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イスカード

最小 2 枚の 1Gbps ネットワークインターフェイスカード。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イス向けの場合には、追加のネットワークインターフェイスを使用します。

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イ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。

2.4.2.1. 仮想化サポート

Red Hat では、Red Hat Virtualization プラットフォーム上の仮想コントローラーノードのみをサポートします。詳細は、[仮想コントロールプレーンの作成](#) を参照してください。

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ロールを果たします。

配置グループ

デプロイメントの規模によらず、動的で効率的なオブジェクトの追跡を容易に実施するために、Ceph では配置グループが使用されています。OSD の障害やクラスターのリバランスの際には、Ceph は配置グループおよびその内容を移動または複製することができるので、Ceph クラスターは効率的にリバランスおよび復旧を行うことができます。director が作成するデフォルトの配置グループ数が常に最適とは限らないので、実際の要件に応じて正しい配置グループ数を計算することが重要です。配置グループの計算ツールを使用して、正しい数を計算することができます ([Ceph Placement Groups \(PGs\) per Pool Calculator](#) を参照)。

プロセッサー

Intel 64 または AMD64 CPU 拡張機能をサポートする 64 ビット x86 プロセッサー。

メモリー

Red Hat では、OSD ホスト 1 台につき 16 GB の RAM をベースラインとし、これに OSD デーモンごとに 2 GB の RAM を追加する設定を推奨します。

ディスクのレイアウト

ディスクサイズは、ストレージのニーズに依存します。推奨される Red Hat Ceph Storage ノードの設定には、少なくとも以下のようなレイアウトの 3 つのディスクが必要です。

- **/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。このディスクには、少なくとも 50 GB の空きディスク領域が必要です。
- **/dev/sdb**: ジャーナルディスク。このディスクは、Ceph OSD ジャーナル向けにパーティションに分割されます。たとえば、**/dev/sdb1**、**/dev/sdb2**、**/dev/sdb3** (以下同様) のように分割されます。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ソリッドステートドライブ (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 は **ceph-ansible** を使用しますが、Ceph Storage ノードのルートディスクへの OSD インストールには対応しません。したがって、サポートされる Ceph Storage ノードには少なくとも 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イスカード

最小 1 枚の 1 Gbps ネットワークインターフェイスカード (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イス向けの場合には、追加のネットワークインターフェイスを使用します。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 インターフェイ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イ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。

イメージ属性

Red Hat Ceph Storage ブロックデバイスのパフォーマンスを向上させるために、**virtio-scsi** ドライバーを使用するように Glance イメージを設定することができます。イメージの推奨イメージ属性に関する詳細は、Red Hat Ceph Storage ドキュメントの [Configuring Glance](#) を参照してください。

Ceph Storage クラスタを使用するオーバークラウドのインストールについての詳しい情報は、[コンテナ化された Red Hat Ceph を持つオーバークラウドのデプロイ](#) を参照してください。

2.4.4. Object Storage ノードの要件

オブジェクトストレージノードは、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。オブジェクトストレージプロキシは、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 拡張機能をサポートする 64 ビット 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。

ディスク容量

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/sdd** 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イスカード

最小 2 枚の 1Gbps ネットワークインターフェイスカード。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イス向けの場合には、追加のネットワークインターフェイスを使用します。

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イ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) または Red Hat Satellite Server 5 もしくは Red Hat Satellite Server 6 を使用した Red Hat リポジトリへのアクセスが必要です。Red Hat Satellite Server を使用する場合は、必要なリポジトリをお使いの OpenStack Platform 環境に同期する必要があります。以下の CDN チャンネル名一覧を参考にしてください。

表2.1 OpenStack Platform リポジトリ

名前	リポジトリ	要件の説明
Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rpms	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
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	rhel-7-server-extras-rpms	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	rhel-7-server-rh-common-rpms	Red Hat OpenStack Platform のデプロイと設定用のツールが含まれます。
Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs) x86_64	rhel-7-server-satellite-tools-6.3-rpms	Red Hat Satellite Server 6 でのホスト管理ツール。これより新しいバージョンの Satellite Tools リポジトリを使用すると、アンダークラウドのインストールが失敗する可能性があることに注意してください。
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	rhel-ha-for-rhel-7-server-rpms	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。

名前	リポジトリ	要件の説明
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	rhel-7-server-openstack-13-rpms	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)	rhel-7-server-rhceph-3-osd-rpms	(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)	rhel-7-server-rhceph-3-mon-rpms	(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)	rhel-7-server-rhceph-3-tools-rpms	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。Ceph Storage クラスターと共にオーバークラウドをデプロイする場合や、オーバークラウドを既存の Ceph Storage クラスターと統合する場合に、すべてのノードでこのリポジトリを有効にします。
Red Hat OpenStack 13 Director Deployment Tools for RHEL 7 (RPMs)	rhel-7-server-openstack-13-deployment-tools-rpms	(Ceph Storage ノード向け) 現在のバージョンの Red Hat OpenStack Platform director に対応したデプロイメントツールのセットを提供します。アクティブな Red Hat OpenStack Platform サブスクリプションがない Ceph ノードにインストールされます。
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	rhel-7-server-nfv-rpms	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タノードで有効化する必要があります。注記: このリポジトリにアクセスするには、別途 Red Hat OpenStack Platform for Real Time SKU のサブスクリプションが必要です。

名前	リポジトリ	要件の説明
Red Hat OpenStack Platform 13 Extended Life Cycle Support for RHEL 7 (RPMs)	rhel-7-server-openstack-13-els-rpms	2021 年 6 月 26 日に開始した延長ライフサイクルサポートの更新が含まれています。このリポジトリを利用するには、 "Entitlement for OpenStack 13 Platform Extended Life Cycle Support" (MCT3637) が必要です。

IBM POWER 用の OpenStack Platform リポジトリ

これらのリポジトリは、[付録G Red Hat OpenStack Platform for POWER](#)で説明する機能に使われま

す。

名前	リポジトリ	要件の説明
Red Hat Enterprise Linux for IBM Power, little endian	rhel-7-for-power-le-rpms	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
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	rhel-7-server-openstack-13-for-power-le-rpms	ppc64le システム用 Red Hat OpenStack Platform のコアリポジトリ

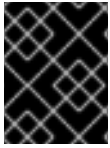


注記

ネットワークがオフラインの Red Hat OpenStack Platform 環境用リポジトリを設定するには、Red Hat カスタマーポータルで [オフライン環境で Red Hat OpenStack Platform Director を設定する](#) のアート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

以下の項で、Red Hat OpenStack Platform 環境のさまざまな要素をプランニングする際のガイドラインを説明します。これには、ノードロールの定義、ネットワークポロジのプランニング、ストレージなどが含まれます。



重要

デプロイ後は、オーバークラウドノードの名前を変更しないでください。デプロイメント後にノードの名前を変更すると、インスタンスの管理に問題が生じます。

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。高可用性に対応した実稼働レベルの環境の場合は、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 台のノードで設定される環境は実稼働用ではなく、テスト目的でしか使用することができません。2 台のノードまたは 4 台以上のノードで設定される環境はサポートされません。

コンピュータ

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。

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設定例と各シナリオで使用するノード種別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	コンピュータ	Ceph Storage	Swift Storage	合計
小規模のオーバークラウド	3	1	-	-	4

中規模のオーバークラウド	3	3	-	-	6
追加のオブジェクトストレージがある中規模のオーバークラウド	3	3	-	3	9
Ceph Storage クラスターがある中規模のオーバークラウド	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**オーバークラウドの高度なカスタマイズの [コンポーザブルサービスとカスタムロール](#)** 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、自律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。このネットワークトラフィック種別には以下が含まれます。

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	director は、このネットワークト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 OpenStack Platform 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 OpenStack 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、コンピュート、Cinder Storage、Swift Storage
テナント	Neutron 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。ネットワークトラフィックは、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、コンピュート
ストレージ	Block Storage、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	OpenStack Object Storage (swift) 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す。プロキシサービスは、ユーザー要求と下層のストレージレイヤーの間の仲介インターフェイ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。Ceph バックエンドを使用するサービスは、Ceph 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。RBD ドライバーは例外で、このトラフィックは直接 Ceph に接続する点に注意してください。	コントローラー、Ceph Storage、Cinder Storage、Swift Storage

ストレージ NFS	このネットワークは、CephFS を NFS バックエンドにマッピングする ganesh サービスと共に Shared File System サービス (manila) を使用する場合にのみ必要です。	コントローラー
外部	グラフィカルシステム管理用の OpenStack Dashboard (Horizon)、OpenStack 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 SNAT 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 (RFC-1918 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 NAT を実行する必要があります。	コントローラーとアンダークラウド
Floating IP	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、インターフェイ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。	コントローラー
管理	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。	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つのインターフェイ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 Tenant ネットワークを作成することは変わりませんが、Tenant 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。

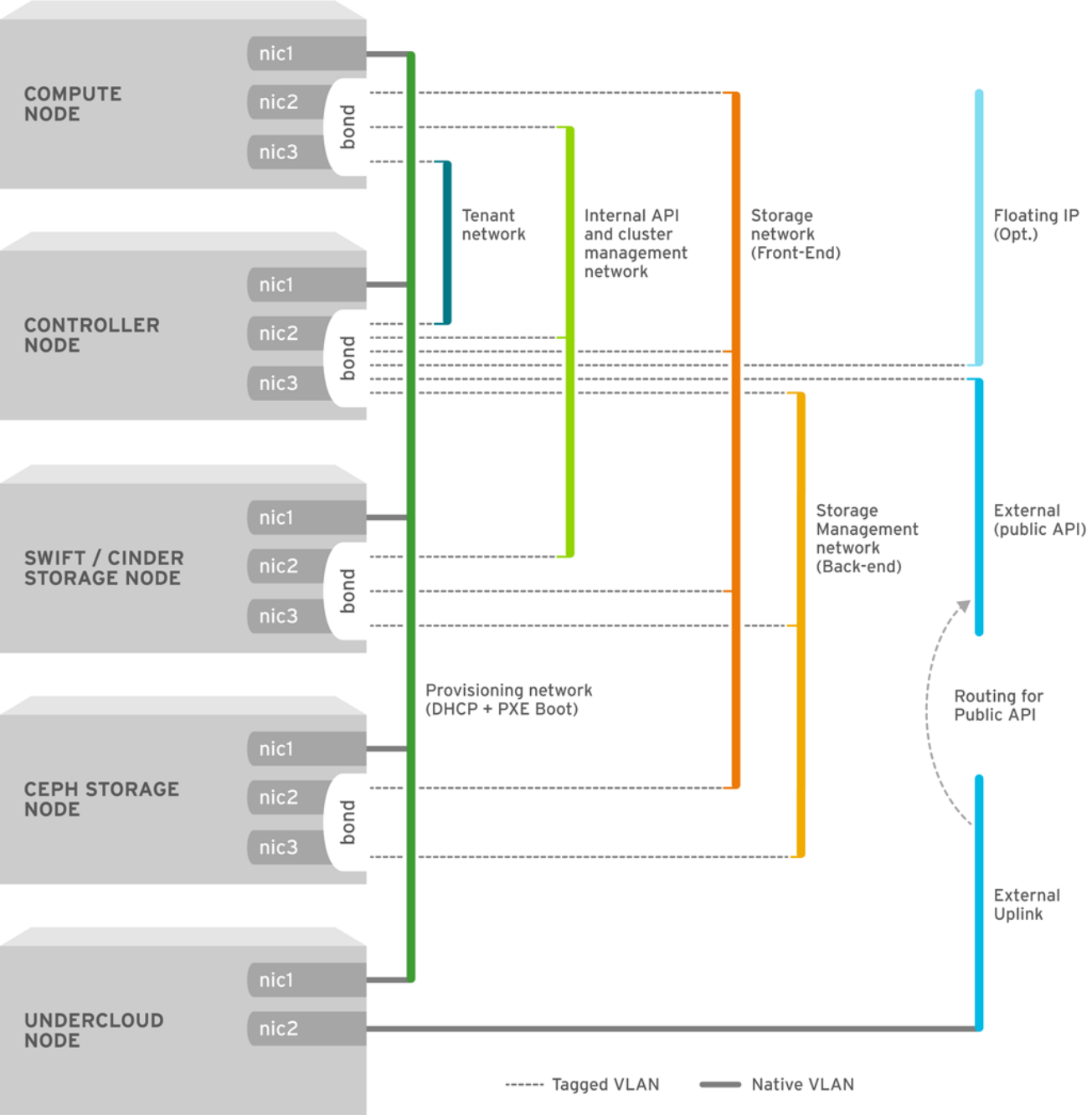
director は、トラフィック種別の中から 6 つを特定のサブネットまたは VLAN にマッピングする方法を提供します。このようなトラフィック種別には、以下が含まれます。

- 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 (オプション)

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**) のインターフェイスを使用して、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。

図3.1 ボンディングインターフェイスを使用する VLAN トポロジーの例



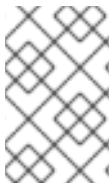
OPENSTACK_364029_0715

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	インターフェイスの総数	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: 管理 (オプション)</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イス 2 つを含む)	7



注記

Red Hat Virtualization (RHV) を使用している場合には、オーバークラウドのコントロールプレーンを仮想化することができます。詳細は、[仮想コントロールプレーンの作成](#) を参照してください。

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、ボリュームの可視性/可用性、およびデータ破損の問題が生じます。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、[ストレージガイドのバックエンド](#) および KCS アーティクル [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。イメージを変更することはできません。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。glance を使用して、Ceph ブロックデバイスにイメージを保管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 では、ボリュームを使用して仮想マシンをブートしたり、ボリュームを実行中の仮想マシンにアタッチしたりします。OpenStack は cinder サービスを使用してボリュームを管理します。イメージの CoW (Copy-on-Write) クローンを使用して、cinder により仮想マシンをブートすることができます。
- **ファイルシステム:** manila 共有はファイルシステムによりバックアップされます。OpenStack ユーザーは、manila サービスを使用して共有を管理します。manila を使用して、Ceph Storage ノードにデータを保管する CephFS ファイルシステムにバックアップされる共有を管理することができます。
- **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 で仮想マシンをブートすると、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)。Ceph 内にあるすべての仮想マシンは、Cinder を使用せずにブートすることができます。これにより、ライブマイグレーションのプロセスを使用して、簡単にメンテナンス操作を実行することができます。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別の場所でも実行することもできるので便利です。



重要

サポートされるイメージフォーマットの情報については、[インスタンス&イメージガイドの Image サービス](#) の章を参照してください。

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。

3.4. 高可用性のプランニング

高可用性なオーバークラウドをデプロイするために、director は複数のコントローラー、コンピュータ、およびストレージノードを単一のクラスターとして連携するように設定します。ノードで障害が発生すると、障害が発生したノードのタイプに応じて、自動フェンシングおよび再起動プロセスがトリガーされます。オーバークラウドの高可用性アーキテクチャーおよびサービスに関する情報は、[高可用性デプロイメントと使用方法](#) を参照してください。



重要

STONITH を使用しない高可用性オーバークラウドのデプロイはサポートの対象外です。高可用性オーバークラウドの Pacemaker クラスターの一部である各ノードには、STONITH デバイスを設定する必要があります。STONITH および Pacemaker の詳細は、[Fencing in a Red Hat High Availability Cluster](#) および [Support Policies for RHEL High Availability Clusters - General Requirements for Fencing/STONITH](#) を参照してください。

director を使用して、コンピュートインスタンスの高可用性 (インスタンス HA) を設定することもできます。このメカニズムにより、ノードで障害が発生するとコンピュートノード上のインスタンスが自動的に退避および再起動されます。インスタンス HA に対する要件は通常のオーバークラウドの要件と同じですが、追加のステップを実施してデプロイメントのために環境を準備する必要があります。インスタンス HA の仕組みおよびインストール手順に関する情報は、[コンピュートインスタンスの高可用性](#) を参照してください。

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. プロキシを使用してアンダークラウドを実行する際の考慮事項

ご自分の環境でプロキシを使用している場合は、以下の考慮事項を確認して、Red Hat OpenStack Platform の一部とプロキシを統合する際のさまざまな設定手法、およびそれぞれの手法の制限事項を十分に理解するようにしてください。

システム全体のプロキシ設定

アンダークラウド上のすべてのネットワークトラフィックに対してプロキシ通信を設定するには、この手法を使用します。プロキシ設定を定義するには、`/etc/environment` ファイルを編集して以下の環境変数を設定します。

http_proxy

標準の HTTP リクエストに使用するプロキシ

https_proxy

HTTPS リクエストに使用するプロキシ

no_proxy

プロキシ通信から除外するドメインのコンマ区切りリスト

システム全体のプロキシ手法には、以下の制限事項があります。

- **no_proxy** 変数は、主にドメイン名 (www.example.com)、ドメイン接尾辞 (example.com)、およびワイルドカード付きのドメイン (*.example.com) を使用します。ほとんどの Red Hat OpenStack Platform サービスは **no_proxy** の IP アドレスを解釈しますが、コンテナのヘルスチェックなどの特定のサービスは、`cURL` と `wget` による制限のため **no_proxy** 環境変数の IP アドレスを解釈しません。アンダークラウドでシステム全体のプロキシを使用するには、インストール中に `undercloud.conf` ファイルの **container_healthcheck_disabled** パラメーターを使用してコンテナヘルスチェックを無効にします。詳細については、[BZ#1837458: Container health checks fail to honor no_proxy CIDR notation](#) を参照してください。
- 一部のコンテナでは、`/etc/environments` の環境変数が正しくバインドおよび解析されないため、これらのサービスの実行時に問題が発生します。詳細については、[BZ#1916070: proxy configuration updates in /etc/environment files are not being picked up in containers correctly](#) および [BZ#1918408: mistral_executor container fails to properly set no_proxy environment parameter](#) を参照してください。

dnf プロキシ設定

すべてのトラフィックがプロキシを通過するように **dnf** を設定するには、この手法を使用します。プロキシ設定を定義するには、`/etc/dnf/dnf.conf` ファイルを編集して以下のパラメーターを設定します。

proxy

プロキシサーバーの URL

proxy_username

プロキシサーバーへの接続に使用するユーザー名

proxy_password

プロキシサーバーへの接続に使用するパスワード

proxy_auth_method

プロキシサーバーが使用する認証方法

これらのオプションの詳細については、**man dnf.conf** を実行してください。

dnf プロキシ手法には、以下の制限事項があります。

- この手法では、**dnf** に対してのみプロキシがサポートされます。
- **dnf** プロキシ手法には、特定のホストをプロキシ通信から除外するオプションは含まれていません。

Red Hat Subscription Manager プロキシ

すべてのトラフィックがプロキシを通過するように Red Hat Subscription Manager を設定するには、この手法を使用します。プロキシ設定を定義するには、**/etc/rhsm/rhsm.conf** ファイルを編集して以下のパラメーターを設定します。

proxy_hostname

プロキシのホスト

proxy_scheme

プロキシをリポジトリ定義に書き出す際のプロキシのスキーム

proxy_port

プロキシのポート

proxy_username

プロキシサーバーへの接続に使用するユーザー名

proxy_password

プロキシサーバーへの接続に使用するパスワード

no_proxy

プロキシ通信から除外する特定ホストのホスト名接尾辞のコンマ区切りリスト

これらのオプションの詳細については、**man rhsm.conf** を実行してください。

Red Hat Subscription Manager プロキシ手法には、以下の制限事項があります。

- この手法では、Red Hat Subscription Manager に対してのみプロキシがサポートされます。
- Red Hat Subscription Manager プロキシ設定の値は、システム全体の環境変数に設定されたすべての値をオーバーライドします。

透過プロキシ

アプリケーション層のトラフィックを管理するのにネットワークで透過プロキシが使用される場合は、プロキシ管理が自動的に行われるため、アンダークラウド自体をプロキシと対話するように設定する必要はありません。透過プロキシは、Red Hat OpenStack Platform のクライアントベースのプロキシ設定に関連する制限に対処するのに役立ちます。

4.2. 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. 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。

4.3. テンプレートとイメージ用のディレクトリーの作成

director はシステムのイメージと Heat テンプレートを使用して、オーバークラウド環境を構築します。これらのファイルを整理するには、イメージとテンプレート用にディレクトリーを作成するように推奨します。

```
[stack@director ~]$ mkdir ~/images
[stack@director ~]$ mkdir ~/templates
```

4.4. 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完全修飾ホスト名が出力されない、または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
```


- 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
```

4.5. アンダークラウドの登録と更新

前提条件

Director をインストールする前に、以下のタスクを完了してください。

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

手順

- コンテンツ配信ネットワークにシステムを登録します。要求されたら、カスタマーポータルของผู้ใช้名およびパスワードを入力します。

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

- 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。

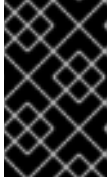
```
[stack@director ~]$ sudo subscription-manager list --available --all --matches="Red Hat
OpenStack"
Subscription Name:  Name of SKU
Provides:           Red Hat Single Sign-On
                   Red Hat Enterprise Linux Workstation
                   Red Hat CloudForms
                   Red Hat OpenStack
                   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
```

- 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。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。

```
[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 --enable=rhel-7-server-
rhceph-3-tools-rpms
```



重要

「[リポジトリの要件](#)」に記載されたりポジトリだけを有効にします。パッケージとソフトウェアの競合が発生する可能性があるため、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
```

6. システムをリブートします。

```
[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。

4.6. 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
```

4.7. CEPH-ANSIBLE のインストール

Red Hat OpenStack Platform で Ceph Storage を使用する場合、**ceph-ansible** パッケージが必要です。

Red Hat Ceph Storage を使用する場合、またはデプロイメントで外部の Ceph Storage クラスターを使用する場合、**ceph-ansible** パッケージをインストールします。既存 Ceph Storage クラスターとの統合についての詳しい情報は、[オーバークラウドの既存 Red Hat Ceph クラスターとの統合](#) を参照してください。

手順

1. Ceph Tools リポジトリを有効にします。

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
```

2. **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
```

4.8. 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ーに **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。このテンプレートを **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。

4.9. 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。エラーを避けるために、パラメーターは決して該当するセクションから削除しないでください。

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。設定すると、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。未設定のままにすると、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 経由の director のパブリック API エンドポイントに定義する IP アドレスまたはホスト名。director の設定により、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 director のソフトウェアブリッジに接続されます。

undercloud_admin_host

SSL/TLS 経由の director の管理 API エンドポイントに定義する IP アドレスまたはホスト名。director の設定により、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 director のソフトウェアブリッジに接続されます。

undercloud_nameservers

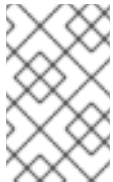
アンダークラウドのホスト名解決に使用する DNS ネームサーバーの一覧

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

オーバークラウドを設定する際に、**CloudDomain** パラメーターに `overcloud_domain_name` と同じ値を設定する必要があります。オーバークラウドを設定する際に、環境ファイルでこのパラメーターを設定します。

subnets

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、**ctiplane-subnet** サブネットのみです。

local_subnet

PXE ブートと DHCP インターフェイ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 **ctiplane-subnet** です。

masquerade_network

undercloud.conf ファイルの **[ctiplane-subnet]** セクションで **masquerade: true** と設定する場合に、**masquerade_network** パラメーターに空の値を定義する必要があります。

undercloud_service_certificate

OpenStack SSL/TLS 通信の証明書の場合とファイル名。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。このオプションは、オーバークラウドのデプロイに関係します。詳しくは、「[アンダークラウド CA を信頼するオーバークラウドノードの設定](#)」を参照してください。

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どうかを定義します。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 はこの証明書に署名します。このオプションは、オーバークラウドのデプロイに関係します。詳しくは、「[アンダークラウド CA を信頼するオーバークラウドノードの設定](#)」を参照してください。

certificate_generation_ca

要求した証明書に署名する 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イス。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。この値を選択したデバイスに変更します。接続されているデバイスを確認するには、**ip addr** コマンドを使用します。**ip addr** コマンドの出力結果の例を、以下に示します。

```
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、このインターフェイスが **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。アンダークラウドでは 1500 以下にします。

hieradata_override

director に Puppet hieradata を設定するための **hieradata** オーバーライドファイルへのパス。これにより、サービスに対して **undercloud.conf** パラメーター以外のカスタム設定を行うことができます。設定すると、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** ディレクトリにコピーされ、階層の最初のファイルに設定されます。この機能の使用方法については、「[アンダークラウドへの hieradata の設定](#)」を参照してください。

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。これを設定すると、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。ボンディングを設定する場合、またはインターフェイスにオプションを追加する場合に、このパラメーターを使用します。**/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_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。詳しくは、「[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。詳細は、[付録D 代替ブートモード](#)を参照してください。

enable_node_discovery

introspection 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

自動的に登録されたノード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。この値を **true** に設定して有効にします。

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。デフォルト値は **false** で、アンダークラウド上の telemetry が無効になります。このパラメーターは、Red Hat CloudForms などのメトリックデータを消費する他の製品を使用している場合に必要です。

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 インターフェイスを使用して、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。詳細は、[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。詳細は、[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

アンダークラウドで有効にするハードウェアタイプの一覧。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。

additional_architectures

オーバークラウドがサポートする (カーネル) アーキテクチャーの一覧。現在、このパラメーターの値は **ppc64le** だけに制限されています。



注記

ppc64le のサポートを有効にする場合には、**ipxe_enabled** を **False** に設定する必要もあります。

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_passwm** 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。特殊文字を使用すると構文エラーが発生する可能性があるため、これらのパスワードには英数字のみを使用してください。



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

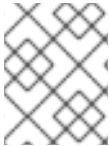
undercloud.conf ファイルには、各プロビジョニングサブネットの名前が付いたセクションがあります。たとえば、**ctlplane-subnet** という名前のサブネットを作成するとセクションは以下のようになります。

```
[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。

gateway

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。

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** サービス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。

masquerade

外部ネットワークへのアクセスのために、**cidr** で定義したネットワークをマスカレードするかどうかを定義します。このパラメーターにより、director 経由で外部ネットワークにアクセスすることができるよう、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) の一部メカニズムが提供されます。**masquerade** パラメーターを **true** に設定する場合には、**undercloud.conf** ファイルの **[DEFAULT]** セクションで **masquerade_network** パラメーターに空のマスカレードネットワーク値を定義する必要があります。

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。

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。値をコンマ区切り、この IP アドレス範囲の開始アドレスおよび終了アドレスを定義します。たとえば、**192.168.24.100,192.168.24.120** のように定義します。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。

これらのパラメーターの値は、設定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。

4.10. アンダークラウドへの HIERADATA の設定

director に Puppet hieradata を設定して、サービスに対して利用可能な **undercloud.conf** パラメーター以外のカスタム設定を行うことができます。この機能を使用するには、以下の手順を実施します。

手順

1. hieradata オーバーライドファイルを作成します (例: **/home/stack/hieradata.yaml**)。
2. カスタマイズした hieradata をファイルに追加します。たとえば、Compute (nova) サービスのパラメーター **force_raw_images** をデフォルト値の **True** から **False** に変更するには、以下の設定を追加します。

```
nova::compute::force_raw_images: False
```

設定するパラメーターに Puppet 実装がない場合には、以下の手段によりパラメーターを設定します。

```
nova::config::nova_config:
  DEFAULT/<parameter_name>:
    value: <parameter_value>
```

以下に例を示します。


```
nova::config::nova_config:
  DEFAULT/network_allocate_retries:
    value: 20
  ironic/serial_console_state_timeout:
    value: 15
```

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
```

4.11. 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を使用できるようになります。以下のコマンドで **stack** ユーザーのアクセス権限をリフレッシュします。

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

このコマンドを実行すると、再度ログインが求められます。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。

4.12. 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。

4.12.1. 単一 CPU アーキテクチャーのオーバークラウド

CPU アーキテクチャーがデフォルトの x86-64 の場合には、オーバークラウドのデプロイメントに以下のイメージおよび手順が必要です。

手順

1. source コマンドで **stackrc** ファイルを読み込み、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**) にイメージアーカイブを展開します。

```
(undercloud) [stack@director ~]$ mkdir ~/images
(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**

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. 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
```

4.12.2. 複数 CPU アーキテクチャーのオーバークラウド

追加の CPU アーキテクチャーのサポートを有効にしてオーバークラウドをデプロイするには、以下のイメージおよび手順が必要です。現在、追加の CPU アーキテクチャーは ppc64le Power アーキテクチャーに限定されています。

手順

1. source コマンドで **stackrc** ファイルを読み込み、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-all** パッケージをインストールします。

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

3. アーキテクチャー個別のディレクトリーにアーカイブを展開します。このディレクトリーは、**stack** ユーザーのホーム内の **images** ディレクトリー (**/home/stack/images**) 下に作成します。

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ for arch in x86_64 ppc64le ; do mkdir $arch ; done
(undercloud) [stack@director images]$ for arch in x86_64 ppc64le ; do for i in
/usr/share/rhosp-director-images/overcloud-full-latest-13.0-${arch}.tar /usr/share/rhosp-
director-images/ironic-python-agent-latest-13.0-${arch}.tar ; do tar -C $arch -xf $i ; done ;
done
```

4. これらのイメージを director にインポートします。

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ openstack overcloud image upload --image-path
~/images/ppc64le --architecture ppc64le --whole-disk --http-boot /tftpboot/ppc64le
(undercloud) [stack@director images]$ openstack overcloud image upload --image-path
~/images/x86_64/ --http-boot /tftpboot
```

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**
- **ppc64le-bm-deploy-kernel**
- **ppc64le-bm-deploy-ramdisk**
- **ppc64le-overcloud-full**
スクリプトにより、director 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. 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                               | Status |
+-----+-----+-----+
| 6d1005ba-ec82-473b-8e33-88aadb5b6792 | bm-deploy-kernel                   | active |
| fb723b33-9f11-45f5-b25b-c008bf509290 | bm-deploy-ramdisk                  | active |
| 6a6096ba-8f79-4343-b77c-4349f7b94960 | overcloud-full                     | active |
| de2a1bde-9351-40d2-bbd7-7ce9d6eb50d8 | overcloud-full-initrd              | active |
| 67073533-dd2a-4a95-8e8b-0f108f031092 | overcloud-full-vmlinuz             | active |
| 69a9ffe5-06dc-4d81-a122-e5d56ed46c98 | ppc64le-bm-deploy-kernel           | active |
| 464dd809-f130-4055-9a39-cf6b63c1944e | ppc64le-bm-deploy-ramdisk          | active |
| f0fedcd0-3f28-4b44-9c88-619419007a03 | ppc64le-overcloud-full             | active |
+-----+-----+-----+
```

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 **/tftpboot** にコピーします。

```
(undercloud) [stack@director images]$ ls -l /tftpboot/tftpboot/ppc64le/
/tftpboot:
total 422624
-rwxr-xr-x. 1 root root 6385968 Aug 8 19:35 agent.kernel
-rw-r--r--. 1 root root 425530268 Aug 8 19:35 agent.ramdisk
-rwxr--r--. 1 ironic ironic 20832 Aug 8 02:08 chain.c32
-rwxr--r--. 1 ironic ironic 715584 Aug 8 02:06 ipxe.efi
-rw-r--r--. 1 root root 22 Aug 8 02:06 map-file
drwxr-xr-x. 2 ironic ironic 62 Aug 8 19:34 ppc64le
-rwxr--r--. 1 ironic ironic 26826 Aug 8 02:08 pxelinux.0
drwxr-xr-x. 2 ironic ironic 21 Aug 8 02:06 pxelinux.cfg
-rwxr--r--. 1 ironic ironic 69631 Aug 8 02:06 undionly.kpxe

/tftpboot/ppc64le/:
total 457204
-rwxr-xr-x. 1 root root 19858896 Aug 8 19:34 agent.kernel
-rw-r--r--. 1 root root 448311235 Aug 8 19:34 agent.ramdisk
-rw-r--r--. 1 ironic-inspector ironic-inspector 336 Aug 8 02:06 default
```

4.12.3. 最小限のオーバークラウドイメージ

他の Red Hat OpenStack Platform サービスを実行したくない場合や、サブスクリプションエンタイトルメントの1つを消費したくない場合に、**overcloud-minimal** イメージを使用してベア OS をプロビジョニングすることができます。

手順

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

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

2. **overcloud-minimal** パッケージをインストールします。

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

3. イメージのアーカイブを、**stack** ユーザーのホームディレクトリー下の **images** ディレクトリー (**/home/stack/images**) に展開します。

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ tar xf /usr/share/rhosp-director-images/overcloud-minimal-latest-13.0.tar
```

4. 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/ --os-image-name overcloud-minimal.qcow2
```

このスクリプトにより、以下のイメージが director にアップロードされます。

- **overcloud-minimal**

- **overcloud-minimal-initrd**
- **overcloud-minimal-vmlinuz**

5. 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                               |
+-----+-----+
| ef793cd0-e65c-456a-a675-63cd57610bd5 | overcloud-full                    |
| 9a51a6cb-4670-40de-b64b-b70f4dd44152 | overcloud-full-initrd             |
| 4f7e33f4-d617-47c1-b36f-cbe90f132e5d | overcloud-full-vmlinuz            |
| 32cf6771-b5df-4498-8f02-c3bd8bb93fdd | overcloud-minimal                 |
| 600035af-dbbb-4985-8b24-a4e9da149ae5 | overcloud-minimal-initrd          |
| d45b0071-8006-472b-bbcc-458899e0d801 | overcloud-minimal-vmlinuz         |
+-----+-----+
```



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。ただし、完全なディスクイメージをインポートして使用することも可能です。詳しくは、[付録C 完全なディスクイメージ](#)を参照してください。

4.13. 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. source コマンドで **stackrc** ファイルを読み込み、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。

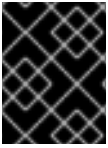
```
(undercloud) [stack@director images]$ openstack subnet set --dns-nameserver
[nameserver1-ip] --dns-nameserver [nameserver2-ip] ctlplane-subnet
```

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 **DnsServers** パラメーターを使用します。

4.14. 次のステップ

これで director の設定およびインストールが完了しました。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

すべてのオーバークラウドサービスがコンテナ化されます。したがって、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となります。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。

- 本ガイドには、オーバークラウドを設定してレジストリーを使用するさまざまなユースケースを記載しています。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、「[コンテナイメージの準備コマンドの使用法](#)」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 では、以下のレジストリータイプがサポートされています。

リモートレジストリー

オーバークラウドは、**registry.redhat.io** 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。ただし、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

アンダークラウドは、**docker-distribution** サービスを使用してレジストリーとして機能します。これにより、director は **registry.redhat.io** からプルしたイメージを同期し、それを **docker-distribution** レジストリーにプッシュすることができます。オーバークラウドを作成する際に、オーバークラウドはアンダークラウドの **docker-distribution** レジストリー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。



注記

docker-distribution サービスは、**docker** とは別に動作します。**docker** は、イメージを **docker-distribution** レジストリーにプッシュおよびプルするのに使用されますが、イメージをオーバークラウドに提供することはありません。オーバークラウドが **docker-distribution** レジストリーからイメージをプルします。

Satellite Server

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。



注記

マルチアーキテクチャクラウドの構築では、ローカルレジストリーのオプションはサポートされません。

5.2. コンテナイメージの準備コマンドの使用方法

本項では、**openstack overcloud container image prepare** コマンドの使用方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環境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メントコマンドで指定します。**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-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.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  DockerAodhConfigImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  ...
```

環境ファイルには、**DockerInsecureRegistryAddress** パラメーターもアンダークラウドレジストリーの IP アドレスとポートに設定されます。このパラメーターにより、SSL/TLS 証明書なしにアンダークラウドレジストリーからイメージにアクセスするオーバークラウドノードが設定されます。

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.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  - imagename: registry.redhat.io/rhosp13/openstack-aodh-evaluator:13.0-34
  ...
```

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接頭辞を定義します。

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

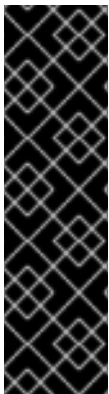
--tag および **--tag-from-label** オプションを併用して、各コンテナイメージのタグを設定します。

--tag

ソースからの全イメージに特定のタグを設定します。このオプションだけを使用した場合、director はこのタグを使用してすべてのコンテナイメージをプルします。ただし、このオプションを **--tag-from-label** の値と共に使用する場合は、director はソースイメージとして **--tag** を使用して、ラベルに基づいて特定のバージョンタグを識別します。**--tag** オプションは、デフォルトで **latest** に設定されます。

--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付きタグを検出してプルします。director は **--tag** に設定した値がタグ付けされた各コンテナイメージを検査し、続いてコンテナイメージラベルを使用して新しいタグを構築し、レジストリーからプルします。たとえば、**--tag-from-label {version}-{release}** を設定すると、director は **version** および **release** ラベルを使用して新しいタグを作成します。あるコンテナについて、**version** を **13.0** に設定し、**release** を **34** に設定した場合、タグは **13.0-34** となります。

**重要**

Red Hat コンテナレジストリーでは、すべての Red Hat OpenStack Platform コンテナイメージをタグ付けするのに、特定のバージョン形式が使用されます。このバージョン形式は **{version}-{release}** で、各コンテナイメージがコンテナメタデータのラベルとして保存します。このバージョン形式は、ある **{release}** から次のリリースへの更新を容易にします。このため、**openstack overcloud container image prepare** コマンドを実行する際には、必ず **--tag-from-label {version}-{release}** を使用する必要があります。コンテナイメージをプルするのに **--tag** だけを単独で使用しないでください。たとえば、**--tag latest** を単独で使用する、更新の実行時に問題が発生します。director は、コンテナイメージを更新するのにタグの変更を必要とするためです。

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ラー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	environments/ceph-ansible/ceph-ansible.yaml
Collectd	environments/services-docker/collectd.yaml
Congress	environments/services-docker/congress.yaml
Fluentd	environments/services-docker/fluentd.yaml
OpenStack Bare Metal (ironic)	environments/services-docker/ironic.yaml
OpenStack Data Processing (sahara)	environments/services-docker/sahara.yaml
OpenStack EC2-API	environments/services-docker/ec2-api.yaml
OpenStack Key Manager (barbican)	environments/services-docker/barbican.yaml
OpenStack Load Balancing-as-a-Service (octavia)	environments/services-docker/octavia.yaml
OpenStack Shared File System Storage (manila)	environments/manila-{backend-name}-config.yaml 注記: 詳細は、 OpenStack Shared File System (manila) を参照してください。
Open Virtual Network (OVN)	environments/services-docker/neutron-ovn-dvr-ha.yaml
Sensu	environments/services-docker/sensu-client.yaml

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 クラスター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 Ceph Storage 固有のパラメーターを設定してください。

--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.redhat.io/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。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 \
...
```

OpenStack Neutron SR-IOV

オーバークラウドで OpenStack Neutron SR-IOV をデプロイする場合には、director がイメージを準備できるように **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml** 環境ファイルを追加します。デフォルトの Controller ロールおよび Compute ロールは SR-IOV サービスをサポートしないため、**-r** オプションを使用して SR-IOV サービスが含まれるカスタムロールファイルも追加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 \
...
-r ~/custom_roles_data.yaml
-e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml \
...
```

OpenStack Load Balancing-as-a-Service (octavia)

オーバークラウドで OpenStack Load Balancing-as-a-Service をデプロイする場合には、director がイメージを準備できるように `/usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml` 環境ファイルを追加します。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/octavia.yaml
\
...
```

OpenStack Shared File System (manila)

`manila-{backend-name}-config.yaml` のフォーマットを使用してサポート対象のバックエンドを選択し、そのバックエンドを用いて Shared File System をデプロイすることができます。以下の環境ファイルから任意のファイルを追加して、Shared File System サービスのコンテナを準備することができます。

```
environments/manila-isilon-config.yaml
environments/manila-netapp-config.yaml
environments/manila-vmax-config.yaml
environments/manila-cephfsnative-config.yaml
environments/manila-cephfsганеша-config.yaml
environments/manila-unity-config.yaml
environments/manila-vnx-config.yaml
```

環境ファイルのカスタマイズおよびデプロイに関する詳細は、以下の資料を参照してください。

- Shared File System サービスの NFS バックエンドに CephFS を使用した場合のガイドの [更新された環境のデプロイ](#)
- NetApp Back End Guide for the Shared File System Service の [Deploy the Shared File System Service with NetApp Back Ends](#)
- CephFS Back End Guide for the Shared File System Service の [Deploy the Shared File System Service with a CephFS Back End](#)

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.redhat.io 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のが最も簡単な方法です。レジストリーはすでに設定済みで、プルするイメージの URL と名前空間を指定するだけで良いからです。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レジストリー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。したがって、実稼働環境ではこの方法は推奨されません。実稼働環境用には、この方法ではなく以下のいずれかの方法を使用してください。

- 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.redhat.io** 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。以下のコマンドを実行してコンテナイメージの環境ファイルを生成します。

```
(undercloud) $ sudo openstack overcloud container image prepare \
--namespace=registry.redhat.io/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。
 - カスタムロールファイルを指定するには、**-r** 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. **overcloud_images.yaml** ファイルを変更し、デプロイメント時に **registry.redhat.io** との間で認証を行うために以下のパラメーターを追加します。

```
ContainerImageRegistryLogin: true
ContainerImageRegistryCredentials:
  registry.redhat.io:
    <USERNAME>: <PASSWORD>
```

- **<USERNAME>** および **<PASSWORD>** を **registry.redhat.io** の認証情報に置き換えます。**overcloud_images.yaml** ファイルには、アンダークラウド上のイメージの場所が含まれます。このファイルをデプロイメントに追加します。



注記

openstack overcloud deploy コマンドを実行する前に、リモートレジストリーにログインする必要があります。

```
(undercloud) $ sudo docker login registry.redhat.io
```

レジストリーの設定が完了しました。

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。

director を使用して、**registry.redhat.io** から各イメージをプルし、アンダークラウドで実行する **docker-distribution** レジストリーに各イメージをプッシュできます。director を使用してオーバークラウドを作成する場合は、オーバークラウドの作成プロセス中に、ノードは関連するイメージをアンダークラウドの **docker-distribution** レジストリーからプルします。

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。アドレスは次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. **registry.redhat.io** にログインします。

```
(undercloud) $ docker login registry.redhat.io --username $RH_USER --password $RH_PASSWD
```

3. 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.redhat.io/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。
- カスタムロールファイルを指定するには、**-r** 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**

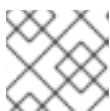
4. 次の 2 つのファイルが作成されていることを確認します。

- 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.redhat.io**) からイメージをアンダークラウドにプルします。
- 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**。このファイルをデプロイメントで指定します。

5. コンテナイメージをリモートレジストリーからプルし、アンダークラウドレジストリーにプッシュします。

```
(undercloud) $ 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。

6. これで、イメージがアンダークラウドの **docker-distribution** レジストリーに保管されます。アンダークラウドの **docker-distribution** レジストリーのイメージ一覧を表示するには、以下のコマンドを実行します。

```
(undercloud) $ curl http://192.168.24.1:8787/v2/_catalog | jq .repositories[]
```



注記

_catalog リソース自体は、イメージを 100 個のみ表示します。追加のイメージを表示するには、**?n=<interger>** クエリー文字列を **_catalog** リソースと共に使用して、多数のイメージを表示します。

```
(undercloud) $ curl http://192.168.24.1:8787/v2/_catalog?n=150 | jq .repositories[]
```

特定イメージのタグの一覧を表示するには、**skopeo** コマンドを使用します。

```
(undercloud) $ curl -s http://192.168.24.1:8787/v2/rhosp13/openstack-keystone/tags/list | jq .tags
```

タグ付けられたイメージを検証するには、**skopeo** コマンドを使用します。

```
(undercloud) $ skopeo inspect --tls-verify=false
docker://192.168.24.1:8787/rhosp13/openstack-keystone:13.0-44
```

レジストリーの設定が完了しました。

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 Server 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。
- カスタムロールファイルを指定するには、**-r** 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

上記の **openstack overcloud container image prepare** コマンドは、**registry.redhat.io** のレジストリーをターゲットにしてイメージの一覧を生成します。この後のステップでは、**openstack overcloud container image prepare** コマンドで別の値を使用します。

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 6 の **hammer** ツールがインストールされているシステムに **satellite_images_names** ファイルをコピーします。あるいは、[Hammer CLI ガイド](#) に記載の手順に従って、アンダークラウドに **hammer** ツールを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.redhat.io \
  --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.redhat.io \
--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 Server 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 Server 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。[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バージョンを作成して、イメージを取り入れます。
10. **base** イメージに使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 のレジストリーポートは 5000 です。たとえば、**--namespace=satellite6.example.com:5000** のように設定します。



注記

Red Hat Satellite バージョン 6.10 を使用している場合は、ポートを指定する必要はありません。デフォルトのポート **443** が使用されます。詳細は、"[How can we adapt RHOSP13 deployment to Red Hat Satellite 6.10?](#)" を参照してください。

- **--prefix=** - この接頭辞は、ラベルの Satellite 6 の命名規則に基づいており、この接頭辞は小文字を使用し、アンダースコアの代わりにスペースを使用します。この接頭辞は、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、設定は **[org]-[environment]-[content view]-[product]-** です。たとえば、**acme-production-myosp13-osp13_containers-** のようになります。
 - コンテンツビューを使用しない場合、設定は **[org]-[product]-** です。たとえば、**acme-osp13_containers-** のようになります。
- **--tag-from-label {version}-{release}**: 各イメージの最新のタグを識別します。
- **-e**: オプションのサービスの環境ファイルを指定します。
- **-r**: カスタムロール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 固有の接頭辞が追加された点に注意してください。この接頭辞は、**--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。

12. **overcloud_images.yaml** ファイルには、Satellite サーバー上のイメージの場所が含まれます。このファイルをデプロイメントに追加します。

レジストリーの設定が完了しました。

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、カスタム機能は含まれません。ただし、[オーバークラウドの高度なカスタマイズ](#)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、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。これ以外のサポート対象電源管理ドライバー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。

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、各ノード種別に設定された要件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
コンピューター	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。ただし、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":"ipmi",
      "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":"ipmi",
      "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。この例で使用的是 IPMI ドライバー (**ipmi**) で、電源管理の推奨ドライバーです。



注記

IPMI が推奨されるサポート対象電源管理ドライバーです。これ以外のサポート対象電源管理ドライバー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。それらの電源管理ドライバーが想定どおりに機能しない場合には、電源管理に IPMI を使用してください。

pm_user、pm_password

IPMI のユーザー名およびパスワード。IPMI および Redfish では、これらの属性は任意です。また、iLO および iDRAC では必須です。

pm_addr

IPMI デバイスの IP アドレス

pm_port

(オプション) 特定の IPMI デバイスにアクセスするためのポート

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

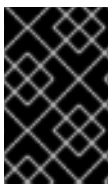
(オプション) メモリーサイズ (MB 単位)

disk

(オプション) ハードディスクのサイズ (GB 単位)

arch

(オプション) システムアーキテクチャー

**重要**

マルチアーキテクチャクラウドをビルドする場合には、**x86_64** アーキテクチャーを使用するノードと **ppc64le** アーキテクチャーを使用するノードを区別するために **arch** キーが必須です。

テンプレートを作成したら、以下のコマンドを実行してフォーマットおよび構文を検証します。

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

stack ユーザーのホームディレクトリーにファイルを保存し (**/home/stack/instackenv.json**)、続いて以下のコマンドを実行してテンプレートを director にインポートします。

```
(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ブートします。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ーリングします。

```
(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。ベアメタルノードの場合には、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。

ノードのイントロスペクション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 data save <UUID> | jq .
```

<UUID> をイントロスペクション情報を取得するノードの UUID に置き換えてください。

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イス情報の取得

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イスに関する 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イ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イ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]
```

以下に例を示します。

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

インターフェイ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]
```

以下に例を示します。

```
(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** のインストールと使用方法の [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、この追加ハードウェアイントロスペクションの一部で、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 = ipmi

```

- **enable_node_discovery**: 有効にすると、PXE を使って introspection 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: **ipmi**

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。

```
$ 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。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},
      {"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. アーキテクチャーに固有なロールの生成

マルチアーキテクチャクラウドをビルドする場合には、**roles_data.yaml** にアーキテクチャー固有のロールを追加する必要があります。以下に示す例では、デフォルトのロールに加えて

ComputePPC64LE ロールを追加しています。[roles_data ファイルの作成](#) セクションには、ロールについての情報が記載されています。

```
openstack overcloud roles generate \
  --roles-path /usr/share/openstack-tripleo-heat-templates/roles -o ~/templates/roles_data.yaml \
  Controller Compute ComputePPC64LE BlockStorage ObjectStorage CephStorage
```

6.5. 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、検査した後には、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	Controller 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	control フレーバーは、ノードをコントローラーとして使用するため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 director が決定できるように、このフレーバーを Controller ロールに割り当てます。
プロファイル	control プロファイルは、 control フレーバーに適用するタグです。これにより、フレーバーに属するノードが定義されます。
ノード	また、各ノードに control プロファイルタグを適用して、 control フレーバーにグループ化します。これにより、director が Controller 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。詳しくは、[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ープロファイルとコンピュータ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。詳細は、「[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
```

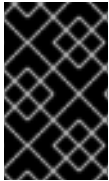
6.6. ルートディスクの定義

ノードで複数のディスクが使用されている場合には、director はプロビジョニング時にルートディスクを特定する必要があります。たとえば、ほとんどの Ceph Storage ノードでは、複数のディスクが使用されます。デフォルトのプロビジョニングプロセスでは、director はルートディスクにオーバークラウドイメージを書き込みます。

以下のプロパティーを定義すると、director がルートディスクを特定するのに役立ちます。

- **model** (文字列): デバイスの ID
- **vendor** (文字列): デバイスのベンダー
- **serial** (文字列): ディスクのシリアル番号
- **hctl** (文字列): SCSI のホスト、チャンネル、ターゲット、Lun
- **size** (整数): デバイスのサイズ (GB 単位)

- **wwn** (文字列): 一意のストレージ ID
- **wwn_with_extension** (文字列): ベンダー拡張子を追加した一意のストレージ ID
- **wwn_vendor_extension** (文字列): 一意のベンダーストレージ ID
- **rotational** (ブール値): 回転式デバイス (HDD) には true、そうでない場合 (SSD) には false
- **name** (文字列): デバイス名 (例: /dev/sdb1)。
- **by_path** (文字列): デバイスの一意の PCI パス。デバイスの UUID を使用しない場合は、このプロパティを使用。



重要

name プロパティは、永続的なデバイス名が付いたデバイスにのみ使用します。他のデバイスのルートディスクを設定する際に、**name** を使用しないでください。この値は、ノードの起動時に変更される可能性があります。

シリアル番号を使用してルートデバイスを指定するには、以下の手順を実施します。

手順

1. 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。以下のコマンドを実行して、ノード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"
  }
]

```

2. ノード定義の **root_device** パラメーターに変更を加えます。以下の例では、ルートデバイスをシリアル番号が **61866da04f380d001ea4e13c12e36ad6** のディスク 2 に設定する方法を説明します。

```

(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を含めるようにします。最初にネットワークからの起動を試み、次にルートディスクからの起動を試みるように、起動順序を設定します。

director は、ルートディスクとして使用する特定のディスクを把握します。**openstack overcloud deploy** コマンドを実行すると、director はオーバークラウドをプロビジョニングし、ルートディスクにオーバークラウドのイメージを書き込みます。

6.7. OVERCLOUD-MINIMAL イメージの使用による RED HAT サブスクリプションエンタイトルメントの使用回避

デフォルトでは、プロビジョニングプロセス中 director はルートディスクに QCOW2 **overcloud-full** イメージを書き込みます。**overcloud-full** イメージには、有効な Red Hat サブスクリプションが使用されます。ただし、**overcloud-minimal** イメージを使用して、たとえばベア OS をプロビジョニングすることもできます。この場合、他の OpenStack サービスは使用されないため、サブスクリプションエンタイトルメントは消費されません。

この典型的なユースケースは、Ceph デーモンのみを持つノードをプロビジョニングする場合です。この場合や類似のユースケースでは、**overcloud-minimal** イメージのオプションを使用して、有償の Red Hat サブスクリプションが限度に達するのを避けることができます。**overcloud-minimal** イメージの取得方法についての情報は、[オーバークラウドノードのイメージの取得](#) を参照してください。

手順

1. **overcloud-minimal** イメージを使用するように director を設定するには、以下のイメージ定義を含む環境ファイルを作成します。

```

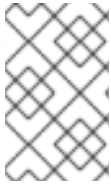
parameter_defaults:
  <roleName>Image: overcloud-minimal

```

2. **<roleName>** をロール名に置き換え、ロール名に **Image** を追加します。Ceph Storage ノードの **overcloud-minimal** イメージの例を以下に示します。

```
parameter_defaults:
  CephStorageImage: overcloud-minimal
```

3. この環境ファイルを **openstack overcloud deploy** コマンドに渡します。



注記

OVS は OpenStack サブスクリプションエンタイトルメントが必要な OpenStack サービスです。したがって、**overcloud-minimal** イメージでサポートされるのは標準の Linux ブリッジだけで、OVS はサポートされません。

6.8. ノード数とフレーバーを定義する環境ファイルの作成

デフォルトでは、director は **baremetal** フレーバーを使用して1つのコントローラーノードとコンピュートノードを持つオーバークラウドをデプロイします。ただし、この設定は概念検証のためのデプロイメントにしか適しません。異なるノード数およびフレーバーを指定して、デフォルトの設定をオーバーライドすることができます。小規模な実稼働環境では、コントローラーノードとコンピュートノードを少なくとも3つにし、適切なリソース仕様でノードが作成されるように特定のフレーバーを割り当てる必要があります。以下の手順では、ノード数およびフレーバー割り当てを定義する環境ファイル **node-info.yaml** の作成方法を説明します。

1. **/home/stack/templates/** ディレクトリーに **node-info.yaml** ファイルを作成します。

```
(undercloud) $ touch /home/stack/templates/node-info.yaml
```

2. ファイルを編集し、必要なノード数およびフレーバーを設定します。以下の例では、3つのコントローラーノード、コンピュートノード、および Ceph Storage ノードをデプロイします。

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

このファイルは、この後「[オーバークラウド作成時の環境ファイルの追加](#)」で使用します。

6.9. アンダークラウド CA を信頼するオーバークラウドノードの設定

アンダークラウドで TLS が使用され CA が一般に信頼できない場合には、以下の手順に従う必要があります。アンダークラウドは、SSL エンドポイントの暗号化のために自己の認証局 (CA) を運用します。デプロイメントの他の要素からアンダークラウドのエンドポイントにアクセスできるようにするには、アンダークラウドの CA を信頼するようにオーバークラウドノードを設定します。



注記

この手法が機能するためには、オーバークラウドノードにアンダークラウドの公開エンドポイントへのネットワークルートが必要です。スパイン/リーフ型ネットワークに依存するデプロイメントでは、この設定を適用する必要があります。

アンダークラウドの証明書の概要

アンダークラウドで使用するのことができるカスタム証明書には、2つのタイプがあります。ユーザーの提供する証明書と自動的に生成される証明書です。

- ユーザーの提供する証明書: 自己の証明書を提供している場合がこれに該当します。自己の CA からの証明書、または自己署名の証明書がその例です。この証明書は **undercloud_service_certificate** オプションを使用して渡されます。この場合、自己署名の証明書または CA のどちらかを信頼する必要があります (デプロイメントによります)。
- 自動生成される証明書: **certmonger** により自己のローカル CA を使用して証明書を生成する場合がこれに該当します。このプロセスは、**generate_service_certificate** オプションを使用して有効にします。この場合、CA 証明書 (**/etc/pki/ca-trust/source/anchors/cm-local-ca.pem**) およびアンダークラウドの HAProxy インスタンスが使用するサーバー証明書が用いられます。この証明書を OpenStack に提示するには、CA 証明書を **inject-trust-anchor-hiera.yaml** ファイルに追加する必要があります。

undercloud_service_certificate および **generate_service_certificate** オプションの説明および使用方法については、「[director の設定パラメーター](#)」を参照してください。

アンダークラウドでのカスタム証明書の使用

以下の例では、**/home/stack/ca.crt.pem** に保存された自己署名の証明書が使われています。自動生成される証明書を使用する場合には、代わりに **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** を使用する必要があります。

1. 証明書ファイルを開き、証明書部分だけをコピーします。鍵を含めないでください。

```
$ vi /home/stack/ca.crt.pem
```

必要となる証明書部分の例を、以下に示します。

```
-----BEGIN CERTIFICATE-----
MIIDITCCAn2gAwIBAgIJAOOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV
BAYTAIVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECg
wH
UmVkiEhhdDELMAkGA1UECwwCUUUxXFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
-----END CERTIFICATE-----
```

2. 以下に示す内容で **/home/stack/inject-trust-anchor-hiera.yaml** という名称の新たな YAML ファイルを作成し、PEM ファイルからコピーした証明書を追加します。

```
parameter_defaults:
  CAMap:
    overcloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV
        BAYTAIVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECg
        wH
        UmVkiEhhdDELMAkGA1UECwwCUUUxXFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
        -----END CERTIFICATE-----
    undercloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV
```

```
BAYTAIVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECg
wH
UmVkiEhhdDELMAkGA1UECwwCUUUxFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
-----END CERTIFICATE-----
```



注記

証明書の文字列は PEM の形式に従い、**content** パラメーター内では正しい YAML インデントを使用する必要があります。

オーバークラウドのデプロイメント時に CA 証明書がそれぞれのオーバークラウドノードにコピーされ、アンダークラウドの SSL エンドポイントが提示する暗号化を信頼するようになります。環境ファイル追加の詳細は、「[オーバークラウド作成時の環境ファイルの追加](#)」を参照してください。

6.10. 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。ただし、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サーを使用している場合には、外部の負荷分散機能の環境ファイル。詳しい情報は、[オーバークラウド用の外部ロードバランサー](#)を参照してください。
- 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詳細は、[オーバークラウドの登録](#)を参照してください。
- その他のカスタム環境ファイル



注記

/usr/share/openstack-tripleo-heat-templates/environments ディレクトリーには、コンテナ化されたサービスを有効にする環境ファイル (**docker.yaml** および **docker-ha.yaml**) が含まれます。OpenStack Platform director は、オーバークラウドのデプロイメント時にこれらのファイルを自動的に追加します。デプロイコマンドでこれらのファイルを手動で追加しないでください。

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

[オーバークラウドの高度なカスタマイズ](#) 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することができます。

Heat テンプレートおよび環境ファイルの詳細については、オーバークラウドの高度なカスタマイズの [Heat テンプレートの理解](#) の章を参照してください。



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。

6.11. 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
--templates [TEMPLATES]	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。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 /usr/share/openstack-tripleo-heat-templates/ を使用します。
--stack STACK	作成または更新するスタックの名前
-t [TIMEOUT]、--timeout [TIMEOUT]	デプロイメントのタイムアウト (分単位) このオプションを keystone トークンのタイムアウト制限 (デフォルトでは 240 分) よりも高い値に設定しないでください。

パラメーター	説明
--libvirt-type [LIBVIRT_TYPE]	ハイパーバイザーに使用する仮想化タイプ
--ntp-server [NTP_SERVER]	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: --ntp-server 0.centos.pool.org,1.centos.pool.org)。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。
--no-proxy [NO_PROXY]	環境変数 no_proxy 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。
--overcloud-ssh-user OVERCLOUD_SSH_USER	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 heat-admin ユーザーで実行されます。
-e [EXTRA HEAT TEMPLATE]、--extra-template [EXTRA HEAT TEMPLATE]	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。 openstack overcloud deploy 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。
--environment-directory	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、その後にアルファベット順で処理します。
--validation-errors-nonfatal	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。
--validation-warnings-fatal	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。
--dry-run	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。

パラメーター	説明
--skip-postconfig	オーバークラウドのデプロイ後の設定を省略します。
--force-postconfig	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。
--skip-deploy-identifier	DeployIdentifier パラメーターの一意の 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。
--answers-file ANSWERS_FILE	引数とパラメーターが記載された YAML ファイルへのパス。
--rhel-reg	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。
--reg-method	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 の場合は satellite 、カスタマーポータルの場合は portal に設定します。
--reg-org [REG_ORG]	登録に使用する組織
--reg-force	すでに登録済みでもシステムを登録します。
--reg-sat-url [REG_SAT_URL]	オーバークラウドノードを登録する Satellite サーバーのベース URL。このパラメーターには、HTTPS URL ではなく、Satellite の HTTP URL を使用します。たとえば、 https://satellite.example.com ではなく http://satellite.example.com を使用します。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 katello-ca-consumer-latest.noarch.rpm ファイルを取得して subscription-manager に登録し、 katello-agent 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 RHN-ORG-TRUSTED-SSL-CERT ファイルを取得して rhncat に登録します。
--reg-activation-key [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
--control-scale	スケールアウトするコントローラーノード数	ControllerCount
--compute-scale	スケールアウトするコンピューターノード数	ComputeCount
--ceph-storage-scale	スケールアウトする Ceph Storage ノードの数	CephStorageCount
--block-storage-scale	スケールアウトする Cinder ノード数	BlockStorageCount
--swift-storage-scale	スケールアウトする Swift ノード数	ObjectStorageCount
--control-flavor	コントローラーノードに使用するフレーバー	OvercloudControllerFlavor
--compute-flavor	コンピューターノードに使用するフレーバー	OvercloudComputeFlavor
--ceph-storage-flavor	Ceph Storage ノードに使用するフレーバー	OvercloudCephStorageFlavor
--block-storage-flavor	Cinder ノードに使用するフレーバー	OvercloudBlockStorageFlavor
--swift-storage-flavor	Swift Storage ノードに使用するフレーバー	OvercloudSwiftStorageFlavor
--neutron-flat-networks	フラットなネットワークが neutron 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 datacentre に設定されています。	NeutronFlatNetworks

パラメーター	説明	Heat テンプレートのパラメーター
--neutron-physical-bridge	各ハイパーバイザーで作成する Open vSwitch ブリッジ。デフォルトは br-ex です。通常、このパラメーターを変更する必要はありません。	HypervisorNeutronPhysicalBridge
--neutron-bridge-mappings	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。ホストの外部ブリッジ (br-ex) を物理名 (datacentre) 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。	NeutronBridgeMappings
--neutron-public-interface	ネットワークノード向けに br-ex にブリッジするインターフェイスを定義します。	NeutronPublicInterface
--neutron-network-type	Neutron のテナントネットワーク種別	NeutronNetworkType
--neutron-tunnel-types	neutron 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。	NeutronTunnelTypes
--neutron-tunnel-id-ranges	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	NeutronTunnelIdRanges
--neutron-vni-ranges	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	NeutronVniRanges
--neutron-network-vlan-ranges	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 datacentre 上の任意の VLAN を許可するように設定されています。	NeutronNetworkVLANRanges
--neutron-mechanism-drivers	neutron テナントネットワークのメカニズムドライバー。デフォルトは openvswitch です。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。	NeutronMechanismDrivers

パラメーター	説明	Heat テンプレートのパラメーター
--neutron-disable-tunneling	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
--validation-errors-fatal	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廃止される予定です。



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
```

6.12. 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。ただし、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サーを使用している場合には、外部の負荷分散機能の環境ファイル。詳しい情報は、[オーバークラウド用の外部ロードバランサー](#)を参照してください。
- 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詳細は、[オーバークラウドの登録](#)を参照してください。

- その他のカスタム環境ファイル



注記

/usr/share/openstack-tripleo-heat-templates/environments ディレクトリーには、コンテナ化されたサービスを有効にする環境ファイル (**docker.yaml** および **docker-ha.yaml**) が含まれます。OpenStack Platform director は、オーバークラウドのデプロイメント時にこれらのファイルを自動的に追加します。デプロイコマンドでこれらのファイルを手動で追加しないでください。

-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 \
-e /home/stack/inject-trust-anchor-hiera.yaml \
-r /home/stack/templates/roles_data.yaml \
--ntp-server pool.ntp.org \
```

上記のコマンドでは、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。

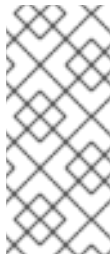
```
parameter_defaults:
  OvercloudControllerFlavor: control
  OvercloudComputeFlavor: compute
  OvercloudCephStorageFlavor: ceph-storage
  ControllerCount: 3
  ComputeCount: 3
  CephStorageCount: 3
```

-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。

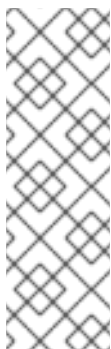


注記

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。



注記

openstack overcloud netenv validate コマンドを実行して、**network-environment.yaml** ファイルの構文を検証します。このコマンドにより、Compute、コントローラー、ストレージ、およびコンポーザブルロールのネットワークファイルの個別 nic-config ファイルも検証されます。**-f** または **--file** オプションを使用して、検証するファイルを指定します。

```
$ openstack overcloud netenv validate -f ~/templates/network-environment.yaml
```

-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。

-e /home/stack/inject-trust-anchor-hiera.yaml

アンダークラウドにカスタム証明書をインストールする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。

-r /home/stack/templates/roles_data.yaml

(オプション) カスタムロールを使用する、またはマルチアーキテクチャクラウドを有効にする場合に生成されるロールデータ。詳しくは、[「アーキテクチャーに固有なロールの生成」](#)を参照してください。

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。

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、ファイルの

処理順を制御するために、ファイル名に数字の接頭辞を使用することを推奨します。以下に例を示します。

```
(undercloud) $ ls -l ~/templates
00-node-info.yaml
10-overcloud_images.yaml
20-network-isolation.yaml
30-network-environment.yaml
40-storage-environment.yaml
50-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。

templates

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。

environments

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
```

オーバークラウドの設定および環境ファイル管理のガイドライン

以下のガイドラインを使用して、環境ファイルおよびオーバークラウド設定を管理してください。

- コア Heat テンプレートを直接変更しないでください。望ましくない結果につながり、環境が壊れる可能性があります。環境ファイルを使用してオーバークラウドの設定を変更します。
- オーバークラウドの設定は直接編集しないでください。手動で設定しても、director で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います。環境ファイルを介してオーバークラウドの設定を変更し、デプロイメントコマンドを再実行します。
- deploy コマンドを含む bash スクリプトを作成し、オーバークラウドの更新を実行するときに

このスクリプトを使用します。このスクリプトは、**openstack overcloud deploy** コマンドを再実行するときに正確なオプションおよび環境ファイルの一貫性を保つのに役立ち、オーバークラウドの破損を回避するのに役立ちます。

- 環境ファイルを保持するディレクトリーのリビジョンを維持して、不要な変更を回避し、過去に行われた変更を追跡します。

6.13. 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、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。

6.14. 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) $ openstack container save 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。

6.15. 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。



注記

最初のオーバークラウドの作成に失敗した場合は、**openstack stack delete overcloud** コマンドを使用して部分的にデプロイされたオーバークラウドを削除し、再試行できます。これらの最初のオーバークラウドの作成が失敗した場合にのみ、このコマンドを実行してください。このコマンドは、完全にデプロイされて稼働中のオーバークラウドでは実行しないでください。そうしないと、オーバークラウド全体が削除されます。

6.16. オーバークラウドデプロイメント出力の表示

オーバークラウドのデプロイメントが正常に完了すると、シェルから以下の情報が返されます。この情報を使用してオーバークラウドにアクセスすることができます。

```
Overcloud configuration completed.
Overcloud Endpoint: http://192.168.24.113:5000
Overcloud Horizon Dashboard URL: http://192.168.24.113:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
Overcloud Deployed
```

6.17. 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ます。director は、このファイル **overcloudrc** を **stack** ユーザーのホームディレクトリに保存します。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
```

これにより、director 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
```

6.18. 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、カスタム機能は含まれません。ただし、[オーバークラウドの高度なカスタマイズ](#)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、仕様に合わせてカスタマイズすることができます。

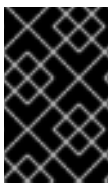
本章の例では、すべてのノードが電源管理に IPMI を使用したベアメタルシステムとなっています。これ以外のサポート対象電源管理ドライバーおよびそのオプションに関する詳細は、[付録B 電源管理ドライバー](#)を参照してください。

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、各ノード種別に設定された要件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。ただし、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。



重要

マルチアーキテクチャクラウドを有効にする場合には、UI ワークフローはサポートされません。[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](#)に記載の手順に従ってください。

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 (keystone)	UI への認証と他のサービスのエンドポイント検出
OpenStack Orchestration (heat)	デプロイメントのステータス
OpenStack Bare Metal (ironic)	ノードの制御
OpenStack Object Storage (swift)	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 (mistral)	director タスクのアクセスおよび実行
OpenStack Messaging (zaqar)	特定のタスクのステータスを検索する Websocket ベースのサービス

7.2. WEB UI のナビゲーション

UI は主に 3 つのセクションで設定されています。

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

The screenshot displays the OpenStack Director web interface. On the left, a vertical navigation pane shows four steps: 1. Prepare Hardware, 2. Specify Deployment Configuration, 3. Configure Roles and Assign Nodes, and 4. Deploy. Step 3 is currently selected. The main content area shows the configuration for step 3, titled '7 Nodes available to assign'. It contains five role cards: Block Storage (0 of 6 nodes assigned), Ceph Storage (0 of 6 nodes assigned), Compute (1 of 8 nodes assigned), Controller (1 of 7 nodes assigned), and Object Storage (0 of 6 nodes assigned). Each card has a dropdown menu to select the number of nodes. At the bottom, there is a 'Validate and Deploy' button.

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 ▾	Add filter	Name ▾	↓↑	Intropect Nodes	Provide Nodes	
9 Nodes Select All						
> <input type="checkbox"/>		node01	Off	Intropection: finished	Provision State: available	Profile: compute 1 CPU Core 4096 MB RAM 49 GB Disk
> <input type="checkbox"/>		node02	Off	Intropection: finished	Provision State: available	Profile: compute 1 CPU Core 4096 MB RAM 49 GB Disk
> <input type="checkbox"/>		node03	Off	Intropection: finished	Provision State: available	Profile: control 2 CPU Cores 10240 MB RAM 99 GB Disk
> <input type="checkbox"/>		node04	Off	Intropection: finished	Provision State: available	Profile: - 2 CPU Cores 10240 MB RAM 99 GB Disk
> <input type="checkbox"/>		node05	Off	Intropection: 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

×

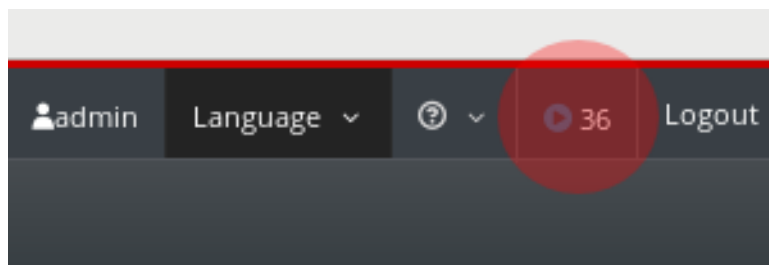
/dev/sda
 Root Device
 Type: **HDD** Size: **299.44 GB**

Model: PERC H330 Mini
Serial: 61866da04f37e8001ea4e109127d48f0
Vendor: DELL
WWN: 0x61866da04f37e800
WWN Vendor Extension: 0x1ea4e109127d48f0
WWN with Extension: 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。ただし、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 ▾ Add filter


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 [オーバークラウドの高度なカスタマイズ](#) 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 4 つの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**

をクリックします。これにより、現在の **プラン** の一覧が表示されます。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:** プランのプレーンテキスト名。たとえば、**overcloud**。
- **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](#)

Plan Name	Status
my-other-overcloud	Not deployed
overcloud	Not deployed

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。

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 (**ipmi**)、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、IPMI のパスワード

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 単位)

ディスク (GB)

(オプション) ハードディスクのサイズ (GB 単位)

NIC の MAC アドレス

ノード上のネットワークインターフェイスの 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ブートします。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 を使用したロ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。ベアメタルノードの場合には、通常 15 分ほどかかります。

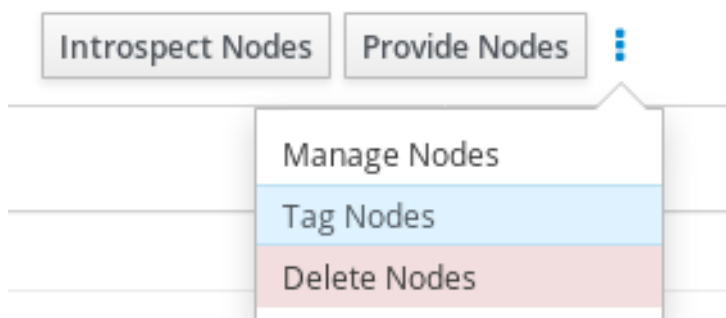
イントロスペクションのプロセスを完了したら、**プロビジョニングの状態** が **manageable** に設定されているノードをすべて選択して、**プロビジョン** ボタンをクリックします。**プロビジョニングの状態** が **available** になるまで待ちます。

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

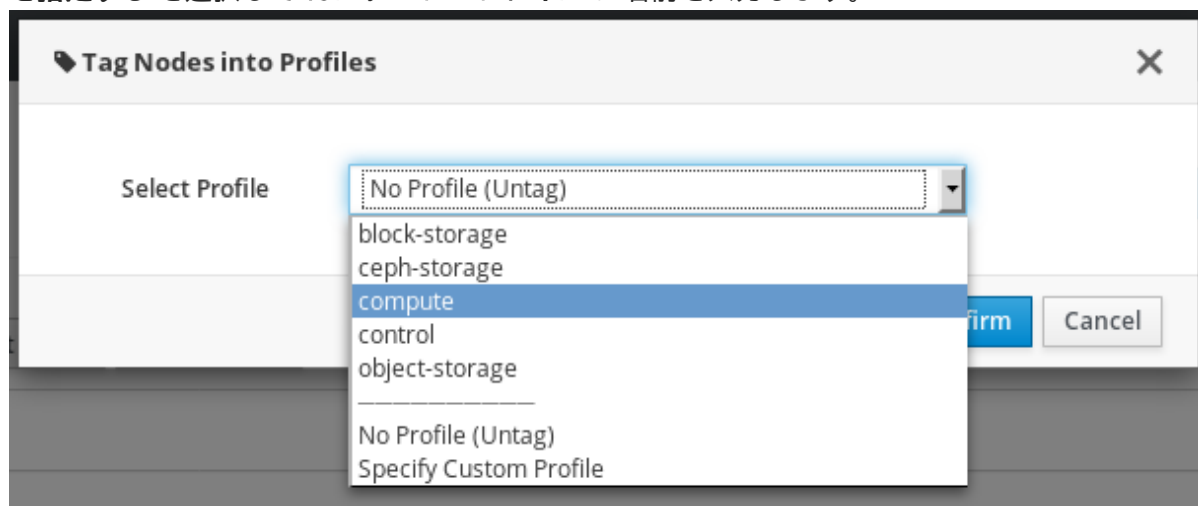
各ノードにプロファイルのセットを割り当てることができます。それぞれのプロファイルが、個々のフレーバーおよびロールに対応します (詳細は、「[プロファイルへのノードのタグ付け](#)」を参照してください)。

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、機能毎に異なる環境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
0
Number of BlockStorage nodes to deploy

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

BlockStorageHostnameFormat
%stackname%-blockstorage-%index%
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']}]}

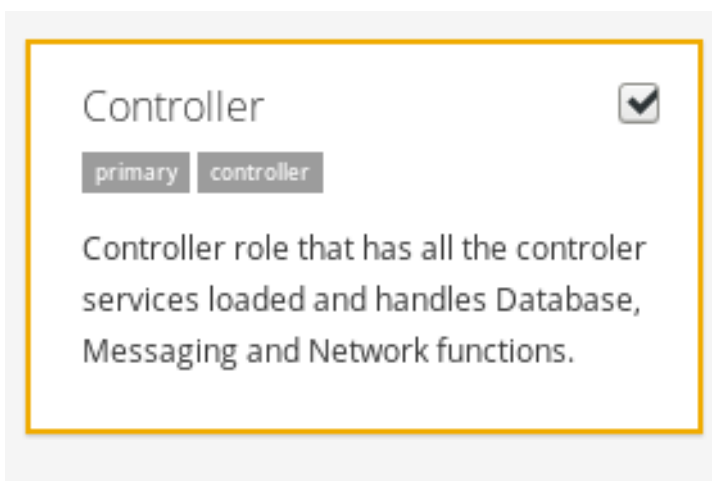
BlockStorageSchedulerHints
{}
Optional scheduler hints to pass to nova

7.8. WEB UI でのロールの追加

3 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。

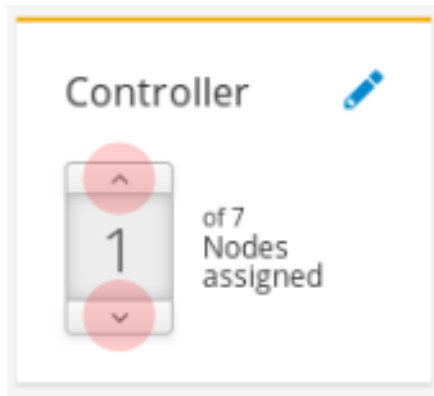


ロールを選択したら、**変更の保存** をクリックします。

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。

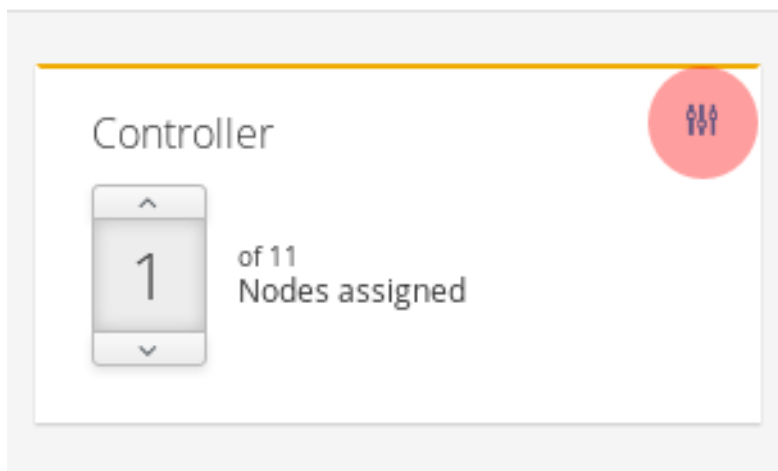
ロールにノードを割り当てるには、**プラン** の画面で **3 ロールの設定とノードの割り当て** セクションまでスクロールします。各ロールで、スピナーウィジェットを使ってノード数をロールに割り当てます。それぞれのロールで利用可能なノードの数は、[「Web UI を使用したプロファイルへのノードのタグ付け」](#) でタグ付けしたノードに基づきます。



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。詳しくは、[「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

Parameters Services Network Configuration

AodhApi

AodhEvaluator

AodhListener

AodhNotifier

CACerts

CeilometerAgentCentral

CeilometerAgentNotification

CeilometerApi

CeilometerCollector

CeilometerExpirer

CinderApi

CinderScheduler

CinderVolume

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

256

Maximum number of simultaneously processed requests.

ApacheServerLimit

256

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

Parameters
Services
Network Configuration

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

ControlPlaneIp	<input type="text"/>
ExternalIpSubnet	<input type="text"/> IP address/subnet on the external network
InternalApiIpSubnet	<input type="text"/> IP address/subnet on the internal_api network
ManagementIpSubnet	<input type="text"/> IP address/subnet on the management network
StorageIpSubnet	<input type="text"/> IP address/subnet on the storage network
StorageMgmtIpSubnet	<input type="text"/> IP address/subnet on the storage_mgmt network
TenantIpSubnet	<input type="text"/>



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、[オーバークラウドの高度なカスタマイズ](#)の **コンポーザブルロール** に関する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。そのためには、**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。

 **Validate and Deploy**

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?

Deploy

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
ObjectStorageIpListMap	INIT_COMPLETE	2016-11-24T07:00:08Z
ControllerIpListMap	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

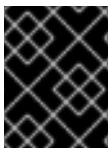
これで director の UI を使用したオーバークラウドの作成が完了しました。作成後の機能については、[9 章 オーバークラウド作成後のタスクの実行](#)を参照してください。

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 では、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す。

このシナリオでは、カスタム機能を持たない基本的な設定のみを説明します。ただし、[オーバークラウドの高度なカスタマイズ](#)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、仕様に合わせてカスタマイズすることができます。



重要

事前に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、各ノード種別に設定された要件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、ホストオペレーティングシステムとして Red Hat Enterprise Linux 7.5 以降をインストールする必要があります。Red Hat では、利用可能な最新バージョンの使用を推奨します。
- 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
Controller 0	192.168.24.2
Compute 0	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も、OpenStack サービス用のコントロールプレーンネットワークを使用します。ただし、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。
- いずれかのノードで Pacemaker リソースが使用される場合、サービスユーザー **hacluster** およびサービスグループ **haclient** の UID/GID は、**189** でなければなりません。これは [CVE-2018-16877](#) に対応するためです。オペレーティングシステムと共に Pacemaker をインストールした場合、インストールによりこれらの ID が自動的に作成されます。ID の値が正しく設定されていない場合は、アークティクル [OpenStack minor update / fast-forward upgrade can fail on the controller nodes at pacemaker step with "Could not evaluate: backup_cib"](#) の手順に従って ID の値を変更します。
- 一部のサービスが誤った IP アドレスにバインドされてデプロイメントに失敗するのを防ぐために、**/etc/hosts** ファイルに **node-name=127.0.0.1** のマッピングが含まれないようにします。

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。

- 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-0 ~]# useradd stack
[root@controller-0 ~]# passwd stack # specify a password
```

- 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。

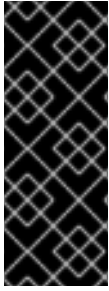
```
[root@controller-0 ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/stack
[root@controller-0 ~]# chmod 0440 /etc/sudoers.d/stack
```

- 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

それぞれのノードには、Red Hat サブスクリプションへのアクセスが必要です。



重要

スタンドアロンの Ceph ノードは例外で、Red Hat OpenStack Platform サブスクリプションは必要ありません。スタンドアロンの Ceph ノードの場合には、director に最新の ansible パッケージをインストールする必要があります。Red Hat OpenStack Platform に対応したデプロイメントツールを取得するには、アクティブな Red Hat OpenStack Platform サブスクリプションがないすべての Ceph ノードで **rhel-7-server-openstack-13-deployment-tools-rpms** リポジトリを有効にすることが必須要件です。

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-0 ~]# 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-0 ~]# 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-0 ~]# 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-0 ~]# sudo subscription-manager repos --disable=*
```

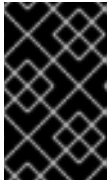
5. 必要な Red Hat Enterprise Linux リポジトリを有効にします。

- 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-0 ~]# 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-3-osd-rpms --enable=rhel-7-server-rhceph-3-mon-rpms --enable=rhel-7-server-rhceph-3-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-0 ~]# 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-0 ~]# sudo yum update -y
[root@controller-0 ~]# 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-0 ~]# 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-0 ~]# 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。オーバークラウドノードがパブリック 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。[オーバークラウドの高度なカスタマイズ](#)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、[オーバークラウドの高度なカスタマイズ](#)の以下のセクションを参照してください。

- [基本的なネットワーク分離](#)
- [ノード配置の制御](#)



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 が再設定され、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-0-ctlplane:
      fixed_ips:
        - ip_address: 192.168.24.2
      subnets:
```



```
- cidr: 24
compute-0-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。形式は **<node_hostname>-<network>** です。ここで、**<node_hostname>** の値はノードの短いホスト名で、**<network>** はネットワークの小文字を使った名前です。たとえば、**controller-0.example.com** であれば **controller-0-ctlplane** となり、**compute-0.example.com** の場合は **compute-0-ctlplane** となります。
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。ただし、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 と通信することができません。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 と通信するように設定しなければならない場合があります。

このシナリオには、いくつかの要件があります。

- 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
Controller 0	192.168.100.2
Compute 0	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す。

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 undercloud 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](#)」に記載の手順と類似しています。ただし、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 アドレスを含め、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。このネットワークアーキテクチャに対応するように、「[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を以下に示します。

```
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 ❶

parameter_defaults:
  NeutronPublicInterface: eth1
  EC2MetadataIp: 192.168.100.1 ❷
  ControlPlaneDefaultRoute: 192.168.100.1
  DeployedServerPortMap:
    control_virtual_ip:
      fixed_ips:
        - ip_address: 192.168.100.1
      subnets:
        - cidr: 24
    controller-0-ctlplane:
      fixed_ips:
        - ip_address: 192.168.100.2
      subnets:
        - cidr: 24
    compute-0-ctlplane:
      fixed_ips:
        - ip_address: 192.168.100.3
      subnets:
        - cidr: 24
```

- ❶ **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。
- ❷ **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向けの CEPH STORAGE の設定

ceph-ansible およびデプロイ済みのサーバーを使用する場合、デプロイメントの前にアンダークラウドから以下のようなコマンドを実行する必要があります。

```
export OVERCLOUD_HOSTS="192.168.1.8 192.168.1.42"
```

```
bash /usr/share/openstack-tripleo-heat-templates/deployed-server/scripts/enable-ssh-admin.sh
```

例に示す **export** コマンドを使用して、**OVERCLOUD_HOSTS** 変数を Ceph クライアントとして使用するオーバークラウドホストの IP アドレスに設定します (Compute、Block Storage、Image、File System、Telemetry サービスなど)。**enable-ssh-admin.sh** スクリプトにより、Ansible が Ceph クライアントの設定に使用するオーバークラウドノードのユーザーが設定されます。

8.8. 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

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](#)」に記載された標準の 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/deployed-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/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。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/deployed-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、**deployed-server/deployed-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。
- **deployed-server/deployed-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
  ...
```

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を、以下に示します。

```
$ 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
```

このコマンドにより、オーバークラウドの設定を開始されます。ただし、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。

8.9. 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ーを実行するには、**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相関します。

[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ーを実行します。
- 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続行されます。

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、各ノードのメタデータ 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=2147483586

== 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=2147483586
```

■

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ます。これにより、手動の変更はエージェントにより上書きされなくなります。

```
$ 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 を使用するように設定します。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_6fce4e6019264a5b8283e7125f05b764/ov-
edServer-ts6lr4tm5p44-deployed-server-td42md2tap4g/43d302fa-d4c2-40df-b3ac-
624d6075ef27?
temp_url_sig=58313e577a93de8f8d2367f8ce92dd7be7aac3a1&temp_url_expires=214748358
6
```

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。

8.10. 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。

8.11. 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ます。director は、このファイル **overcloudrc** を **stack** ユーザーのホームディレクトリーに保存します。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
```

これにより、director 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) $
```

8.12. 事前にプロビジョニングされたノードのスケーリング

事前にプロビジョニングされたノードをスケーリングするプロセスは、[12章 オーバークラウドノードのスケーリング](#)に記載する標準のスケーリング手順と類似しています。ただし、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。

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の概略は、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新たに準備します。
2. ノードをスケールアップします。手順については[12章 オーバークラウドノードのスケーリング](#)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手動で設定します。

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を持つオーバークラウドをスケールダウンするには、[12章 オーバークラウドノードのスケーリング](#)に記載する通常のスケールダウン手順に従います。

ほとんどのスケーリング操作では、ノードの UUID 値を取得して **openstack overcloud node delete** に渡す必要があります。この UUID を取得するには、ロールを指定してリソースの一覧を表示します。

```
$ openstack stack resource list overcloud -c physical_resource_id -c stack_name -n5 --filter
type=OS::TripleO::<RoleName>Server
```

上記コマンドの **<RoleName>** を、スケールダウンする実際のロール名に置き換えます。**ComputeDeployedServer** ロールの例を以下に示します。

```
$ openstack stack resource list overcloud -c physical_resource_id -c stack_name -n5 --filter
type=OS::TripleO::ComputeDeployedServerServer
```

コマンド出力の **stack_name** 列から、各ノードに関連付けられた UUID を確認します。**stack_name** には、Heat リソースグループ内のノードインデックスの整数値が含まれます。出力の例を以下に示します。

```

+-----+-----+
| physical_resource_id | stack_name |
+-----+-----+
| 294d4e4d-66a6-4e4e-9a8b- | overcloud-ComputeDeployedServer- |
| 03ec80beda41 | no7yfgnh3z7e-1-ytfqdeclwvcg |
| d8de016d- | overcloud-ComputeDeployedServer- |
| 8ff9-4f29-bc63-21884619abe5 | no7yfgnh3z7e-0-p4vb3meacxwn |
| 8c59f7b1-2675-42a9-ae2c- | overcloud-ComputeDeployedServer- |
| 2de4a066f2a9 | no7yfgnh3z7e-2-mmmaayxqnf3o |
+-----+-----+

```

stack_name 列のインデックス 0、1、または 2 は、Heat リソースグループ内のノード順に対応します。**physical_resource_id** 列の該当する UUID 値を、**openstack overcloud node delete** コマンドに渡します。

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。ただし、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する必要があります。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するまでは、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。

8.13. 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。詳しくは、[「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するまでは、再利用しないようにしてください。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。

8.14. オーバークラウド作成の完了

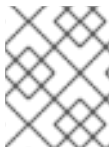
これで、事前にプロビジョニングされたノードを使用したオーバークラウドの作成が完了しました。作成後の機能については、[9章 オーバークラウド作成後のタスクの実行](#)を参照してください。

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。本項では、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、**Getting Started with Containers**の [Working with Docker formatted containers](#) 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
```

コンテナの属性の確認

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す。

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イ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。オーバークラウドでは、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、デプロイ中に追加のブリッジをマッピングしている限り、**br-ex** だけでなく、任意のブリッジを使用できます。

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イスをアタッチされているネットワークインターフェイ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. 基本的なオーバークラウドフレーバーの作成

本ガイドの検証ステップは、インストール環境にフレーバーが含まれていることを前提としています。まだ1つのフレーバーも作成していない場合には、以下のコマンドを使用してさまざまなストレージおよび処理能力に対応する基本的なデフォルトフレーバーセットを作成してください。

```
$ openstack flavor create m1.tiny --ram 512 --disk 0 --vcpus 1
$ openstack flavor create m1.smaller --ram 1024 --disk 0 --vcpus 1
$ openstack flavor create m1.small --ram 2048 --disk 10 --vcpus 1
$ openstack flavor create m1.medium --ram 3072 --disk 10 --vcpus 2
$ openstack flavor create m1.large --ram 8192 --disk 10 --vcpus 4
$ openstack flavor create m1.xlarge --ram 8192 --disk 10 --vcpus 8
```

コマンドオプション

ram

フレーバーの最大 RAM を定義するには、**ram** オプションを使用します。

disk

フレーバーのハードディスク容量を定義するには、**disk** オプションを使用します。

vcpus

フレーバーの仮想 CPU 数を定義するには、**vcpus** オプションを使用します。

openstack flavor create コマンドについての詳しい情報は、**\$ openstack flavor create --help** で確認してください。

9.7. 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 Internal 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
```

9.8. 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。



重要

カスタム環境ファイルからパラメーターを削除しても、パラメーター値はデフォルト設定に戻りません。**/usr/share/openstack-tripleo-heat-templates** のコア Heat テンプレートコレクションからデフォルト値を特定し、カスタム環境ファイルでその値を手動で設定する必要があります。

新しい環境ファイルを追加する場合は、**-e** オプションを使用して **openstack overcloud deploy** コマンドに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。

9.9. 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
```

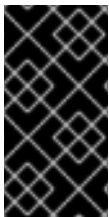
--list オプションを指定すると、全ホストの詳細が表示されます。これにより、動的インベントリーが JSON 形式で出力されます。

```
{"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": "abcdefghijklmnopqrstuvwxyz12345678", "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**
 - 全コンピュートノードの場合には **Compute**
 - コントローラー および コンピュート など、オーバークラウドの全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。

9.10. 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。

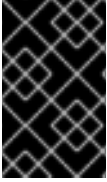
実行中のサーバーのスナップショットを作成して新規イメージを作成し、そのイメージをダウンロードします。

```
$ 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。

9.11. オーバークラウドの削除防止

Heat に含まれるコードベースのデフォルトポリシーセットは、`/etc/heat/policy.json` を作成してカスタムルールを追加することでオーバーライドすることができます。全員 のオーバークラウド削除権限を無効にするには、以下のポリシーを追加します。

```
{"stacks:delete": "rule:deny_everybody"}
```

これにより **heat** クライアントによるオーバークラウドの削除が阻止されます。オーバークラウドを削除できるように設定するには、カスタムポリシーを削除して `/etc/heat/policy.json` を保存します。

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
```

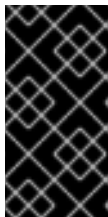
オーバークラウドが削除されていることを確認します。

```
(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。

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、テスト用途にのみご利用いただく機能で、実稼働環境にデプロイすべきではありません。テクノロジープレビュー機能についての詳しい情報は、[対象範囲の詳細](#) を参照してください。

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で役立ちます。Ansible の使用方法については、[Ansible のドキュメント](#) 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。



注記

この設定方法では、どのようなノードへの Ceph Storage クラスターのデプロイもサポートされません。

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. source コマンドで **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。

- 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

- 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-novacompute-0: compute-00-rack01
overcloud-novacompute-1: compute-01-rack01
overcloud-novacompute-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 の実行を処理します。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 ワークフロー実行の 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** ユーザーのアクセス権限をリフレッシュします。

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

このコマンドを実行すると、再度ログインが求められます。**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** を実行した Mistral の実行に対応する UUID です。

手順

1. **openstack workflow execution list** コマンドですべての実行を一覧表示して、選択した Mistral の実行 (**config-download** を実行したもの) 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 の実行の 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. 手動での CONFIG-DOWNLOAD の実行

/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 の実行の 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、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。

5. 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されます。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、タグ付けし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** の全タスクを順番どお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

- source コマンドで **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

- 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 仮想コントロールプレーンの作成

仮想コントロールプレーンは、ベアメタルではなく仮想マシン (VM) 上にあるコントロールプレーンです。仮想コントロールプレーンを使用することで、コントロールプレーンに必要なベアメタルマシンの数を減らすことができます。

本章では、Red Hat OpenStack Platform (RHOSP) および Red Hat Virtualization を使用して、オーバークラウドの RHOSP コントロールプレーンを仮想化する方法について説明します。

11.1. 仮想コントロールプレーンのアーキテクチャー

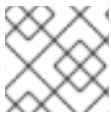
Red Hat OpenStack Platform (RHOSP) director を使用して、Red Hat Virtualization クラスターにデプロイされたコントローラーノードを使用するオーバークラウドをプロビジョニングします。その後、これらの仮想コントローラーを仮想コントロールプレーンノードとしてデプロイできます。



注記

仮想コントローラーノードは、Red Hat Virtualization 上でのみサポートされます。

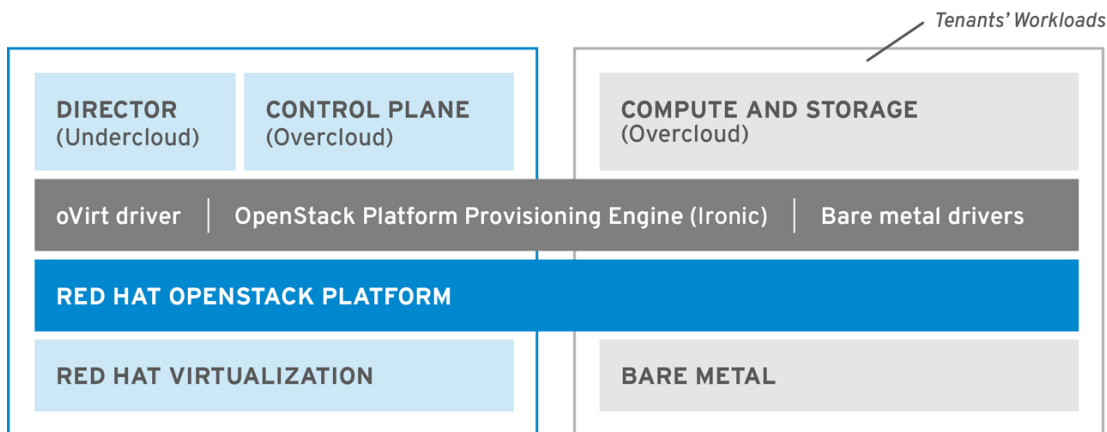
以下のアーキテクチャー図は、仮想コントロールプレーンのデプロイ方法を示しています。オーバークラウドは、Red Hat Virtualization 上の仮想マシンで実行中のコントローラーノードに分散されます。コンピュートノードおよびストレージノードは、ベアメタル上で実行されます。



注記

OpenStack 仮想アンダークラウドは、Red Hat Virtualization 上で実行されます。

仮想コントロールプレーンのアーキテクチャー



OPENSTACK_477985_1018

OpenStack Bare Metal Provisioning (ironic) サービスには、Red Hat Virtualization の仮想マシン用ドライバー **staging-ovirt** が含まれています。このドライバーを使用して、Red Hat Virtualization 環境内の仮想ノードを管理できます。このドライバーを使用して、Red Hat Virtualization 環境内の仮想マシンとしてオーバークラウドのコントローラーをデプロイすることもできます。

11.2. RHOSP オーバークラウドのコントロールプレーンを仮想化する際の利点と制限

RHOSP オーバークラウドのコントロールプレーンを仮想化する利点は数多くありますが、すべての設定に適用できる訳ではありません。

利点

オーバークラウドのコントロールプレーンを仮想化することには、ダウンタイムを回避し、パフォーマンスを向上させる数多くの利点があります。

- ホットプラグおよびホットアンプラグを使用して CPU およびメモリーを必要に応じてスケーリングし、リソースを仮想コントローラーに動的に割り当てることができます。これにより、ダウンタイムを防ぎ、プラットフォームの拡張に合わせて増大した能力を活用できます。
- 同じ Red Hat Virtualization クラスターに追加のインフラストラクチャー仮想マシンをデプロイすることができます。これにより、データセンターのサーバーフットプリントが最小限に抑えられ、物理ノードの効率が最大化されます。
- コンポーザブルロールを使用して、より複雑な RHOSP コントロールプレーンを定義することができます。したがって、リソースをコントロールプレーンの特定のコンポーネントに割り当てることができます。
- 仮想マシンのライブマイグレーション機能を使用すると、サービスを中断せずにシステムをメンテナンスすることができます。
- Red Hat Virtualization がサポートするサードパーティーまたはカスタムツールを統合することができます。

制限

仮想コントロールプレーンには、使用できる設定の種類に制限があります。

- 仮想 Ceph Storage ノードおよびコンピュートノードはサポートされません。
- ファイバーチャネルを使用するバックエンドについては、Block Storage (cinder) のイメージからボリュームへの転送はサポートされません。Red Hat Virtualization は N_Port ID Virtualization (NPIV) をサポートしていません。したがって、ストレージのバックエンドからコントローラー (デフォルトで **cinder-volume** を実行) に LUN をマッピングする必要のある Block Storage (cinder) ドライバーは機能しません。**cinder-volume** のロールは、仮想コントローラーに含めるのではなく、専用のロールとして作成する必要があります。詳しい情報は、オーバークラウドの高度なカスタマイズの [コンポーザブルサービスとカスタムロール](#) を参照してください。

11.3. RED HAT VIRTUALIZATION ドライバーを使用した仮想コントローラーのプロビジョニング

本項では、RHOSP および Red Hat Virtualization を使用して、オーバークラウドの仮想 RHOSP コントロールプレーンをプロビジョニングする方法について説明します。

前提条件

- Intel 64 または AMD64 CPU 拡張機能をサポートする、64 ビット x86 プロセッサが必要です。
- 以下のソフトウェアがすでにインストールされ、設定されている必要があります。
 - Red Hat Virtualization。詳しい情報は、[Red Hat Virtualization ドキュメントスイート](#) を参照してください。

- Red Hat OpenStack Platform (RHOSP)。詳しい情報は、[director のインストールと使用方法](#) を参照してください。
- 事前に仮想コントローラーノードを準備しておく必要があります。これらの要件は、ベアメタルのコントローラーノードの要件と同じです。詳しい情報は、[コントローラーノードの要件](#) を参照してください。
- オーバークラウドのコンピュートノードおよびストレージノードとして使用するベアメタルノードを、事前に準備しておく必要があります。ハードウェアの仕様については、[コンピュートノードの要件](#) および [Ceph Storage ノードの要件](#) を参照してください。POWER (ppc64le) ハードウェアにオーバークラウドのコンピュートノードをデプロイするには、[Red Hat OpenStack Platform for POWER](#) を参照してください。
- 論理ネットワークが作成され、クラスターまたはホストネットワークで複数ネットワークによるネットワーク分離を使用する用意ができています。詳しい情報は、Red Hat Virtualization Administration Guide の [Logical Networks](#) を参照してください。
- 各ノードの内部 BIOS クロックが UTC に設定されている必要があります。タイムゾーンオフセットを適用する前に hwclock が BIOS クロックを同期すると、ファイルのタイムスタンプに未来の日時が設定されていました。この設定により、この問題を防ぐことができます。

ヒント

パフォーマンスのボトルネックを防ぐために、コンポーザブルロールを使用しデータプレーンサービスをベアメタルのコントローラーノード上に維持します。

手順

1. **undercloud.conf** 設定ファイルの **enabled_hardware_types** に **staging-ovirt** ドライバーを追加して、director アンダークラウドでこのドライバーを有効にします。

```
enabled_hardware_types = ipmi,redfish,ilo,idrac,staging-ovirt
```

2. アンダークラウドに **staging-ovirt** ドライバーが含まれることを確認します。

```
(undercloud) [stack@undercloud ~]$ openstack baremetal driver list
```

アンダークラウドが正しく設定されていると、コマンドにより以下の結果が返されます。

```
+-----+-----+
| Supported driver(s) | Active host(s) |
+-----+-----+
| idrac              | localhost.localdomain |
| ilo                | localhost.localdomain |
| ipmi               | localhost.localdomain |
| pxe_drac           | localhost.localdomain |
| pxe_ilo            | localhost.localdomain |
| pxe_ipmitool       | localhost.localdomain |
| redfish            | localhost.localdomain |
| staging-ovirt      | localhost.localdomain |
```

3. **python-ovirt-engine-sdk4.x86_64** パッケージをインストールします。

```
$ sudo yum install python-ovirt-engine-sdk4
```

4. オーバークラウドノードの定義テンプレート (例: **nodes.json**) を更新し、Red Hat Virtualization がホストする仮想マシンを director に登録します。詳しい情報は、[オーバークラウドノードの登録](#) を参照してください。以下のキー/値のペアを使用して、オーバークラウドでデプロイする仮想マシンの特性を定義します。

表11.1 オーバークラウド用仮想マシンの設定

キー	この値に設定します
pm_type	oVirt/RHV 仮想マシン用の OpenStack Bare Metal Provisioning (ironic) サービスドライバー staging-ovirt
pm_user	Red Hat Virtualization Manager のユーザー名
pm_password	Red Hat Virtualization Manager のパスワード
pm_addr	Red Hat Virtualization Manager サーバーのホスト名または IP
pm_vm_name	コントローラーが作成される Red Hat Virtualization Manager の仮想マシンの名前

以下に例を示します。

```
{
  "nodes": [
    {
      "name": "osp13-controller-0",
      "pm_type": "staging-ovirt",
      "mac": [
        "00:1a:4a:16:01:56"
      ],
      "cpu": "2",
      "memory": "4096",
      "disk": "40",
      "arch": "x86_64",
      "pm_user": "admin@internal",
      "pm_password": "password",
      "pm_addr": "rhvm.example.com",
      "pm_vm_name": "{vernum}-controller-0",
      "capabilities": "profile:control,boot_option:local"
    },
  ],
}
```

Red Hat Virtualization Host ごとに1つのコントローラーを設定します。

5. Red Hat Virtualization でアフィニティーグループをソフトネガティブアフィニティーに設定し、コントローラー用仮想マシンの高可用性を確保します。詳しい情報は、Red Hat Virtualization Virtual Machine Management Guide の [Affinity Groups](#) を参照してください。

6. Red Hat Virtualization Manager のインターフェイスにアクセスし、これを使用してそれぞれの VLAN をコントローラー用仮想マシンの個別の論理仮想 NIC にマッピングします。詳しい情報は、Red Hat Virtualization Administration Guide の [Logical Networks](#) を参照してください。
7. director とコントローラー用仮想マシンの仮想 NIC で **no_filter** を設定し、仮想マシンを再起動します。これにより、コントローラー用仮想マシンにアタッチされたネットワークで MAC スプーフィングフィルターが無効化されます。詳しい情報は、Red Hat Virtualization Administration Guide の [Virtual Network Interface Cards](#) を参照してください。
8. オーバークラウドをデプロイして、新しい仮想コントローラーノードを環境に追加します。

```
(undercloud) [stack@undercloud ~]$ openstack overcloud deploy --templates
```

第12章 オーバークラウドノードのスケーリング

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場合は、オーバークラウドを更新する必要があります。



警告

オーバークラウドからノードを削除する場合は、**openstack server delete** を使用しないでください。本項で説明する手順をよく読み、適切にノードの削除/置き換えを行ってください。



注記

オーバークラウドノードのスケールアウトまたは削除を開始する前に、ベアメタルノードがメンテナンスモードに設定されていないことを確認してください。

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12.1 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	説明
コントローラー	非対応	非対応	13章コントローラーノードの置き換え に記載の手順を使用して、コントローラーノードを置き換えることができます。
Compute	対応	対応	
Ceph Storage ノード	対応	非対応	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。
オブジェクトストレージノード	対応	対応	



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。

12.1. オーバークラウドへのノード追加

director のノードプールにさらにノードを追加するには、以下の手順を実施します。

手順

1. 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 (**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":"ipmi",
      "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":"ipmi",
      "pm_user":"admin",
      "pm_password":"p@55w0rd!",
      "pm_addr":"192.168.24.208"
    }
  ]
}
```

2. 以下のコマンドを実行して、新規ノードを登録します。

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

3. 新しいノードを登録したら、次のコマンドを実行してノードを一覧表示し、新しいノードの UUID を特定します。

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

4. 次のコマンドを実行して、新しいノードごとにイントロスペクションプロセスを起動します。

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

このプロセスにより、ノードのハードウェア属性の検出とベンチマークが実行されます。

5. ノードのイメージプロパティを設定します。

```
(undercloud) $ openstack overcloud node configure [NODE UUID]
```

12.2. ロールのノード数の追加

特定ロールのオーバークラウドノード (たとえばコンピュートノード) をスケーリングするには、以下の手順を実施します。

手順

1. それぞれの新規ノードを希望するロールにタグ付けします。たとえば、ノードをコンピュートロールにタグ付けするには、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' [NODE UUID]
```

2. オーバークラウドをスケーリングするには、ノード数が含まれる環境ファイルを編集してオーバークラウドを再デプロイする必要があります。たとえば、オーバークラウドをコンピュートノード 5 台にスケーリングするには、**ComputeCount** パラメーターを編集します。

```
parameter_defaults:
...
ComputeCount: 5
...
```

3. 更新したファイルを使用して、デプロイメントコマンドを再度実行します。以下の例では、このファイルは **node-info.yaml** という名前です。

```
(undercloud) $ openstack overcloud deploy --templates -e /home/stack/templates/node-
info.yaml [OTHER_OPTIONS]
```

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。これには、コンピュート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が含まれます。

4. デプロイメント操作が完了するまで待ちます。

12.3. コンピュートノードの削除または交換

状況によっては、オーバークラウドからコンピュートノードを削除する必要があります。たとえば、問題のあるコンピュートノードを置き換える必要がある場合などです。コンピュートノードを削除すると、スケールアウト操作中にインデックスが再利用されないように、ノードのインデックスがデフォルトで拒否リストに追加されます。

オーバークラウドデプロイメントからノードを削除した後、削除されたコンピュートノードを置き換えることができます。

前提条件

- 削除するノードでは、ノードが新しいインスタンスをスケジュールできないようにするために、コンピュートサービスが無効になっています。コンピュートサービスが無効になっていることを確認するには、次のコマンドを使用します。


```
(overcloud)$ openstack compute service list
```

コンピュートサービスが無効になっていない場合は、無効にします。

```
(overcloud)$ openstack compute service set <hostname> nova-compute --disable
```

ヒント

--disable-reason オプションを使用して、サービスを無効にする理由についての簡単な説明を追加します。これは、コンピュートサービスを再デプロイする場合に役立ちます。

- コンピュートノードのワークロードは、他のコンピュートノードに移行されました。詳しくは、[コンピュートノード間の仮想マシンインスタンスの移行](#) を参照してください。
- インスタンス HA が有効になっている場合は、次のいずれかのオプションを選択します。
 - コンピュートノードにアクセスできる場合は、**root** ユーザーとしてコンピュートノードにログインし、**shutdown -h now** コマンドを使用してクリーンシャットダウンを実行します。
 - コンピュートノードにアクセスできない場合は、**root** ユーザーとしてコントローラーノードにログインし、コンピュートノードの STONITH デバイスを無効にして、ベアメタルノードをシャットダウンします。

```
[root@controller-0 ~]# pcs stonith disable <stonith_resource_name>
[stack@undercloud ~]$ source stackrc
[stack@undercloud ~]$ openstack baremetal node power off <UUID>
```

手順

1. source コマンドでアンダークラウド設定を読み込みます。

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

2. オーバークラウドスタックの UUID を特定します。

```
(undercloud)$ openstack stack list
```

3. 削除するコンピュートノードの UUID またはホスト名を特定します。

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

4. (オプション) **--update-plan-only** オプションを指定して **overcloud deploy** コマンドを実行し、テンプレートからの最新の設定でプランを更新します。これにより、コンピュートノードを削除する前に、オーバークラウドの設定が最新の状態になります。

```
$ openstack overcloud deploy --update-plan-only \
  --templates \
  -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 \
-e /home/stack/templates/rhel-registration/environment-rhel-registration.yaml \
[-e |...]
```



注記

オーバークラウドノードの拒否リストを更新した場合は、この手順が必要です。オーバークラウドノードを拒否リストに追加する方法の詳細については、[Blacklisting nodes](#) を参照してください。

5. スタックからコンピューットノードを削除します。

```
$ openstack overcloud node delete --stack <overcloud> \
<node_1> ... [node_n]
```

- **<overcloud>** は、オーバークラウドスタックの名前または UUID に置き換えてください。
- **<node_1>** (およびオプションとして **[node_n]** までのすべてのノード) を、削除するコンピューットノードのコンピューサービスホスト名または UUID に置き換えます。UUID とホスト名を混在させて使用しないでください。UUID だけ、またはホスト名だけを使用します。



注記

ノードの電源がすでにオフになっている場合、このコマンドは **WARNING** メッセージを返します。

```
Ansible failed, check log at /var/lib/mistral/overcloud/ansible.log
WARNING: Scale-down configuration error. Manual cleanup of some
actions may be necessary. Continuing with node removal.
```

このメッセージは無視しても問題ありません。

6. コンピューットノードが削除されるのを待ちます。
7. ノードの削除が完了したら、オーバークラウドスタックのステータスを確認します。

```
(undercloud)$ openstack stack list
```

表12.2 結果

ステータス	説明
UPDATE_COMPLETE	削除操作は正常に完了しました。

ステータス	説明
UPDATE_FAILED	<p>削除操作が失敗しました。</p> <p>削除操作が失敗する一般的な理由は、削除するノードの IPMI インターフェイスに到達できないことです。</p> <p>削除操作が失敗した場合は、コンピュートノードを手動で削除する必要があります。詳細は、コンピュートノードを手動で削除する を参照してください。</p>

8. インスタンス HA が有効な場合は、以下の操作を実行します。

- a. ノードの Pacemaker リソースをクリーンアップします。

```
$ sudo pcs resource delete <scaled_down_node>
$ sudo cibadmin -o nodes --delete --xml-text '<node id="<scaled_down_node>"/>'
$ sudo cibadmin -o fencing-topology --delete --xml-text '<fencing-level target="
<scaled_down_node>"/>'
$ sudo cibadmin -o status --delete --xml-text '<node_state id="<scaled_down_node>"/>'
$ sudo cibadmin -o status --delete-all --xml-text '<node id="<scaled_down_node>"/>' --
force
```

- b. ノードの STONITH デバイスを削除します。

```
$ sudo pcs stonith delete <device-name>
```

9. オーバークラウドで削除されたコンピュートノードを置き換えない場合は、ノード数を含む環境ファイルの **ComputeCount** パラメーターを減らします。このファイルは、通常 **node-info.yaml** という名前です。たとえば、ノードを1つ削除する場合、ノード数を4から3に減らします。

```
parameter_defaults:
...
ComputeCount: 3
```

ノード数を減らすと、**openstack overcloud deploy** を実行しても director は新規ノードをプロビジョニングしません。

オーバークラウドデプロイメントで削除されたコンピュートノードを置き換える場合は、[Replacing a removed Compute node](#) を参照してください。

12.3.1. コンピュートノードを手動で削除する

openstack overcloud node delete コマンドが到達不能なノードのために失敗した場合は、オーバークラウドからのコンピュートノードの削除を手動で完了する必要があります。

前提条件

- [コンピュートノードの削除または置換](#) 手順を実行すると、ステータス **UPDATE_FAILED** が返されました。

手順

1. オーバークラウドスタックの UUID を特定します。

```
(undercloud)$ openstack stack list
```

2. 手動で削除するノードの UUID を特定します。

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

3. 削除するノードをメンテナンスモードに移動します。

```
(undercloud)$ openstack baremetal node maintenance set <node_uuid>
```

4. コンピュートサービスがその状態をベアメタルサービスと同期するのを待ちます。これには最大 4 分かかる場合があります。

5. source コマンドでオーバークラウド設定を読み込みます。

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

6. 削除したノードのネットワークエージェントを削除します。

```
(overcloud)$ for AGENT in $(openstack network agent list --host <scaled_down_node> -c ID -f value) ; do openstack network agent delete $AGENT ; done
```

<scaled_down_node> を、削除するノードの名前に置き換えます。

7. ノードが新しいインスタンスをスケジュールできないように、オーバークラウド上の削除されたノードでコンピュートサービスが無効になっていることを確認します。

```
(overcloud)$ openstack compute service list
```

コンピュートサービスが無効になっていない場合は、無効にします。

```
(overcloud)$ openstack compute service set <hostname> nova-compute --disable
```

ヒント

--disable-reason オプションを使用して、サービスを無効にする理由についての簡単な説明を追加します。これは、コンピュートサービスを再デプロイする場合に役立ちます。

8. source コマンドでアンダークラウド設定を読み込みます。

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

9. スタックからコンピュートノードを削除します。

```
(undercloud)$ openstack overcloud node delete --stack <overcloud> <node>
```

- **<overcloud>** は、オーバークラウドスタックの名前または UUID に置き換えてください。

- **<node>** を、削除するコンピュータノードのコンピュータサービスホスト名または UUID に置き換えます。



注記

ノードの電源がすでにオフになっている場合、このコマンドは **WARNING** メッセージを返します。

```
Ansible failed, check log at /var/lib/mistral/overcloud/ansible.log
WARNING: Scale-down configuration error. Manual cleanup of some
actions may be necessary. Continuing with node removal.
```

このメッセージは無視しても問題ありません。

10. オーバークラウドノードが削除されるのを待ちます。
11. ノードの削除が完了したら、オーバークラウドスタックのステータスを確認します。

```
(undercloud)$ openstack stack list
```

表12.3 結果

ステータス	説明
UPDATE_COMPLETE	削除操作は正常に完了しました。
UPDATE_FAILED	<p>削除操作が失敗しました。</p> <p>メンテナンスモード中にオーバークラウドノードの削除に失敗した場合は、問題はハードウェアにある可能性があります。</p>

12. インスタンス HA が有効な場合は、以下の操作を実行します。

- a. ノードの Pacemaker リソースをクリーンアップします。

```
$ sudo pcs resource delete <scaled_down_node>
$ sudo cibadmin -o nodes --delete --xml-text '<node id="<scaled_down_node>"/>'
$ sudo cibadmin -o fencing-topology --delete --xml-text '<fencing-level target="
<scaled_down_node>"/>'
$ sudo cibadmin -o status --delete --xml-text '<node_state id="<scaled_down_node>"/>'
$ sudo cibadmin -o status --delete-all --xml-text '<node id="<scaled_down_node>"/>' --
force
```

- b. ノードの STONITH デバイスを削除します。

```
$ sudo pcs stonith delete <device-name>
```

13. オーバークラウドで削除されたコンピュータノードを置き換えない場合は、ノード数を含む環境ファイルの **ComputeCount** パラメーターを減らします。このファイルは、通常 **node-info.yaml** という名前です。たとえば、ノードを1つ削除する場合、ノード数を4から3に減らします。

```
parameter_defaults:
...
  ComputeCount: 3
...
```

ノード数を減らすと、**openstack overcloud deploy** を実行しても director は新規ノードをプロビジョニングしません。

オーバークラウドデプロイメントで削除されたコンピュートノードを置き換える場合は、[Replacing a removed Compute node](#) を参照してください。

12.3.2. 削除されたコンピュートノードの交換

オーバークラウドデプロイメントで削除されたコンピュートノードを置き換えるには、新しいコンピュートノードを登録して検査するか、削除したコンピュートノードを再度追加します。また、ノードをプロビジョニングするようにオーバークラウドを設定する必要があります。

手順

1. オプション: 削除されたコンピュートノードのインデックスを再利用するには、コンピュートノードが削除されたときに拒否リストを置き換えるロールの **RemovalPoliciesMode** パラメーターと **RemovalPolicies** パラメーターを設定します。

```
parameter_defaults:
  <RoleName>RemovalPoliciesMode: update
  <RoleName>RemovalPolicies: [{'resource_list': []}]
```

2. 削除されたコンピュートノードを置き換えます。

- 新しいコンピュートノードを追加するには、新しいノードを登録、検査、およびタグ付けして、プロビジョニングの準備をします。詳細については、[Configuring a basic overcloud](#) を参照してください。
- 手動で削除したコンピュートノードを再度追加するには、ノードをメンテナンスモードから削除します。

```
(undercloud)$ openstack baremetal node maintenance unset <node_uuid>
```

3. 既存のオーバークラウドのデプロイに使用した **openstack overcloud deploy** コマンドを再実行します。
4. デプロイメントプロセスが完了するまで待ちます。
5. ディレクターが新しいコンピュートノードを正常に登録したことを確認します。

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

6. 手順1を実行して、**update** 用のロールの **RemovalPoliciesMode** を設定した場合は、ロールの **RemovalPoliciesMode** をデフォルト値の **append** にリセットして、コンピュートノードが削除されたときにコンピュートノードインデックスを現在の拒否リストに追加する必要があります。

```
parameter_defaults:
  <RoleName>RemovalPoliciesMode: append
```

7. 既存のオーバークラウドのデプロイに使用した **openstack overcloud deploy** コマンドを再実行します。

12.4. CEPH STORAGE ノードの置き換え

director を使用して、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、[コンテナ化された Red Hat Ceph を持つオーバークラウドのデプロイ](#) を参照してください。

12.5. オブジェクトストレージノードの置き換え

本項の手順で、クラスターの整合性を保ちながらオブジェクトストレージノードを置き換える方法を説明します。以下の例では、2 台のノードで設定されるオブジェクトストレージクラスターで、ノード **overcloud-objectstorage-1** を置き換える必要があります。この手順の目的は、ノードを 1 台追加し、その後 **overcloud-objectstorage-1** を削除することです (結果的に、置き換えることになります)。

手順

1. **ObjectStorageCount** パラメーターを使用してオブジェクトストレージ数を増やします。このパラメーターは、通常ノード数を指定する環境ファイルの **node-info.yaml** に含まれています。

```
parameter_defaults:
  ObjectStorageCount: 4
```

ObjectStorageCount パラメーターで、環境内のオブジェクトストレージノードの数を定義します。今回の例では、ノードを 3 つから 4 つにスケーリングします。

2. 更新した **ObjectStorageCount** パラメーターを使用して、デプロイメントコマンドを実行します。

```
$ source ~/stackrc
(undercloud) $ openstack overcloud deploy --templates -e node-info.yaml
ENVIRONMENT_FILES
```

3. デプロイメントコマンドの実行が完了すると、オーバークラウドには追加のオブジェクトストレージノードが含まれるようになります。
4. 新しいノードにデータを複製します。ノードを削除する前に (この場合は **overcloud-objectstorage-1**)、複製のパス が新規ノードで完了するのを待ちます。**/var/log/swift/swift.log** ファイル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 サービスは以下の例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
```

5. リングから不要になったノードを削除するには、**ObjectStorageCount** パラメーターの数を減らして不要になったノードを削除します。今回は 3 に減らします。

```
parameter_defaults:
  ObjectStorageCount: 3
```

- 新規環境ファイル (**remove-object-node.yaml**) を作成します。このファイルでオブジェクトストレージノードを特定し、そのノードを削除します。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']}
```

- デプロイメントコマンドに **node-info.yaml** ファイルと **remove-object-node.yaml** ファイルの両方を含めます。

```
(undercloud) $ openstack overcloud deploy --templates -e node-info.yaml
ENVIRONMENT_FILES -e remove-object-node.yaml
```

director は、オーバークラウドからオブジェクトストレージノードを削除して、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。

12.6. 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。

```
$ 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、**Red Hat OpenStack Platform のアップグレード**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。たとえば、新規コンピュートノードを使用してオーバークラウドをスケールアウトする際に既存のコンピュートノードをブラックリストに登録すると、ブラックリストに登録したノードには **/etc/hosts** および **/etc/ssh/ssh_known_hosts** に加えられた情報が反映されません。これにより、移行先ホストによっては、ライブマイグレーションが失敗する可能性があります。コンピュートノードのブラックリスト登録が解除される次のオーバークラウドデプロイメント時に、これらのノードは **/etc/hosts** および **/etc/ssh/ssh_known_hosts** に加えられた情報で更新されます。**/etc/hosts** ファイルおよび **/etc/ssh/ssh_known_hosts** ファイルは手動で変更しないでください。**/etc/hosts** ファイルおよび **/etc/ssh/ssh_known_hosts** ファイルを変更するには、**Clearing the Blacklist** セクションで説明するように、オーバークラウドのデプロイコマンドを実行します。

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: []
```

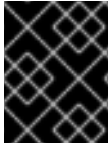
**警告**

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13章 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ます。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。

以下のシナリオの手順を実施して、コントローラーノードを置き換えます。コントローラーノードを置き換えるプロセスでは、**openstack overcloud deploy** コマンドを実行し、コントローラーノードを置き換えるリクエストでオーバークラウドを更新します。



重要

以下の手順は、高可用性環境にのみ適用されます。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。

13.1. コントローラー置き換えの準備

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。

手順

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。

2. 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
```

3. アンダークラウドに、新規ノードプロビジョニング時の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ことを確認します。
4. 新規コントローラーノード用に IP アドレスを再利用する場合は、古いコントローラーが使用したポートを削除するようにしてください。

```
(undercloud) $ openstack port delete <port>
```

5. 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。

6. オーバークラウド 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。以下の例では、コントローラーノードの IP アドレスは、それぞれ 192.168.0.47 と 192.168.0.46 です。

```
(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
```

7. 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。

8. 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"
```

9. 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
```

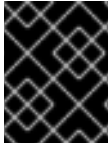
出力では、メンテナンスモードに入っていないすべてのノードが **up** のステータスで表示されるはずです。

10. 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
```

13.2. バックアップまたはスナップショットからのコントローラーノードの復元

コントローラーノードに障害が発生しても物理ディスクがまだ機能している特定のケースでは、ノード自体を交換せずに、バックアップまたはスナップショットからノードを復元できます。



重要

障害が発生したコントローラーノードで PXE ブートに使用される NIC の MAC アドレスが、ディスクの交換後も同じであることを確認してください。

手順

- コントローラーノードが Red Hat Virtualization ノードであり、スナップショットを使用してコントローラーノードのバックアップを取得する場合は、スナップショットからノードを復元します。詳細は、Red Hat Virtualization 仮想マシンガイドの [スナップショットを使用して仮想マシンの復元](#) を参照してください。
- コントローラーノードが Red Hat Virtualization ノードで、バックアップストレージドメインを使用している場合は、バックアップストレージドメインからノードを復元します。詳細は、Red Hat Virtualization 管理ガイドの [バックアップストレージドメインを使用した仮想マシンのバックアップと復元](#) を参照してください。
- Relax-and-Recover (ReaR) ツールからのコントローラーノードのバックアップイメージがある場合は、ReaR ツールを使用してノードを復元します。詳細は、アンダークラウドとコントロールプレーンのバックアップおよびリストアガイドの [コントロールプレーンのリストア](#) を参照してください。
- バックアップまたはスナップショットからノードを復元した後、Galera ノードを個別に復元しないといけない場合があります。詳しくは、アールティクル [How Galera works and how to rescue Galera clusters in the context of Red Hat OpenStack Platform](#) を参照してください。
- バックアップの復元が完了したら、必要なすべての環境ファイルを指定して **openstack overcloud deploy** コマンドを実行し、コントローラーノードの設定がクラスター内の他のノードの設定と一致するようにします。
- ノードのバックアップがない場合は、標準のコントローラー交換手順に従う必要があります。

13.3. CEPH MONITOR デーモンの削除

ストレージクラスターから **ceph-mon** デーモンを削除するには、以下の手順に従います。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

クラスターに新しいコントローラーを追加すると、新しい Ceph monitor デーモンも自動的に追加されます。

手順

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>
```

以下に例を示します。

```
# systemctl stop ceph-mon@overcloud-controller-1
```

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-1 を削除した場合には、controller-1 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
```

変更後:

```
mon host = 172.18.0.22,172.18.0.24
mon initial members = overcloud-controller-2,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 によって該当するオーバークラウドノード上の **ceph.conf** ファイルが更新されます。通常は、director がこの設定ファイルを管理するだけで、手動でファイルを編集する必要はありません。ただし、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は、一貫性を保つために手動でファイルを編集することができます。

6. オプションとして、monitor データをアーカイブし、アーカイブを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>
```

13.4. コントローラーを置き換えるためのクラスター準備

古いノードを置き換える場合には、Pacemaker がノード上で実行されていないことを確認してからそのノードを Pacemaker クラスターから削除する必要があります。

手順

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。

```
(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-1 | ctlplane=192.168.0.45 |
| overcloud-controller-2 | ctlplane=192.168.0.46 |
+-----+-----+
```

2. まだ古いノードにアクセスできる場合は、残りのノードのいずれかにログインし、古いノード上の Pacemaker を停止します。以下の例では、overcloud-controller-1 上の Pacemaker を停止しています。

```
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo pcs status | grep -w Online | grep -w overcloud-controller-1"
(undercloud) $ ssh heat-admin@192.168.0.47 "sudo pcs cluster stop overcloud-controller-1"
```



注記

古いノードが物理的に利用できない、または停止している場合には、そのノードでは Pacemaker はすでに停止しているので、この操作を実施する必要はありません。

3. 古いノードの Pacemaker を停止したら (**pcs status** で **Stopped** と表示される)、各ノードの **corosync** 設定から古いノードを削除して、Corosync を再起動します。この例では、以下のコマンドで **overcloud-controller-0** および **overcloud-controller-2** にログインし、ノードを削除します。

```
(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
```

4. 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
```

5. オーバークラウドデータベースは、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-bundle"
```

13.5. コントローラーノードの再利用

障害が発生したコントローラーノードを再利用して、新しいノードとして再デプロイできます。交換に使用する余分なノードがない場合は、この方法を使用します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 障害が発生したノードをオーバークラウドから関連付け解除します。

```
$ openstack baremetal node undeploy <FAILED_NODE>
```

<FAILED_NODE> を、障害が発生したノードの UUID に置き換えます。このコマンドは、OpenStack ベアメタル (ironic) のノードと OpenStack コンピュート (nova) のオーバークラウドサーバーの関連付けを解除します。ノードのクリーニングを有効にしている場合、このコマンドはノードディスクからファイルシステムも削除します。

3. 新規ノードを **control** プロファイルにタグ付けします。

```
(undercloud) $ openstack baremetal node set --property  
capabilities='profile:control,boot_option:local' <FAILED NODE>
```

4. ディスクの障害が原因でコントローラーノードに障害が発生した場合は、この時点でディスクを交換し、ノードでイントロスペクションを実行して、新しいディスクからイントロスペクションデータを更新できます。

```
$ openstack baremetal node manage <FAILED NODE>  
$ openstack overcloud node introspect --all-manageable --provide
```

障害が発生したノードは、ノードの交換および再デプロイの準備が整いました。ノードの交換を実行すると、障害が発生したノードは新しいノードとして機能し、増加したインデックスを使用します。たとえば、コントロールプレーンクラスターに **overcloud-controller-0**、**overcloud-controller-1**、および **overcloud-controller-2** が含まれており、**overcloud-controller-1** を新しいノードとして再利用する場合は、新しいノード名が **overcloud-controller-3** となります。

13.6. BMC IP アドレスの再利用

障害が発生したコントローラーノードを新しいノードに交換できますが、同じ BMC IP アドレスを保持できます。障害が発生したノードを削除し、BMC IP アドレスを再割り当てし、新しいノードを新しいベアメタルレコードとして追加して、イントロスペクションを実行します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 障害のあるノードを削除します。

```
$ openstack baremetal node undeploy <FAILED_NODE>  
$ openstack baremetal node maintenance set <FAILED_NODE>  
$ openstack baremetal node delete <FAILED_NODE>
```


<**FAILED_NODE**> を、障害が発生したノードの UUID に置き換えます。キューに前のコマンドがある場合は、**openstack baremetal node delete** コマンドが一時的に失敗することがあります。**openstack baremetal node delete** コマンドが失敗した場合は、前のコマンドが完了するまで待ちます。これには最大 5 分かかる場合があります。

3. 障害が発生したノードの BMC IP アドレスを新しいノードに割り当てます。
4. 新しいノードを新しいベアメタルレコードとして追加します。

```
$ openstack overcloud node import newnode.json
```

オーバークラウドノードの登録の詳細は、[オーバークラウドノードの登録](#) を参照してください。

5. 新しいノードでイントロスペクションを実行します。

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

6. 関連付けられていないノードの一覧を表示し、新規ノードの ID を特定します。

```
$ openstack baremetal node list --unassociated
```

7. 新規ノードを **control** プロファイルにタグ付けします。

```
(undercloud) $ openstack baremetal node set --property  
capabilities='profile:control,boot_option:local' <NEW NODE UUID>
```

13.7. コントローラーノード置き換えのトリガー

古いコントローラーノードを削除して新規コントローラーノードに置き換えるには、以下の手順を実施します。

手順

1. 削除するノードの UUID を把握し、それを **NODEID** 変数に保管します。**NODE_NAME** は、削除するノードの名前に置き換えてください。

```
$ NODEID=$(openstack server list -f value -c ID --name NODE_NAME)
```

2. Heat リソース ID を特定するには、以下のコマンドを入力します。

```
$ openstack stack resource show overcloud ControllerServers -f json -c attributes | jq --arg  
NODEID "$NODEID" -c '.attributes.value | keys[] as $k | if .[$k] == $NODEID then "Node  
index \($k) for \(.[$k])" else empty end'
```

3. 以下の内容で環境ファイル **~/templates/remove-controller.yaml** を作成し、削除するコントローラーノードのノードインデックスを含めます。

```
parameters:  
  ControllerRemovalPolicies:  
    [{'resource_list': ['NODE_INDEX']}
```

4. ご自分の環境に該当するその他の環境ファイルと共に **remove-controller.yaml** 環境ファイルを指定して、オーバークラウドデプロイメントコマンドを実行します。

```
(undercloud) $ openstack overcloud deploy --templates \
-e /home/stack/templates/remove-controller.yaml \
[OTHER OPTIONS]
```



注記

-e ~/templates/remove-controller.yaml は、デプロイメントコマンドのこのインスタンスに対してのみ指定します。これ以降のデプロイメント操作からは、この環境ファイルを削除してください。

5. director は古いノードを削除し、次のノードインデックス ID で新しいノードを作成し、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
```

6. デプロイメントコマンドの実行が完了すると、director の出力には古いノードが新規ノードに置き換えられたことが表示されます。

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

これで、新規ノードが稼動状態のコントロールプレーンサービスをホストするようになります。

13.8. コントローラーノード置き換え後のクリーンアップ

ノードの置き換えが完了したら、以下の手順を実施してコントローラークラスターの最終処理を行います。

手順

1. コントローラーノードにログインします。
2. Galera クラスターの Pacemaker 管理を有効にし、新規ノード上で Galera を起動します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource refresh galera-bundle
[heat-admin@overcloud-controller-0 ~]$ sudo pcs resource manage galera-bundle
```

3. 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を解決し、そのサービスを再起動します。

4. director に戻ります。

```
[heat-admin@overcloud-controller-0 ~]$ exit
```

5. 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。

```
$ source ~/overcloudrc
```

6. オーバークラウド環境のネットワークエージェントを確認します。

```
(overcloud) $ openstack network agent list
```

7. 古いノードにエージェントが表示される場合には、そのエージェントを削除します。

```
(overcloud) $ for AGENT in $(openstack network agent list --host overcloud-controller-1.localdomain -c ID -f value) ; do openstack network agent delete $AGENT ; done
```

8. 必要に応じて、新規ノード上の L3 エージェントにホストルーターを追加します。以下のコマンド例では、UUID に 2d1c1dc1-d9d4-4fa9-b2c8-f29cd1a649d4 を使用して L3 エージェントにホストルーター r1 を追加しています。

```
(overcloud) $ openstack network agent add router -l3 2d1c1dc1-d9d4-4fa9-b2c8-f29cd1a649d4 r1
```

9. 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。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
```

10. 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 -c ID -f value) ; do openstack compute service delete $SERVICE ; done
```

第14章 ノードのリブート

アンダークラウドおよびオーバークラウドで、ノードをリブートしなければならない場合があります。以下の手順では、異なるノード種別をリブート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リブートする場合には、各ノードを個別にリブートすることを推奨しています。この方法は、リブート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 環境の全ノードをリブートする場合、リブートの順序は以下のリストを参考にしてください。

推奨されるノードリブート順

1. アンダークラウドノードのリブート
2. コントローラーノードおよびその他のコンポーザブルノードのリブート
3. スタンドアロンの Ceph MON ノードのリブート
4. Ceph Storage ノードのリブート
5. Object Storage サービス (swift) ノードを再起動します。
6. コンピュートノードのリブート

14.1. アンダークラウドノードのリブート

以下の手順では、アンダークラウドノードをリブートします。

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リブートします。

```
$ sudo reboot
```

3. ノードがブートするまで待ちます。

14.2. コントローラーノードおよびコンポーザブルノードのリブート

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リブートします。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. リブートするノードにログインします。
2. オプション: ノードが Pacemaker リソースを使用している場合は、クラスターを停止します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo pcs cluster stop
```

3. ノードをリブートします。

```
[heat-admin@overcloud-controller-0 ~]$ sudo reboot
```

4. ノードがブートするまで待ちます。
5. サービスを確認します。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加わっ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、すべてのサービスが有効化されていることを確認します。

```
[heat-admin@overcloud-controller-0 ~]$ sudo systemctl status
```

- c. すべてのコントローラーノードおよびコンポーザブルノードについて、上記の手順を繰り返します。

14.3. スタンドアロンの CEPH MON ノードのリブート

手順

1. Ceph MON ノードにログインします。
2. ノードをリブートします。

```
$ sudo reboot
```

3. ノードがブートして MON クラスターに再度加わるまで待ちます。

クラスター内の各 MON ノードで、この手順を繰り返します。

14.4. CEPH STORAGE (OSD) クラスターのリブート

以下の手順では、Ceph Storage (OSD) ノードのクラスターをリブートします。

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 Cluster 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. リブートする最初の Ceph Storage ノードを選択して、ログインします。
 3. ノードをリブートします。

```
$ sudo reboot
```

4. ノードがブートするまで待ちます。

5. Ceph MON またはコントローラーノードにログインし、クラスターのステータスを確認します。

```
$ sudo ceph -s
```

pgmap により、すべての **pgs** が正常な状態 (**active+clean**) として報告されることを確認します。

6. Ceph MON またはコントローラーノードからログアウトし、次の Ceph Storage ノードを再起動して、そのステータスを確認します。全 Ceph Storage ノードがリブートされるまで、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
```

14.5. OBJECT STORAGE サービス (SWIFT) ノードの再起動

次の手順では、Object Storage サービス (swift) ノードを再起動します。クラスター内のすべてのオブジェクトストレージノードに対して、次の手順を実行します。

手順

1. オブジェクトストレージノードにログインします。
2. ノードをリブートします。

```
$ sudo reboot
```

3. ノードがブートするまで待ちます。
4. クラスター内のオブジェクトストレージノードごとに再起動を繰り返します。

14.6. コンピュートノードのリブート

コンピュートノードをリブートするには、以下のワークフローを実施します。

- リブートするコンピュート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する。
- インスタンスのダウンタイムを最小限に抑えるために、インスタンスを別のコンピュートノードに移行する。
- 空のコンピュートノードをリブートして有効にする。

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. 再起動するコンピュートノードを特定するには、すべてのコンピュートノードを一覧表示します。

```
$ source ~/stackrc  
(undercloud) $ openstack server list --name compute
```

3. オーバークラウドから、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
```

4. 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
```

5. インスタンスを移行します。移行計画についての詳細は、インスタンス&イメージガイドの [コンピュートノード間の仮想マシンインスタンスの移行](#) を参照してください。
6. コンピュートノードにログインして、リブートします。

```
[heat-admin@overcloud-compute-0 ~]$ sudo reboot
```

7. ノードがブートするまで待ちます。
8. コンピュートノードを有効にします。

```
$ source ~/overcloudrc  
(overcloud) $ openstack compute service set <hostname> nova-compute --enable
```

9. コンピュートノードが有効化されていることを確認します。

```
(overcloud) $ openstack compute service list
```

第15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** も **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。

15.1. 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]
```

次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]
```

15.2. 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、discovery 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。discovery 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 のブート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。これ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 の状態に変更します。

```
$ 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 (秒単位) に変更します。

introspection ramdisk へのアクセス

introspection 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. **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。以下に例を示します。

```
$ openssl passwd -1 mytestpassword
$1$enjRSylw$/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="$1$enjRSylw$/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二重引用符が必要です。

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*
```

15.3. ワークフローおよび実行に関する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。最初のワークフローは、デプロイメントプランをアップロードします。

```
Removing the current plan files
Uploading new plan files
Started Mistral Workflow. Execution ID: aef1e8c6-a862-42de-8bce-073744ed5e6b
Plan updated
```

2つ目のワークフローは、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-LhRIHX/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追跡します。

アクション

関連タスクが実行される際に OpenStack が実施する特定の指示。これには、シェルスクリプトの実行や HTTP リクエストの実行などが含まれます。OpenStack の一部のコンポーネントには、OpenStack Workflow が使用するアクションが組み込まれています。

タスク

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。

ワークフロー

グループ化されて特定の順番で実行されるタスクのセット

実行

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。失敗した状態 (**ERROR**) のワークフローの実行を一覧表示します。

```
$ 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。

15.4. オーバークラウドの作成に関する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。

- オーケストレーション (Heat および Nova サービス)
- ベアメタルプロビジョニング (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のいずれか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。以下のコマンドを実行して、エラーの詳細を表示することもできます。

```
$ openstack stack failures list <OVERCLOUD_NAME> --long
```

<OVERCLOUD_NAME> を実際のオーバークラウドの名前に置き換えてください。



注記

最初のオーバークラウドの作成に失敗した場合は、**openstack stack delete overcloud** コマンドを使用して部分的にデプロイされたオーバークラウドを削除し、再試行できます。これらの最初のオーバークラウドの作成が失敗した場合にのみ、このコマンドを実行してください。このコマンドは、完全にデプロイされて稼働中のオーバークラウドでは実行しないでください。そうしないと、オーバークラウド全体が削除されます。

15.4.1. デプロイメントコマンド履歴へのアクセス

director のデプロイメントコマンドおよび引数の履歴を把握することは、トラブルシューティングおよびサポートに役立ちます。**/home/stack/.tripleo/history** で、これらの情報を確認することができます。

15.4.2. オーケストレーション

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。

15.4.3. ベアメタルプロビジョニング

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。以下の点をチェックしてください。
 - 表にノードが表示されない、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ています。
- Provision State が **active** で、Power State が **power on** の場合、ベアメタルのデプロイメントは正常に完了しています。これは、デプロイメント後の設定ステップで問題が発生したことを意味します。
- 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。ステータスが変わらない場合には、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、このノードでのベアメタルプロビジョニングは失敗しています。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。エラーメッセージが曖昧な場合、ログを使用して明確にすることができます。

```
(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。

15.4.4. デプロイメント後の設定

設定ステージでは多くの事象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 フィールドで、診断に役立つ可能性のある情報がないか確認します。

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** が失敗しているはずです。その際には、**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設定されます。そのステップは以下のとおりです。

表15.1 コントローラーノードの設定ステップ

ステップ	説明
ControllerDeployment_Step1	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
ControllerDeployment_Step2	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
ControllerDeployment_Step3	OpenStack Object Storage (swift) の初期リング構築。OpenStack Platform の全サービス (nova 、 neutron 、 cinder 、 sahara 、 ceilometer 、 heat 、 horizon 、 aodh 、 gnocchi) の設定

ControllerDeployment_Step4	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
ControllerDeployment_Step5	OpenStack Identity (keystone) のプロジェクト、ロール、およびユーザーの初期設定

15.5. 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
```

15.6. "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ブート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ます。このような場合、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。その際には、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**、および **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せん。
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/プロファイルに対応するノードが十分に存在することを確認します。**openstack baremetal node show** の出力で、**properties** フィールドの **capabilities** キーを確認します。たとえば、Compute 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す。
6. イントロスペクションの後に 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
```

15.7. 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。

15.7.1. 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現在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。

15.7.2. 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

リソースおよびそれに対応するノードの一覧を表示します。

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。

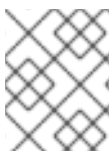
pcs cluster unstandby [node]

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。

15.7.3. 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** コマンドにパイプで渡して、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もできます。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ーを作成します。

これらのオプションを **docker run** コマンドと共に使用して、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、コマンドとして **/bin/bash** を使用して **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

コンテナに障害が発生した場合には、ファイルの内容を詳細に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、調べるができます。

15.7.4. Compute サービスのエラー

コンピュートノードは、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。詳しくは、[コンピュートノード間の仮想マシンインスタンスの移行](#) を参照してください。

15.7.5. 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、[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。

15.8. 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 サービス (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。環境をモニターリングして、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。アンダークラウドの場合は、`crontab -u keystone -e` 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。
- `openstack overcloud deploy` を実行するたびに、Heat はデータベースの `raw_template` テーブルにあるすべての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、過剰なリソースを消費する場合があります。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。同時にプロビジョニング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/galera.cnf` ファイルを編集します。調整項目の推奨値は、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)。50 に設定します。

`innodb_max_purge_lag`

この変数は、パージ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 エンジンを実行します。

```
$ sudo systemctl restart openstack-heat-engine
```

15.9. 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アーティクルを参照してください。

- [How to collect all required logs for Red Hat Support to investigate an OpenStack issue](#)

15.10. 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15.2 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	/var/log/nova/nova-compute.log
OpenStack Compute API の対話	/var/log/nova/nova-api.log
OpenStack Compute コンダクターのログ	/var/log/nova/nova-conductor.log
OpenStack Orchestration ログ	/var/log/heat/heat-engine.log
OpenStack Orchestration API の対話	/var/log/heat/heat-api.log
OpenStack Orchestration CloudFormations ログ	/var/log/heat/heat-api-cfn.log
OpenStack Bare Metal コンダクターのログ	/var/log/ironic/ironic-conductor.log
OpenStack Bare Metal API の対話	/var/log/ironic/ironic-api.log
イントロスペクション	/var/log/ironic-inspector/ironic-inspector.log
OpenStack Workflow Engine ログ	/var/log/mistral/engine.log
OpenStack Workflow Executor のログ	/var/log/mistral/executor.log
OpenStack Workflow API の対話	/var/log/mistral/api.log

表15.3 オーバークラウドの重要なログ

情報	ログの場所
cloud-init に関するログ	/var/log/cloud-init.log
オーバークラウドの設定 (最後に実行した Puppet のサマリー)	/var/lib/puppet/state/last_run_summary.yaml
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)	/var/lib/puppet/state/last_run_report.yaml
オーバークラウドの設定 (全 Puppet レポート)	/var/lib/puppet/reports/overcloud-*/*
オーバークラウドの設定 (実行した Puppet 毎の標準出力)	/var/run/heat-config/deployed/*-stdout.log
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)	/var/run/heat-config/deployed/*-stderr.log
高可用性に関するログ	/var/log/pacemaker.log

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、[オーバークラウドの高度なカスタマイズの オーバークラウドのパブリックエンドポイントでの SSL/TLS の有効化](#) 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しているかどうかを確認してください。ファイルが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。

```
$ 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もあります。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。

```
$ sudo openssl genrsa -out ca.key.pem 4096
$ sudo 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 鍵の作成

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_host** 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。

v3_req セクションに **subjectAltName = @alt_names** を追加します。

alt_names セクションを編集して、以下のエントリーを追加します。

- **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 エントリーとしてパブリック API の IP アドレスも追加します。

openssl.cnf の詳しい情報は **man openssl.cnf** を実行してください。

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 鍵の作成](#)」で作成した SSL/TLS 鍵を **-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。このファイルは **-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** オプションとして追加します。
- 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。「[SSL/TLS 鍵の作成](#)」で作成した SSL/TLS キーと共にこの証明書を使用して、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** ファイル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
```

アンダークラウドがすでにインストールされている状態で、既存の設定を更新するために **openstack undercloud install** を実行する場合には、haproxy サービスを再起動して設定を再読み込みする必要があります。

```
$ sudo systemctl restart haproxy
```

「[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) の開発した、IT インフラストラクチャー向け標準 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

システムリソースへの正規パス。このパスには、そのシステムの root サービス、バージョン、パス/一意 ID を含める必要があります。たとえば、**/redfish/v1/Systems/CX34R87**。

redfish_verify_ca

ベースボード管理コントローラー (BMC) の Redfish サービスが、認識済み認証局 (CA) により署名された有効な TLS 証明書を使用するように設定されていない場合、ironic の Redfish クライアントは BMC への接続に失敗します。**redfish_verify_ca** オプションを **false** に設定して、エラーをミュートします。ただし、BMC 認証を無効にすると、BMC のアクセスセキュリティが低下するので、注意してください。

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イスです。

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イスです。

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 インターフェイスの 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
```

- 正常にイントロスペクションを実施するためには、HP ノードの iLO ファームウェアバージョンは、最低でも 1.85 (2015 年 5 月 13 日版) でなければなりません。この iLO ファームウェアバージョンを使用するノードで、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)



注記

Cisco UCS は非推奨となり、Red Hat OpenStack Platform (RHOSP) 16.0 から廃止される予定です。

Cisco の UCS は、コンピュート、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 インターフェイスの 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、UCS 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 インターフェイスのユーザー名とパスワード

pm_addr

iRMC インターフェイスの 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)

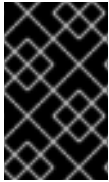
センサーデータの取得方法。 **ipmitool** または **scci** です。デフォルトは **ipmitool** です。

- 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します。したがって、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ていません。

KVM ホストの設定

1. KVM ホスト上で、OpenStack Platform リポジトリを有効化して **python2-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 python2-virtualbmc
```

2. **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 を作成するには、それぞれの BMC にアクセスするためのポートを定義し、認証情報を設定し、以下のコマンドを入力します。

```
$ vbmc add Node01 --port 6230 --username admin --password PASSWORD
$ vbmc add Node02 --port 6231 --username admin --password PASSWORD
```

3. ホストの該当するポートを開放します。

```
$ sudo firewall-cmd --zone=public \
--add-port=6230/udp \
--add-port=6231/udp
```

4. 変更を永続化します。

```
$ sudo firewall-cmd --runtime-to-permanent
```

5. 変更がファイアウォール設定に適用され、ポートが開放されていることを確認します。

```
$ sudo firewall-cmd --list-all
```

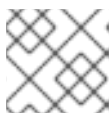


注記

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。

6. 以下のコマンドを使用して、作成した各 BMC を起動します。

```
$ vbmc start Node01
$ vbmc start Node02
```



注記

KVM ホストのリブート後には、この手順を繰り返す必要があります。

7. **ipmitool** を使用してノードを管理できることを確認するには、リモートノードの電源ステータスを表示します。

```
$ ipmitool -I lanplus -U admin -P PASSWORD -H 127.0.0.1 -p 6231 power status
Chassis Power is off
```

ノードの登録

/home/stack/instackenv.json ノード登録ファイルで、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を指定します。

pm_addr

ノードが含まれる KVM ホストの IP アドレスを指定します。

pm_port

KVM ホスト上の特定のノードにアクセスするためのポートを指定します。

mac

ノード上のネットワークインターフェイスの MAC アドレス一覧を指定します。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。

以下に例を示します。

```
{
  "nodes": [
    {
      "pm_type": "ipmi",
      "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": "ipmi",
      "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

非推奨の **pxe_ssh** ドライバーから新しい仮想 BMC 方式の使用に切り替えて、既存のノードを移行することができます。以下のコマンドは、ノードが **ipmi** 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 ipmi \
  --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。

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めます。たとえば、**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イスの 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にするには、以下の手順を実施します。

1. **undercloud.conf** ファイルの **enabled_hardware_types** オプションに **staging-ovirt** を追加します。

```
enabled_hardware_types = ipmi,staging-ovirt
```

2. **python-ovirt-engine-sdk4.x86_64** パッケージをインストールします。

```
$ sudo yum install python-ovirt-engine-sdk4
```

3. **openstack undercloud install** コマンドを実行します。

```
$ openstack undercloud install
```

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プロセス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ていません。

pm_type

このオプションを **fake_pxe** 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にするには、**undercloud.conf** の **enabled_driver** オプションに **fake_pxe** を追加し、**openstack undercloud install** を再実行します。
- **instackenv.json** ノードインベントリーファイルで、手動で管理するノードの **pm_type** を **fake_pxe** に設定します。
- 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、ノードをリブートします。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、ノードのステータスが **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リブートします。

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、ブート時には別のカーネルおよび 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基本的なパーティションレイアウトを作成します。ただし、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できません。



重要

以下のプロセスでは、director のイメージビルド機能を使用します。Red Hat では、本項に記載の指針に従ってビルドされたイメージのみをサポートしています。これらとは異なる仕様でビルドされたカスタムイメージはサポートされていません。

セキュリティが強化されたイメージには、セキュリティ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** ディレクトリーを別のボリュームまたはパーティションにマウントし、**rw**、**nosuid**、**nodedv**、**noexec**、および **relatime** のフラグを付ける。
- **/var**、**/var/log**、および **/var/log/audit** ディレクトリーを別のボリュームまたはパーティションにマウントし、**rw** および **relatime** のフラグを付ける。
- **/home** ディレクトリーを別のパーティションまたはボリュームにマウントし、**rw**、**nodedv**、**relatime** のフラグを付ける。
- **GRUB_CMDLINE_LINUX** の設定に以下の変更を加える。
 - 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する。
- 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** および **telnet**) がデフォルトでインストールされるので、それらをイメージから削除する。
- セキュリティに必要な新しい **screen** パッケージを追加する。

セキュリティが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する
2. 登録固有の環境変数を設定する
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する
4. イメージを作成する

5. そのイメージをデプロイメントにアップロードする

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. ディスクイメージの環境変数

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



注記

イメージのビルドプロセスにより、イメージは一時的に Red Hat サブスクリプションに登録され、システムがイメージのビルドプロセスを完了すると登録を解除します。

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

DIB_LOCAL_IMAGE

ベースに使用するローカルイメージを設定します。

REG_ACTIVATION_KEY

登録プロセスの一部で代わりにアクティベーションキーを使用します。

REG_AUTO_ATTACH

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

REG_BASE_URL

パッケージをプルするためのコンテンツ配信サーバーのベース URL。カスタマーポータル Subscription Management のデフォルトプロセスでは <https://cdn.redhat.com> を使用します。Red Hat Satellite 6 サーバーを使用している場合は、このパラメーターにお使いの Satellite サーバーのベース URL を使用する必要があります。

REG_ENVIRONMENT

組織内の環境に登録します。

REG_METHOD

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

REG_ORG

イメージに登録する組織

REG_POOL_ID

製品のサブスクリプション情報のプール ID

REG_PASSWORD

イメージに登録するユーザーアカウントのパスワードを指定します。

REG_REPOS

コンマ区切りのリポジトリ名の文字列 (空白なし)。この文字列の各リポジトリは **subscription-manager** で有効にされます。

以下に示すセキュリティーが強化された完全なディスクイメージ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_SAT_URL

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

REG_SERVER_URL

使用するサブスクリプションサービスのホスト名を指定します。Red Hat カスタマーポータルの場合のデフォルトは **subscription.rhn.redhat.com** です。Red Hat Satellite 6 サーバーを使用する場合は、このパラメーターにお使いの Satellite サーバーのホスト名を使用する必要があります。

REG_USER

イメージを登録するアカウントのユーザー名を指定します。

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、セキュリティーが強化されたイメージのサイズは 20 GB 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。以下のセクションでは、イメージのサイズを 40 GB に増やしています。ご自分のニーズに合わせて、さらにパーティショニングレイアウトやディスクのサイズを変更することができます。

パーティション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する。
- **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 設定は、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、論理ボリュームの変更後のパーティション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: 40G
- lvm:
  name: lvm
  base: [ root ]
  pvs:
    - name: pv
      base: root
      options: [ "--force" ]
  vgs:
    - name: vg
      base: [ "pv" ]
      options: [ "--force" ]
  lvs:
    - name: lv_root
      base: vg
      extents: 23%VG
    - name: lv_tmp
      base: vg
      extents: 4%VG
    - name: lv_var
      base: vg
      extents: 45%VG
    - name: lv_log
      base: vg
      extents: 23%VG
    - name: lv_audit
      base: vg
      extents: 4%VG
    - name: lv_home
      base: vg
      extents: 1%VG
- mkfs:
  name: fs_root
  base: lv_root
  type: xfs
```

```

    label: "img-rootfs"
    mount:
      mount_point: /
      fstab:
        options: "rw,relatime"
        fsck-passno: 1
- mkfs:
  name: fs_tmp
  base: lv_tmp
  type: xfs
  mount:
    mount_point: /tmp
    fstab:
      options: "rw,nosuid,nodev,noexec,relatime"
      fsck-passno: 2
- mkfs:
  name: fs_var
  base: lv_var
  type: xfs
  mount:
    mount_point: /var
    fstab:
      options: "rw,relatime"
      fsck-passno: 2
- mkfs:
  name: fs_log
  base: lv_log
  type: xfs
  mount:
    mount_point: /var/log
    fstab:
      options: "rw,relatime"
      fsck-passno: 3
- mkfs:
  name: fs_audit
  base: lv_audit
  type: xfs
  mount:
    mount_point: /var/log/audit
    fstab:
      options: "rw,relatime"
      fsck-passno: 4
- mkfs:
  name: fs_home
  base: lv_home
  type: xfs
  mount:
    mount_point: /home
    fstab:
      options: "rw,nodev,relatime"
      fsck-passno: 2
'''

```

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、イメージ用に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 (20 GB) 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しなければならない場合があります。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
```

**注記**

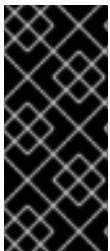
UEFI の完全なディスクイメージの場合は、`/usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-images-uefi.yaml` を使用します。

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 nousb'
  DIB_IMAGE_SIZE: '40' 1
  COMPRESS_IMAGE: '1'
```

1 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、セキュリティが強化されたイメージのサイズを 40 GB 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 40 GB の空き容量が必要となります。

**重要**

最終的に director が物理ディスクにイメージを書き込む際に、ディスクの最後に 64 MB のコンフィグドライブパーティションが作成されます。完全なディスクイメージを作成する場合には、この追加パーティションを収容できるように、物理ディスクのサイズがディスクイメージより大きいことを確認してください。

C.4. セキュリティーが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 \
--config-file /usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-images-rhel7.yaml
```

/home/stack/overcloud-hardened-images-custom.yaml カスタム設定ファイルには、「[イメージサイズの変更](#)」で変更した新たなディスクサイズが含まれます。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。

UEFI の完全なディスクイメージの場合は、**/usr/share/openstack-tripleo-common/image-yaml/overcloud-hardened-images-uefi-rhel7.yaml** 設定ファイルを使用します。

作成した **overcloud-hardened-full.qcow2** イメージには、必要なセキュリティ機能がすべて含まれています。

C.5. セキュリティーが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。セキュリティが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が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、セキュリティが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](#) のアート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** のキャパビリティ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。各ルールでは、**説明**、**条件**、および**アクション** が定義されます。

description

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。

以下に例を示します。

```
"description": "A new rule for my node tagging policy"
```

conditions

ここでは、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、以下の属性が含まれます。

- **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確認します。
- **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、以下の属性が含まれます。

- **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

評価する値を定義します。フィールド、演算、および値の条件が満たされる場合には、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。

以下に例を示します。

```
"conditions": [
  {
    "field": "local_gb",
    "op": "ge",
    "value": 1024
  }
],
```

actions

条件が true を返す場合にアクションが実行されます。ここでは、**action** キーと、**action** の値に応じた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 ノードの属性を設定します。Ironic の属性へのパス (例: `/driver_info/ipmi_address`) を指定する **path** フィールドおよび設定する **value** が必要です。
- **set-capability**: Ironic ノードのケイパビリティを設定します。新しいケイパビリティの名前と値を指定する **name** および **value** 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、その一覧に値を追加します。オプションの **unique** パラメーターを 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しません。

以下に例を示します。

```
"actions": [
  {
    "action": "set-capability",
    "name": "profile",
    "value": "swift-storage"
  }
]
```

E.2. ポリシーファイルの例

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**) の例を以下に示します。

```
[
  {
    "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つのルールで設定されています。

- 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、コンピュートノードまたはコントローラー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。

属性	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、「 ルートディスクの定義 」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 を有効にした場合には (「[director の設定パラメーター](#)」を参照)、HAProxy 設定で使用する 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、改行せずに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 RED HAT OPENSTACK PLATFORM FOR POWER

Red Hat OpenStack Platform を新規インストールする際に、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できるようになりました。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。各システムのインストール詳細は、本ガイドのこれまでのセクションで説明しています。

G.1. CEPH STORAGE

マルチアーキテクチャクラウドにおいて外部 Ceph へのアクセスを設定する場合には、クライアントキーおよびその他の Ceph 固有パラメーターと共に **CephAnsiblePlaybook** パラメーターを **/usr/share/ceph-ansible/site.yml.sample** に設定します。

以下に例を示します。

```
parameter_defaults:
  CephAnsiblePlaybook: /usr/share/ceph-ansible/site.yml.sample
  CephClientKey: AQDLOh1VgEp6FRAAFzT7Zw+Y9V6JJExQAsRnRQ==
  CephClusterFSID: 4b5c8c0a-ff60-454b-a1b4-9747aa737d19
  CephExternalMonHost: 172.16.1.7, 172.16.1.8
```

G.2. コンポーザブルサービス

一般にコントローラーノードの一部となる以下のサービスは、テクノロジープレビューとしてカスタムロールでの使用が可能です。したがって、Red Hat による完全なサポートは提供されません。

- **Cinder**
- **Glance**
- **Keystone**
- **Neutron**
- **Swift**

詳しい情報は、オーバークラウドの高度なカスタマイズの [コンポーザブルサービスとカスタムロール](#) を参照してください。上記のサービスをコントローラーノードから専用の ppc64le ノードに移動する手順の一例を、以下に示します。

```
(undercloud) [stack@director ~]$ rsync -a /usr/share/openstack-tripleo-heat-templates/. ~/templates
(undercloud) [stack@director ~]$ cd ~/templates/roles
(undercloud) [stack@director roles]$ cat <<EO_TEMPLATE >ControllerPPC64LE.yaml
#####
# Role: ControllerPPC64LE                                     #
#####
- name: ControllerPPC64LE
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  tags:
```

```

- primary
- controller
networks:
- External
- InternalApi
- Storage
- StorageMgmt
- Tenant
# For systems with both IPv4 and IPv6, you may specify a gateway network for
# each, such as ['ControlPlane', 'External']
default_route_networks: ['External']
HostnameFormatDefault: '%stackname%-controllerppc64le-%index%'
ImageDefault: ppc64le-overcloud-full
ServicesDefault:
- OS::TripleO::Services::Aide
- OS::TripleO::Services::AuditD
- OS::TripleO::Services::CACerts
- OS::TripleO::Services::CephClient
- OS::TripleO::Services::CephExternal
- OS::TripleO::Services::CertmongerUser
- OS::TripleO::Services::CinderApi
- OS::TripleO::Services::CinderBackendDellPs
- OS::TripleO::Services::CinderBackendDellSc
- OS::TripleO::Services::CinderBackendDellEMCUnity
- OS::TripleO::Services::CinderBackendDellEMCVMAXISCSI
- OS::TripleO::Services::CinderBackendDellEMCVNX
- OS::TripleO::Services::CinderBackendDellEMCXTREMIOISCSI
- OS::TripleO::Services::CinderBackendNetApp
- OS::TripleO::Services::CinderBackendScaleIO
- OS::TripleO::Services::CinderBackendVRTSHyperScale
- OS::TripleO::Services::CinderBackup
- OS::TripleO::Services::CinderHPELeftHandISCSI
- OS::TripleO::Services::CinderScheduler
- OS::TripleO::Services::CinderVolume
- OS::TripleO::Services::Collectd
- OS::TripleO::Services::Docker
- OS::TripleO::Services::Fluentd
- OS::TripleO::Services::GlanceApi
- OS::TripleO::Services::GlanceRegistry
- OS::TripleO::Services::Ipssec
- OS::TripleO::Services::Iscsid
- OS::TripleO::Services::Kernel
- OS::TripleO::Services::Keystone
- OS::TripleO::Services::LoginDefs
- OS::TripleO::Services::MySQLClient
- OS::TripleO::Services::NeutronApi
- OS::TripleO::Services::NeutronBgpVpnApi
- OS::TripleO::Services::NeutronSfcApi
- OS::TripleO::Services::NeutronCorePlugin
- OS::TripleO::Services::NeutronDhcpAgent
- OS::TripleO::Services::NeutronL2gwAgent
- OS::TripleO::Services::NeutronL2gwApi
- OS::TripleO::Services::NeutronL3Agent
- OS::TripleO::Services::NeutronLbaasv2Agent
- OS::TripleO::Services::NeutronLbaasv2Api
- OS::TripleO::Services::NeutronLinuxbridgeAgent

```

- OS::TripleO::Services::NeutronMetadataAgent
- OS::TripleO::Services::NeutronML2FujitsuCfab
- OS::TripleO::Services::NeutronML2FujitsuFossw
- OS::TripleO::Services::NeutronOvsAgent
- OS::TripleO::Services::NeutronVppAgent
- OS::TripleO::Services::Ntp
- OS::TripleO::Services::ContainersLogrotateCron
- OS::TripleO::Services::OpenDaylightOvs
- OS::TripleO::Services::Rhsm
- OS::TripleO::Services::RsyslogSidecar
- OS::TripleO::Services::Securetty
- OS::TripleO::Services::SensuClient
- OS::TripleO::Services::SkydiveAgent
- OS::TripleO::Services::Snmp
- OS::TripleO::Services::Sshd
- OS::TripleO::Services::SwiftProxy
- OS::TripleO::Services::SwiftDispersion
- OS::TripleO::Services::SwiftRingBuilder
- OS::TripleO::Services::SwiftStorage
- OS::TripleO::Services::Timezone
- OS::TripleO::Services::TripleoFirewall
- OS::TripleO::Services::TripleoPackages
- OS::TripleO::Services::Tuned
- OS::TripleO::Services::Vpp
- OS::TripleO::Services::OVNController
- OS::TripleO::Services::OVNMetadataAgent
- OS::TripleO::Services::Ptp

EO_TEMPLATE

```
(undercloud) [stack@director roles]$ sed -i~ -e '/OS::TripleO::Services::\
```

```
(Cinder\|Glance\|Swift\|Keystone\|Neutron\)/d' Controller.yaml
```

```
(undercloud) [stack@director roles]$ cd ../
```

```
(undercloud) [stack@director templates]$ openstack overcloud roles generate \
```

```
--roles-path roles -o roles_data.yaml \
```

```
Controller Compute ComputePPC64LE ControllerPPC64LE BlockStorage ObjectStorage  
CephStorage
```