Red Hat OpenStack Platform 17.1

director を使用した Red Hat OpenStack Platform のインストールと管理

director を使用した Red Hat OpenStack Platform クラウドの作成および管理

Last Updated: 2024-09-11

Red Hat OpenStack Platform 17.1 director を使用した Red Hat OpenStack Platform のインストールと管理

director を使用した Red Hat OpenStack Platform クラウドの作成および管理

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

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 17 をインストールします。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

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、用語の置き換えは、今後の複数のリリースにわたって段階的に実施されます。詳細は、Red Hat CTO である Chris Wright のメッセージ をご覧ください。

RED HAT ドキュメントへのフィードバック (英語のみ)

Red Hat ドキュメントに対するご意見をお聞かせください。ドキュメントの改善点があればお知らせください。

Jira でドキュメントのフィードバックを提供する

問題の作成 フォームを使用して、Red Hat OpenStack Services on OpenShift (RHOSO) または Red Hat OpenStack Platform (RHOSP) の以前のリリースのドキュメントに関するフィードバックを提供します。RHOSO または RHOSP ドキュメントの問題を作成すると、その問題は RHOSO Jira プロジェクトに記録され、フィードバックの進行状況を追跡できるようになります。

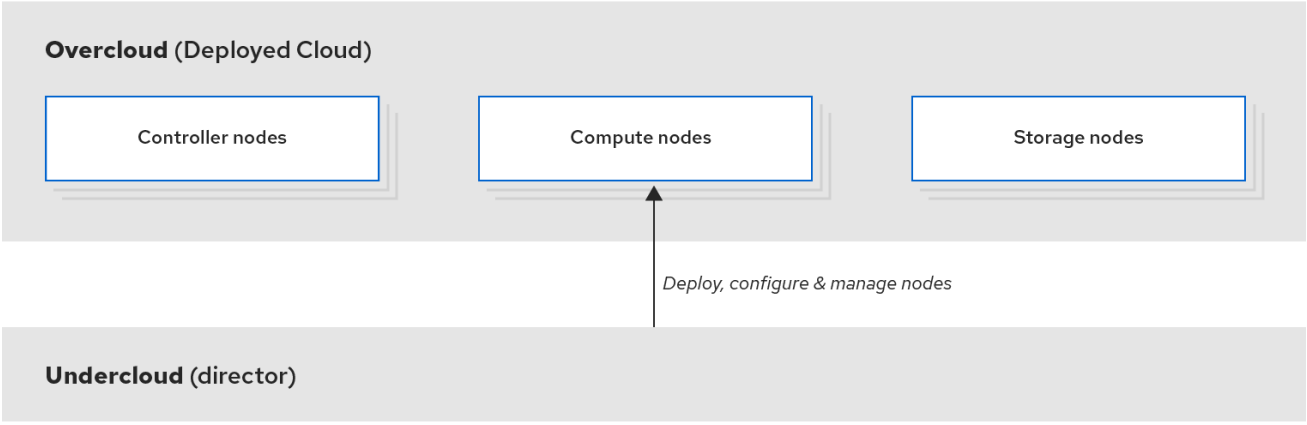
問題の作成 フォームを完了するには、Jira にログインしていることを確認してください。Red Hat Jira アカウントをお持ちでない場合は、https://issues.redhat.com でアカウントを作成できます。

1. 次のリンクをクリックして、**問題の作成** ページを開きます (問題の作成)。
2. **Summary** フィールドと **Description** フィールドに入力します。**Description** フィールドに、ドキュメントの URL、章またはセクション番号、および問題の詳しい説明を入力します。フォーム内の他のフィールドは変更しないでください。
3. **Create** をクリックします。

第1章 DIRECTOR の概要

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 をベースとしています。director を使用すると、RHOSP ノードとして使用するベアメタルシステムをプロビジョニングおよび制御可能な、完全に機能するスリムで堅牢な RHOSP 環境をインストールできます。

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。



1.1. アンダークラウドを理解する

アンダークラウドは、Red Hat OpenStack Platform (RHOSP) director ツールセットを含む主要管理ノードです。これは、RHOSP 環境を形成する RHOSP ノード (オーバークラウド) をプロビジョニングおよび管理するためのコンポーネントを含む、単一システムの RHOSP インストールです。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。

RHOSP サービス

アンダークラウドは、RHOSP サービス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。各サービスは、アンダークラウドの個別のコンテナー内で動作します。

- Identity サービス (keystone): director サービスの認証と認可を提供する
- Bare Metal Provisioning サービス (ironic) と Compute サービス (nova): ベアメタルノードを管理する
- Networking サービス (neutron) および Open vSwitch: ベアメタルノードのネットワークを制御する
- Orchestration サービス (heat): 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を提供する

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の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 RHOSP サービスを含めるかを選択できます。これにより、新しいノードタイプをモデル化したり、特定のコンポーネントを独自のホストに分離したりする方法が提供されます。

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制御を行い、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 RHOSP をインストールします。この機能を使用して、ベアメタルシステムを RHOSP ノードとしてプロビジョニングできます。電源管理ドライバーのすべてのリストについては、電源管理ドライバー を参照してください。

Orchestration

アンダークラウドには、環境のプランのセットに対応する YAML テンプレートセットが含まれます。アンダークラウドはこれらのプランをインポートし、その指示に従って結果として RHOSP 環境を作成します。このプランに含まれるフックを使用して、環境作成プロセスの特定のポイントとして、カスタマイズを組み込むこともできます。

1.2. オーバークラウドについて

オーバークラウドは、アンダークラウドが構築することで得られる Red Hat OpenStack Platform (RHOSP) 環境です。オーバークラウドは、作成する RHOSP 環境に基づいて定義するさまざまなロールを持つ複数のノードで構成されます。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。

Controller

Controller ノードは、RHOSP 環境に管理、ネットワーキング、高可用性を提供します。推奨される RHOSP 環境には、高可用性クラスター内に 3 つの Controller ノードが一緒に含まれています。デフォルトの Controller 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。

- Dashboard サービス (horizon)
- Identity サービス (keystone)
- Compute サービス (nova)
- Networking サービス (neutron)
- Image サービス (glance)
- Block Storage サービス (cinder)
- Object Storage サービス (swift)
- Orchestration サービス (heat)
- Shared File Systems サービス (manila)
- Bare Metal Provisioning サービス (ironic)
- Load Balancing-as-a-Service (octavia)
- Key Manager サービス (barbican)
- MariaDB

- Open vSwitch
- 高可用性サービス向けの Pacemaker および Galera

Compute

Compute ノードは RHOSP 環境にコンピュートリソースを提供します。Compute 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 Compute ノードには、以下のコンポーネントが含まれます。

- Compute サービス (nova)
- KVM/QEMU
- Open vSwitch

ストレージ

Storage ノードは RHOSP 環境にストレージを提供します。以下のリストで、RHOSP のさまざまな Storage ノード種別を説明します。

- 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、環境の一部として Ceph Storage ノードをデプロイする場合には、director により Ceph Monitor が Controller ノードにインストールされます。
- Block Storage (cinder): 高可用性 Controller ノードの外部 Block Storage 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。
  - Block Storage (cinder) ボリューム
  - Telemetry エージェント
  - Open vSwitch
- Object Storage (swift): これらのノードは、RHOSP Object Storage 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 プロキシーを介してオブジェクトストレージノードにアクセスします。オブジェクトストレージノードには、以下のコンポーネントが含まれます。
  - Object Storage (swift) ストレージ
  - Telemetry エージェント
  - Open vSwitch

1.3. RHOSP での RED HAT CEPH STORAGE の使用

通常、Red Hat OpenStack Platform (RHOSP) 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。Block Storage 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。Image サービス (glance)、Block Storage サービス (cinder)、および Compute サービス (nova) を単一ノードにデプロイすると、数千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。この課題は、RHOSP を外部にスケーリングすることで解決できます。

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、RHOSP 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リング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ように思えるかもしれませんが、Red Hat Ceph Storage はブロックデバイスイメージをクラスター全体のオブジェクトとしてストライプ化します。これは、大規模な Ceph Block Device イメージのパフォーマンスが、スタンドアロンディスクよりも優れていることを意味します。Ceph Block Device 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 の詳細な情報は、Red Hat Ceph Storage を参照してください。

第2章 アンダークラウドのプランニング

アンダークラウドに director を設定してインストールする前に、アンダークラウドホストを計画して、特定の要件を満たしていることを確認する必要があります。

2.1. アンダークラウドネットワークの準備

アンダークラウドには、少なくとも 2 枚の 1 Gbps ネットワークインターフェイスカード (NIC) が必要です。これは、以下の主要ネットワークそれぞれに 1 枚必要です。

- **プロビジョニングまたはコントロールプレーンネットワーク**: Ansible 設定の実行時に、director がノードをプロビジョニングし、SSH 経由でノードにアクセスするために使用するネットワーク。このネットワークでは、アンダークラウドからオーバークラウドノードへの SSH アクセスも可能です。アンダークラウドには、このネットワーク上の他のノードのイントロスペクションおよびプロビジョニング用 DHCP サービスが含まれます。つまり、このネットワーク上にその他の DHCP サービスは必要ありません。director は、このネットワークのインターフェイスを設定します。

- **外部ネットワーク**: Red Hat OpenStack Platform (RHOSP) リポジトリー、コンテナーイメージソース、および DNS サーバーや NTP サーバーなどの他のサーバーへのアクセスを可能にします。ワークステーションからアンダークラウドへの標準アクセスには、このネットワークを使用します。外部ネットワークにアクセスするためには、アンダークラウド上でインターフェイスを手動で設定する必要があります。

ネットワークを計画する際には、以下のガイドラインを確認してください。

- Red Hat は、プロビジョニングとコントロールプレーンに 1 つのネットワークを使用し、データプレーンに別のネットワークを使用することを推奨します。

- プロビジョニングおよびコントロールプレーンネットワークは、Linux ボンディング上または個々のインターフェイスで設定できます。Linux ボンディングを使用する場合は、アクティブバックアップボンディングタイプとして設定します。

  - 非 Controller ノードでは、プロビジョニングおよびコントロールプレーンネットワークのトラフィック量は比較的少なく、高帯域幅や負荷分散は必要ありません。

  - Controller では、プロビジョニングおよびコントロールプレーンネットワークに追加の帯域幅が必要です。帯域幅が増加する理由は、Controller が他のロールで多くのノードにサービスを提供するためです。環境に頻繁に変更を加える場合も、より多くの帯域幅が必要になります。
    50 を超える Compute ノードを管理するコントローラー、または 5 つ以上のベアメタルノードを同時にプロビジョニングするコントローラーは、非コントローラーノードのインターフェイスの帯域幅を 4 - 10 倍にする必要があります。

- 50 を超えるオーバークラウドノードがプロビジョニングされる場合、アンダークラウドはプロビジョニングネットワークへのより高い帯域幅の接続を持つ必要があります。

- ワークステーションから director マシンへのアクセスに使用する NIC を、プロビジョニングまたはコントロールプレーン NIC として使用しないでください。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。

- イントロスペクション中は、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ごとに少なくとも 1 つの一時 IP アドレスを含めます。
- デプロイメント中は、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ごとに少なくとも 1 つの永続的な IP アドレスを含めます。
- プロビジョニングネットワーク上のオーバークラウド高可用性クラスターの仮想 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に、この範囲にさらに IP アドレスを追加します。

- Controller 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 Keystone Admin Endpoint を internal_api network に移行する方法 を参照してください。

2.2. 環境規模の判断

アンダークラウドをインストールする前に、環境の規模を判断します。環境をプランニングする際には、以下の要素を考慮してください。

オーバークラウドにデプロイするノードの数

アンダークラウドは、オーバークラウド内の各ノードを管理します。オーバークラウドノードのプロビジョニングには、アンダークラウドのリソースが使用されます。アンダークラウドには、すべてのオーバークラウドノードを適切にプロビジョニングし管理するのに十分なリソースを提供する必要があります。

アンダークラウドで実行する同時操作の数

アンダークラウド上のほとんどの Red Hat OpenStack Platform (RHOSP) サービスは、ワーカーのセットを使用します。それぞれのワーカーは、そのサービスに固有の操作を実行します。複数のワーカーを用いると、同時に操作を実行することができます。アンダークラウドのデフォルトのワーカー数は、アンダークラウドの合計 CPU スレッド数の半分です。ここでは、スレッド数とは CPU コア数にハイパースレッディングの値を掛けたものを指します。たとえば、アンダークラウドの CPU スレッド数が 16 の場合には、デフォルトでは、director のサービスにより 8 つのワーカーが提供されます。デフォルトでは、director のサービスに最小および最大のワーカー数も適用されます。

サービス	最小値	最大値
Orchestration サービス (heat)	4	24
その他すべてのサービス	2	12

アンダークラウドの CPU およびメモリーの最低要件を以下に示します。

- Intel 64 または AMD64 CPU 拡張機能をサポートする、8 スレッド 64 ビット x86 プロセッサー。これにより、各アンダークラウドサービスに 4 つのワーカーが提供されます。
- 最小 24 GB の RAM

多数のワーカーを使用するには、以下の推奨事項に従ってアンダークラウドの仮想 CPU 数およびメモリー容量を増やします。

- **最小値:** 1 スレッドあたり 1.5 GB のメモリーを使用します。たとえば、48 スレッドのマシンの場合、heat 用 24 ワーカーおよびその他のサービス用 12 ワーカーを最低限動作させるのに、72 GB の RAM が必要です。
- **推奨値:** 1 スレッドあたり 3 GB のメモリーを使用します。たとえば、48 スレッドのマシンの場合、heat 用 24 ワーカーおよびその他のサービス用 12 ワーカーを推奨される状態で動作させるのに、144 GB の RAM が必要です。

2.3. アンダークラウドのディスクサイズ

アンダークラウドのディスクサイズとして、ルートディスク上に少なくとも **100 GB** の空きディスク領域があることが推奨されます。

- コンテナーイメージ用に 20 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 70 GB 以上

2.4. 仮想化されたアンダークラウドノードのサポート

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

プラットフォーム	備考
Kernel-based Virtual Machine (KVM)	Red Hat OpenStack Platform、OpenShift Virtualization、および Red Hat Enterprise Linux with KVM の認定ゲストオペレーティングシステム に記載されているように、Red Hat Enterprise Linux によってホストされます。
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。

重要

ハイパーバイザーが Red Hat Enterprise Linux 9.2 ゲストをサポートしていることを確認します。

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検討してください。

ネットワークの考慮事項

電源管理

アンダークラウド仮想マシン (VM) は、オーバークラウドノードの電源管理デバイスにアクセスする必要があります。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

プロビジョニングネットワーク (**ctlplane**) 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 NIC と同じネットワークに接続するブリッジを作成します。

不明なアドレスからのトラフィックを許可する

ハイパーバイザーがアンダークラウドをブロックして、不明なアドレスからトラフィックが送信されないように、仮想アンダークラウドハイパーバイザー (VMware ESX または ESXi) を設定する必要があります。

- IPv4 **ctlplane** ネットワーク: 偽造送信を許可します。
- IPv6 **ctlplane** ネットワーク: 偽造送信、MAC アドレスの変更、無差別モード操作を許可します。
  VMware ESX または ESXi を設定する方法の詳細は、VMware ドキュメント Web サイトの vSphere 標準スイッチのセキュリティー保護 を参照してください。

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。仮想マシンを再起動するだけでは不十分です。

2.5. アンダークラウドのリポジトリー

Red Hat Enterprise Linux (RHEL) 9.2 で Red Hat OpenStack Platform (RHOSP) 17.1 を実行します。

注記

リポジトリーを Red Hat Satellite と同期する場合は、特定バージョンの Red Hat Enterprise Linux リポジトリーを有効にすることができます。ただし、選択したバージョンに関係なく、リポジトリーは同じままです。たとえば、BaseOS リポジトリーの 9.2 バージョンを有効にすることができますが、リポジトリー名は、選択した特定のバージョンに関係なく **rhel-9-for-x86_64-baseos-eus-rpms** のままになります。

警告

ここで指定されたもの以外のリポジトリーはサポートされていません。別途推奨されない限り、以下の表に記載されている以外の製品またはリポジトリーを有効にしないでください。有効にすると、パッケージの依存関係の問題が発生する可能性があります。Extra Packages for Enterprise Linux (EPEL) を有効にしないでください。

コアリポジトリー

以下の表には、アンダークラウドをインストールするためのコアリポジトリーをまとめています。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-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ー
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux の高可用性ツール。Controller ノードの高可用性に使用します。
Red Hat OpenStack Platform for RHEL 9 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform のコアリポジトリー。Red Hat OpenStack Platform director のパッケージが含まれます。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	OpenStack Platform 用 Open vSwitch (OVS) パッケージを提供します。

第3章 DIRECTOR インストールの準備

director をインストールおよび設定するには、アンダークラウドを Red Hat Customer Portal または Red Hat Satellite サーバーに登録し、director パッケージをインストールし、インストール中にコンテナーイメージを取得するために director のコンテナーイメージソースを設定するなどの準備作業を完了する必要があります。

3.1. アンダークラウドの準備

director をインストールする前に、ホストマシンでいくつかの基本設定を完了する必要があります。

手順

1. お使いの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 ~]$
```

6. システムイメージおよび heat テンプレート用のディレクトリーを作成します。

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

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。ローカルファイルシステムの管理を容易にするために、Red Hat ではこれらのディレクトリーを作成することを推奨します。

7. 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
```

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。

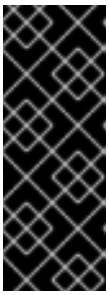
```
[stack@director ~]$ sudo hostnamectl set-hostname undercloud.example.com
```

8. アンダークラウドホストの完全修飾ドメイン名 (FQDN) を解決できる DNS サーバーを使用していない場合は、**/etc/hosts** を編集してシステムホスト名のエントリーを追加します。**/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 FQDN に **undercloud.example.com** が使用され、IP アドレスに **10.0.0.1** を使用する場合には、**/etc/hosts** ファイルに以下の行を追加します。

```
10.0.0.1  undercloud.example.com undercloud
```

9. Red Hat OpenStack Platform director をオーバークラウドまたはその認証プロバイダーとは別のドメインに配置する予定の場合には、追加のドメインを /etc/resolv.conf に加える必要があります。

```
search overcloud.com idp.overcloud.com
```



重要

RHOSP 環境でポートの名前を内部的に解決するには、DNS のポートエクステンションの DNS ドメイン (**dns_domain_ports**) を有効にする必要があります。**NeutronDnsDomain** のデフォルト値 (**openstacklocal**) を使用する場合、ネットワークサービスは DNS のポート名を内部的に解決しません。詳細は、**Red Hat OpenStack Platform ネットワークの設定** の DNS がポートに割り当てる名前を指定する を参照してください。

3.2. アンダークラウドの登録およびサブスクリプションのアタッチ

director をインストールする前に、**subscription-manager** を実行し、アンダークラウドを登録して有効な Red Hat OpenStack Platform サブスクリプションをアタッチする必要があります。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. Red Hat コンテンツ配信ネットワークまたは Red Hat Satellite のどちらかにシステムを登録します。たとえば、システムをコンテンツ配信ネットワークに登録するには、以下のコマンドを実行します。要求されたら、カスタマーポータルのユーザー名およびパスワードを入力します。

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

3. Red Hat OpenStack Platform (RHOSP) director のエンタイトルメントプール ID を検索します。

```
[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)
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
```

4. **Pool ID** の値を特定して、Red Hat OpenStack Platform 17.1 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
```

5. アンダークラウドを Red Hat Enterprise Linux 9.2 にロックします。

```
$ sudo subscription-manager release --set=9.2
```

3.3. アンダークラウド用リポジトリーの有効化

アンダークラウドに必要なリポジトリーを有効にし、システムパッケージを最新バージョンに更新します。

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

```
[stack@director ~]$ sudo subscription-manager repos --disable=*
[stack@director ~]$ sudo subscription-manager repos \
--enable=rhel-9-for-x86_64-baseos-eus-rpms          \
--enable=rhel-9-for-x86_64-appstream-eus-rpms       \
--enable=rhel-9-for-x86_64-highavailability-eus-rpms \
--enable=openstack-17.1-for-rhel-9-x86_64-rpms \
--enable=fast-datapath-for-rhel-9-x86_64-rpms
```

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

3. 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。

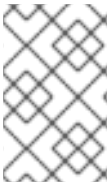
```
[stack@director ~]$ sudo dnf update -y
[stack@director ~]$ sudo reboot
```

4. 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 dnf install -y python3-tripleoclient
```

3.4. コンテナーイメージの準備

アンダークラウドのインストールには、コンテナーイメージの取得先およびその保存方法を定義するための環境ファイルが必要です。コンテナーイメージの準備に使用できる環境ファイルを生成およびカスタマイズします。



注記

アンダークラウド用に特定のコンテナーイメージバージョンを設定する必要がある場合は、イメージを特定のバージョンに固定する必要があります。詳細は、アンダークラウド用コンテナーイメージのピニング を参照してください。

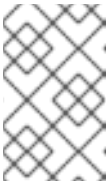
手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. デフォルトのコンテナーイメージ準備ファイルを生成します。

```
$ openstack tripleo container image prepare default \
  --local-push-destination \
  --output-env-file containers-prepare-parameter.yaml
```

上記のコマンドでは、以下の追加オプションを使用しています。

- **--local-push-destination**: コンテナーイメージの保管場所として、アンダークラウド上のレジストリーを設定します。つまり、director は必要なイメージを Red Hat Container Catalog からプルし、それをアンダークラウド上のレジストリーにプッシュします。director はこのレジストリーをコンテナーイメージのソースとして使用します。Red Hat Container Catalog から直接プルする場合には、このオプションを省略します。
- **--output-env-file**: 環境ファイルの名前です。このファイルには、コンテナーイメージを準備するためのパラメーターが含まれます。ここでは、ファイル名は **containers-prepare-parameter.yaml** です。



注記

アンダークラウドとオーバークラウド両方のコンテナーイメージのソースを定義するのに、同じ **containers-prepare-parameter.yaml** ファイルを使用することができます。

3. 要件に合わせて **containers-prepare-parameter.yaml** を変更します。コンテナーイメージのパラメーターの詳細は、コンテナーイメージ準備のパラメーター を参照してください。

3.5. プライベートレジストリーからのコンテナーイメージの取得

registry.redhat.io レジストリーにアクセスしてイメージをプルするには、認証が必要です。**registry.redhat.io** およびその他のプライベートレジストリーで認証するには、**containers-prepare-parameter.yaml** ファイルに **ContainerImageRegistryCredentials** および **ContainerImageRegistryLogin** パラメーターを含めます。

ContainerImageRegistryCredentials

一部のコンテナーイメージレジストリーでは、イメージにアクセスするのに認証が必要です。そのような場合には、**containers-prepare-parameter.yaml** 環境ファイルの **ContainerImageRegistryCredentials** パラメーターを使用します。**ContainerImageRegistryCredentials** パラメーターは、プライベートレジストリーの URL に基づ

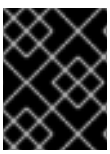
くキーのセットを使用します。それぞれのプライベートレジストリーの URL は、独自のキーと値のペアを使用して、ユーザー名 (キー) およびパスワード (値) を定義します。これにより、複数のプライベートレジストリーに対して認証情報を指定することができます。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      namespace: registry.redhat.io/...
      ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      my_username: my_password
```

上記の例の **my_username** および **my_password** を、実際の認証情報に置き換えてください。Red Hat では、個人のユーザー認証情報を使用する代わりに、レジストリーサービスアカウントを作成し、それらの認証情報を使用して **registry.redhat.io** コンテンツにアクセスすることを推奨します。

複数のレジストリーの認証情報を指定するには、**ContainerImageRegistryCredentials** でレジストリーごとに複数のキー/ペアの値を設定します。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      namespace: registry.redhat.io/...
      ...
  - push_destination: true
    set:
      namespace: registry.internalsite.com/...
      ...
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      myuser: 'p@55w0rd!'
    registry.internalsite.com:
      myuser2: '0th3rp@55w0rd!'
    '192.0.2.1:8787':
      myuser3: '@n0th3rp@55w0rd!'
```



重要

デフォルトの **ContainerImagePrepare** パラメーターは、認証が必要な **registry.redhat.io** からコンテナーイメージをプルします。

詳細は、Red Hat コンテナーレジストリーの認証 を参照してください。

ContainerImageRegistryLogin

ContainerImageRegistryLogin パラメーターを使用して、コンテナーイメージを取得するために、オーバークラウドノードシステムがリモートレジストリーにログインする必要があるかどうかを制御します。このような状況は、アンダークラウドを使用してイメージをホストする代わりに、オーバークラウドノードがイメージを直接プルする場合に発生します。

特定の設定について、**push_destination** が false に設定されている、または使用されていない場合は、**ContainerImageRegistryLogin** を **true** に設定する必要があります。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: false
    set:
      namespace: registry.redhat.io/...
      ...
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      myuser: 'p@55w0rd!'
  ContainerImageRegistryLogin: true
```

ただし、オーバークラウドノードに **ContainerImageRegistryCredentials** で定義されたレジストリーホストへのネットワーク接続がなく、**ContainerImageRegistryLogin** を **true** に設定すると、ログインを試みる際にデプロイメントが失敗する可能性があります。オーバークラウドノードに **ContainerImageRegistryCredentials** で定義されたレジストリーホストへのネットワーク接続がない場合、**push_destination** を **true** に、**ContainerImageRegistryLogin** を **false** に設定して、オーバークラウドノードがアンダークラウドからイメージをプルできるようにします。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      namespace: registry.redhat.io/...
      ...
  ...
  ContainerImageRegistryCredentials:
    registry.redhat.io:
      myuser: 'p@55w0rd!'
  ContainerImageRegistryLogin: false
```

第4章 アンダークラウドへの DIRECTOR のインストール

director を設定してインストールするには、**undercloud.conf** ファイルに適切なパラメーターを設定し、undercloud installation コマンドを実行します。director をインストールしたら、ノードのプロビジョニング中に director がベアメタルノードへの書き込みに使用するオーバークラウドイメージをインポートします。

4.1. アンダークラウドの設定

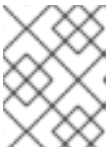
director をインストールする前に、アンダークラウドを設定する必要があります。アンダークラウドは、director が **stack** ユーザーのホームディレクトリーから読み取る **undercloud.conf** ファイルで設定します。

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. サンプルの **undercloud.conf** ファイルを **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/python-tripleoclient/undercloud.conf.sample \
  ~/undercloud.conf
```

3. デプロイメントの **undercloud.conf** ファイル内のパラメーターの値を変更します。アンダークラウドの設定に使用できるパラメーターの詳細は、アンダークラウド設定パラメーター を参照してください。



注記

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、director はデフォルト値を使用します。

4. **undercloud.conf** ファイルを保存します。

4.2. アンダークラウド設定パラメーター

以下のリストで、**undercloud.conf** ファイルを設定するパラメーターを説明します。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。

重要

少なくとも、コンテナーイメージの設定が含まれる環境ファイルに **container_images_file** パラメーターを設定する必要があります。このパラメーターに適切なファイルを正しく設定しないと、director は **ContainerImagePrepare** パラメーターからコンテナーイメージのルールセットを取得すること
や、**ContainerImageRegistryCredentials** パラメーターからコンテナーレジストリーの認証情報を取得することができなくなります。

デフォルト

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

additional_architectures

オーバークラウドがサポートする追加の (カーネル) アーキテクチャーのリスト。現在、オーバークラウドは **x86_64** アーキテクチャーのみをサポートしています。

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に限り、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、証明書をトラストチェーンに追加します。

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。

cleanup

一時的なファイルを削除します。デプロイメント中に使用される一時ファイルを保持するには、これを **False** に設定します。一時ファイルは、エラーが発生した場合にデプロイメントのデバッグに役立ちます。

container_cli

コンテナー管理用の CLI ツール。このパラメーターは、**podman** に設定したままにしてください。Red Hat Enterprise Linux 9.2 がサポートするのは、**podman** だけです。

container_healthcheck_disabled

コンテナー化されたサービスのヘルスチェックを無効にします。Red Hat は、ヘルスチェックを有効にし、このオプションを **false** に設定したままにすることを推奨します。

container_images_file

コンテナーイメージ情報が含まれる heat 環境ファイル。このファイルには、以下のエントリーを含めることができます。

- 必要なすべてのコンテナーイメージのパラメーター
- 必要なイメージの準備を実施する **ContainerImagePrepare** パラメーター。このパラメーターが含まれるファイルの名前は、通常 **containers-prepare-parameter.yaml** です。

container_insecure_registries

podman が使用するセキュアではないレジストリーのリスト。プライベートコンテナーレジストリー等の別のソースからイメージをプルする場合には、このパラメーターを使用します。多くの場合、**podman** は Red Hat Container Catalog または Satellite サーバー (アンダークラウドが Satellite に登録されている場合) のいずれかからコンテナーイメージをプルするための証明書を持ちます。

container_registry_mirror

設定により **podman** が使用するオプションの **registry-mirror**

custom_env_files

アンダークラウドのインストールに追加する新たな環境ファイル

deployment_user

アンダークラウドをインストールするユーザー。現在のデフォルトユーザー **stack** を使用する場合には、このパラメーターを未設定のままにします。

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** パラメーターを有効にし、**enabled_hardware_types** のリストにドライバーを含める必要があります。

enable_ironic; enable_ironic_inspector; enable_tempest; enable_validations

director で有効にするコアサービスを定義します。これらのパラメーターは **true** に設定されたままにします。

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** 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。

enable_routed_networks

ルーティングされたコントロールプレーンネットワークのサポートを有効に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リスト

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。

heat_container_image

使用する heat コンテナーイメージの URL。未設定のままにします。

heat_native

heat-all を使用してホストベースのアンダークラウド設定を実行します。**true** のままにします。

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** ディレクトリーにコピーされ、階層の最初のファイルに設定されます。この機能の使用の詳細は、アンダークラウドへの hieradata の設定 を参照してください。

inspection_extras

イントロスペクション時に追加のハードウェアコレクションを有効化するかどうかを定義します。このパラメーターを使用するには、イントロスペクションイメージに **python-hardware** または **python-hardware-detect** パッケージが必要です。

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_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。ベンチマークを有効にするには、このパラメーターを **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。

ipv6_address_mode

アンダークラウドのプロビジョニングネットワーク用の IPv6 アドレス設定モード。このパラメーターに設定できる値のリストを以下に示します。

- dhcpv6-stateless: ルーター広告 (RA) を使用するアドレス設定と DHCPv6 を使用するオプションの情報
- dhcpv6-stateful: DHCPv6 を使用するアドレス設定とオプションの情報

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。標準の PXE を使用するには、このパラメーターを **false** に設定します。PowerPC デプロイメント、またはハイブリッド PowerPC および x86 デプロイメントの場合は、この値を **false** に設定します。

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。director は、DHCP および PXE ブートサービスにもこのデバイスを使用します。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: em0: <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 em0
       valid_lft 3462sec preferred_lft 3462sec
    inet6 fe80::5054:ff:fe75:2409/64 scope link
       valid_lft forever preferred_lft forever
3: em1: <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 **em0** を使用し、プロビジョニング NIC は、現在設定されていない **em1** を使用します。この場合は、**local_interface** を **em1** 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_ip

director のプロビジョニング NIC 用に定義する IP アドレス。director は、DHCP および PXE ブートサービスにもこの IP アドレスを使用します。この IP アドレスが環境内の既存の IP アドレスまたはサブネットと競合するなどの理由により、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、この値をデフォルトの **192.168.24.1/24** のままにします。
IPv6 の場合、ステートフル接続とステートレス接続の両方をサポートするには、ローカル IP アドレス接頭辞接頭辞の長さを **/64** にする必要があります。

local_mtu

local_interface に使用する最大伝送単位 (MTU)。アンダークラウドでは 1500 以下にします。

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 **ctlplane-subnet** です。

net_config_override

ネットワーク設定のオーバーライドテンプレートへのパス。このパラメーターを設定すると、アンダークラウドは JSON または YAML 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、**undercloud.conf** で設定された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。アンダークラウドネットワークインターフェイスのカスタマイズの詳細は、アンダークラウドネットワークインターフェイスの設定 を参照してください。

networks_file

heat をオーバーライドするネットワークファイル

output_dir

状態、処理された heat テンプレート、および Ansible デプロイメントファイルを出力するディレクトリー

overcloud_domain_name

オーバークラウドをデプロイする際に使用する DNS ドメイン名

注記

オーバークラウドを設定する際に、**CloudDomain** にこのパラメーター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。

roles_file

アンダークラウドのインストールで、デフォルトロールファイルを上書きするのに使用するロールファイル。director のインストールにデフォルトのロールファイルが使用されるように、このパラメーターは未設定のままにすることを強く推奨します。

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。スケジューリング時に競合状態にならないように、この値を 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必要があります。

service_principal

この証明書を使用するサービスの Kerberos プリンシパル。FreeIPA など CA で Kerberos プリンシパルが必要な場合に限り、このパラメーターを使用します。

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リスト。デフォルト値に含まれるのは、**ctlplane-subnet** サブネットのみです。詳細は、サブネット を参照してください。

templates

上書きする heat テンプレートファイル

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_admin_host が **local_ip** と同じ IP ネットワークにない場合は、アンダークラウド上の管理 API がリッスンするインターフェイスを設定する必要があります。デフォルトでは、管理 API は **br-ctlplane** インターフェイスでリッスンします。アンダークラウドネットワークインターフェイスの設定方法に関する詳細は、アンダークラウドネットワークインターフェイスの設定 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。**DEBUG** ログレベルを有効にするには、この値を **true** に設定します。

undercloud_enable_selinux

デプロイメント時に、SELinux を有効または無効にします。問題をデバッグする場合以外は、この値を **true** に設定したままにすることを強く推奨します。

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。このパラメーターを指定すると、アンダークラウドのインストールでホスト名すべてに設定されます。指定しないと、アンダークラウドは現在のホスト名を使用しますが、システムのホスト名すべてを適切に設定しておく必要があります。

undercloud_log_file

アンダークラウドのインストールログおよびアップグレードログを保管するログファイルへのパス。デフォルトでは、ログファイルはホームディレクトリー内の **install-undercloud.log** です。たとえば、**/home/stack/install-undercloud.log** のようになります。

undercloud_nameservers

アンダークラウドのホスト名解決に使用する DNS ネームサーバーのリスト

undercloud_ntp_servers

アンダークラウドの日付と時刻を同期できるようにする Network Time Protocol サーバーのリスト

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_public_host が **local_ip** と同じ IP ネットワークにない場合は、**PublicVirtualInterface** パラメーターを、アンダークラウド上のパブリック API がリッスンする公開インターフェイスに設定する必要があります。デフォルトでは、パブリック API は **br-ctlplane** インターフェイスでリッスンします。カスタム環境ファイルに **PublicVirtualInterface** パラメーターを設定し、**custom_env_files** パラメーターを設定して、**undercloud.conf** ファイルにカスタム環境ファイルを含めます。

アンダークラウドネットワークインターフェイスのカスタマイズの詳細は、アンダークラウドネットワークインターフェイスの設定 を参照してください。

undercloud_service_certificate

OpenStack SSL/TLS 通信の証明書の場所とファイル名。理想的には、信頼できる認証局から、この証明書を取得します。それ以外の場合は、独自の自己署名の証明書を生成します。

undercloud_timezone

アンダークラウド用ホストのタイムゾーン。タイムゾーンを指定しなければ、director は既存のタイムゾーン設定を使用します。

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。

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には、**undercloud.conf** ファイルで以下のような設定を使用します。

```
[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。

重要

director がサブネットを作成した後、director はサブネットの IP アドレスを変更することはできません。

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 経由で外部アクセスすることができるように、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) のメカニズムが提供されます。

注記

director 設定は、適切な **sysctl** カーネルパラメーターを使用して IP フォワーディングも自動的に有効化します。

dhcp_start、dhcp_end

オーバークラウドノードの DHCP 割り当て範囲 (開始アドレスと終了アドレス)。ノードを割り当てるのに十分な IP アドレスがこの範囲に含まれるようにします。サブネットに指定されていない場合、director は **local_ip**、**gateway**、**undercloud_admin_host**、**undercloud_public_host**、および **Inspection_iprange** パラメーターに設定された値をサブネットの完全な IP 範囲から削除して、割り当てプールを決定します。
開始アドレスと終了アドレスのペアのリストを指定することで、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に非連続割り当てプールを設定することができます。または、**dhcp_exclude** オプションを使用して、IP アドレス範囲内の IP アドレスを除外することもできます。たとえば、次の設定は両方とも割り当てプール **172.20.0.100-172.20.0.150** と **172.20.0.200-172.20.0.250** を作成します。

オプション 1

```
dhcp_start = 172.20.0.100,172.20.0.200
dhcp_end = 172.20.0.150,172.20.0.250
```

オプション 2

```
dhcp_start = 172.20.0.100
dhcp_end = 172.20.0.250
dhcp_exclude = 172.20.0.151-172.20.0.199
```

dhcp_exclude

DHCP 割り当て範囲で除外する IP アドレスたとえば、次の設定では、IP アドレス **172.20.0.105** と IP アドレス範囲 **172.20.0.210-172.20.0.219** が除外されます。

```
dhcp_exclude = 172.20.0.105,172.20.0.210-172.20.0.219
```

dns_nameservers

サブネットに固有の DNS ネームサーバー。サブネットにネームサーバーが定義されていない場合には、サブネットは **undercloud_nameservers** パラメーターで定義されるネームサーバーを使用します。

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直接外部ゲートウェイを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。

host_routes

このネットワーク上のオーバークラウドインスタンス用の neutron が管理するサブネットのホストルート。このパラメーターにより、アンダークラウド上の **local_subnet** のホストルートも設定されます。

inspection_iprange

検査プロセス中に使用するこのネットワーク上のノードの一時的な IP 範囲。この範囲は、**dhcp_start** と **dhcp_end** で定義された範囲と重複することはできませんが、同じ IP サブネット内になければなりません。

4.3. 環境ファイルを使用したアンダークラウドの設定

undercloud.conf ファイルを使用して、アンダークラウドの主要なパラメーターを設定します。heat パラメーターが含まれる環境ファイルを使用して、アンダークラウドの追加設定を行うこともできます。

手順

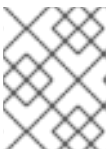
1. **/home/stack/templates/custom-undercloud-params.yaml** という名前で環境ファイルを作成します。
2. このファイルを編集して、必要な heat パラメーターを追加します。たとえば、特定の OpenStack Platform サービスのデバッグを有効にするには、**custom-undercloud-params.yaml** ファイルに以下のスニペットを追加します。

```
parameter_defaults:
  Debug: True
```

アンダークラウドに設定できる heat パラメーターに関する情報は、アンダークラウド設定用の標準 heat パラメーター を参照してください。

3. 環境ファイルを保存します。
4. **undercloud.conf** ファイルを編集し、**custom_env_files** パラメーターまでスクロールします。作成した **custom-undercloud-params.yaml** 環境ファイルをポイントするようにパラメーターを編集します。

```
custom_env_files = /home/stack/templates/custom-undercloud-params.yaml
```



注記

コンマ区切りリストを使用して、複数の環境ファイルを指定することができます。

アンダークラウドの次回インストールまたはアップグレード操作時に、この環境ファイルが director のインストールに追加されます。

4.4. アンダークラウド設定用の標準 HEAT パラメーター

以下の表には、アンダークラウド用のカスタム環境ファイルで設定する標準の heat パラメーターをまとめています。

パラメーター	説明
AdminPassword	アンダークラウドの **admin** ユーザーのパスワードを設定します。
AdminEmail	アンダークラウドの **admin** ユーザーの電子メールアドレスを設定します。
Debug	デバッグモードを有効にします。

カスタム環境ファイルの **parameter_defaults** セクションで、これらのパラメーターを設定します。

```
parameter_defaults:
  Debug: True
  AdminPassword: "myp@ssw0rd!"
  AdminEmail: "admin@example.com"
```

4.5. 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。

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、以下のスニペットを追加します。

   ```
   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** ファイルの **hieradata_override** パラメーターを、新しい **/home/stack/hieradata.yaml** ファイルのパスに設定します。

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

4.6. DIRECTOR のインストール

director をインストールして基本的なインストール後タスクを行うには、以下の手順を実施します。

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
```

このコマンドにより、director の設定スクリプトが起動します。director は追加のパッケージをインストールし、**undercloud.conf** の設定に従ってサービスを設定し、すべての RHOSP サービスコンテナーを起動します。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、2 つのファイルが生成されます。

- **/home/stack/tripleo-deploy/undercloud/tripleo-undercloud-passwords.yaml** - director サービスのすべてのパスワードのリスト。
- **/home/stack/stackrc**: director 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. RHOSP サービスコンテナーが実行中であることを確認します。

```
[stack@director ~]$ sudo podman ps -a --format "{{.Names}} {{.Status}}"
```

次のコマンド出力は、RHOSP サービスコンテナーが実行中 (**Up**) であることを示しています。

```
memcached Up 3 hours (healthy)
haproxy Up 3 hours
rabbitmq Up 3 hours (healthy)
mysql Up 3 hours (healthy)
iscsid Up 3 hours (healthy)
keystone Up 3 hours (healthy)
keystone_cron Up 3 hours (healthy)
neutron_api Up 3 hours (healthy)
logrotate_crond Up 3 hours (healthy)
neutron_dhcp Up 3 hours (healthy)
neutron_l3_agent Up 3 hours (healthy)
neutron_ovs_agent Up 3 hours (healthy)
ironic_api Up 3 hours (healthy)
ironic_conductor Up 3 hours (healthy)
ironic_neutron_agent Up 3 hours (healthy)
ironic_pxe_tftp Up 3 hours (healthy)
ironic_pxe_http Up 3 hours (unhealthy)
ironic_inspector Up 3 hours (healthy)
ironic_inspector_dnsmasq Up 3 hours (healthy)
neutron-dnsmasq-qdhcp-30d628e6-45e6-499d-8003-28c0bc066487 Up 3 hours
...
```

3. **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 コマンドラインツールが使用できるようになりました。

4.7. 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のに、複数のディスクイメージが必要です。

- イントロスペクションカーネルおよび ramdisk: PXE ブートでのベアメタルシステムのイントロスペクション用
- 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用
- オーバークラウドカーネル、ramdisk、完全なイメージで、director がノードのハードディスクに書き込むベースオーバークラウドシステムを形成しています。

必要なイメージを入手してインストールできます。他の Red Hat OpenStack Platform (RHOSP) サービスを実行したくない場合、またはサブスクリプションエンタイトルメントの 1 つを使用したくない場合は、basic イメージを取得してインストールし、ベア OS をプロビジョニングすることもできます。

注記

RHOSP デプロイメントで IPv6 を使用する場合は、**cloud-init** ネットワーク設定を無効にするように、オーバークラウドイメージを変更する必要があります。イメージの変更の詳細は、Red Hat ナレッジベースソリューション Modifying the Red Hat Linux OpenStack Platform Overcloud Image with virt-customize を参照してください。

4.7.1. オーバークラウドイメージのインストール

Red Hat OpenStack Platform (RHOSP) インストールには、director 用の **overcloud-hardened-uefi-full.qcow2** オーバークラウドイメージを提供するパッケージが含まれています。このイメージは、デフォルトの CPU アーキテクチャー x86-64 でオーバークラウドをデプロイするために必要です。これらのイメージを director にインポートすると、イントロスペクションイメージも director PXE サーバーにインストールされます。

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. **stackrc** ファイルを取得します。

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

3. **rhosp-director-images-uefi-x86_64** および **rhosp-director-images-ipa-x86_64** パッケージをインストールします。

```
(undercloud) [stack@director ~]$ sudo dnf install rhosp-director-images-uefi-x86_64 rhosp-director-images-ipa-x86_64
```

4. **stack** ユーザーのホームディレクトリー **/home/stack/images** に **images** ディレクトリーを作成します。

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

ディレクトリーがすでに存在する場合は、この手順をスキップしてください。

5. イメージアーカイブを **images** ディレクトリーにデプロイメントします。

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

6. 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/
```

このコマンドは、イメージ形式を QCOW から RAW に変換し、イメージのアップロードの進行状況に関する詳細な更新を提供します。

7. オーバークラウドのイメージが **/var/lib/ironic/images/** にコピーされていることを確認します。

```
(undercloud) [stack@director images]$ ls -l /var/lib/ironic/images/
total 1955660
-rw-r--r--. 1 root 42422 40442450944 Jan 29 11:59 overcloud-hardened-uefi-full.raw
```

8. director がイントロスペクション PXE イメージを **/var/lib/ironic/httpboot** にコピーしたことを確認します。

```
(undercloud) [stack@director images]$ ls -l /var/lib/ironic/httpboot
total 417296
-rwxr-xr-x. 1 root  root    6639920 Jan 29 14:48 agent.kernel
-rw-r--r--. 1 root  root  420656424 Jan 29 14:48 agent.ramdisk
-rw-r--r--. 1 42422 42422       758 Jan 29 14:29 boot.ipxe
-rw-r--r--. 1 42422 42422       488 Jan 29 14:16 inspector.ipxe
```

4.7.2. 最小限のオーバークラウドイメージ

overcloud-minimal イメージを使用すると、他の Red Hat OpenStack Platform (RHOSP) サービスを実行したり、サブスクリプションエンタイトメントを消費したりしたくないベア OS をプロビジョニングすることが可能です。

RHOSP のインストールには、director 用に次のオーバークラウドイメージを提供する **overcloud-minimal** パッケージが含まれています。

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

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. **stackrc** ファイルを取得します。

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

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

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

4. 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**) にデプロイメントします。

```
(undercloud) [stack@director ~]$ cd ~/images
(undercloud) [stack@director images]$ tar xf /usr/share/rhosp-director-images/overcloud-minimal-latest-17.1.tar
```

5. 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/ --image-type os --os-image-name overcloud-minimal.qcow2
```

このコマンドは、イメージのアップロードの進行状況に関する最新情報を提供します。

```
Image "file:///var/lib/ironic/images/overcloud-minimal.vmlinuz" was copied.
+------------------------------------------------------+------------------+----------+
|                         Path                         |       Name       |   Size   |
+------------------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.vmlinuz | overcloud-minimal | 11172880 |
+------------------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.initrd" was copied.
+-----------------------------------------------------+------------------+----------+
|                         Path                        |       Name       |   Size   |
+-----------------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.initrd | overcloud-minimal | 63575845 |
+-----------------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.raw" was copied.
+-----------------------------------------------------+------------------+------------+
|                         Path                        |       Name       |    Size    |
+-----------------------------------------------------+------------------+------------+
| file:///var/lib/ironic/images/overcloud-minimal.raw | overcloud-minimal | 2912878592 |
+-----------------------------------------------------+------------------+------------+
```

4.8. アンダークラウド設定の更新

新たな要件に合わせて、アンダークラウドの設定を変更する必要がある場合は、該当する設定ファイルを編集し、**openstack undercloud install** コマンドを再度実行して、インストール後のアンダークラウド設定に変更を加えることができます。

手順

1. アンダークラウドの設定ファイルを変更します。以下の例では、**undercloud.conf** ファイルを編集して、有効なハードウェア種別のリストに **idrac** ハードウェア種別を追加しています。

```
enabled_hardware_types = ipmi,redfish,idrac
```

2. **openstack undercloud install** コマンドを実行し、新たな変更を反映させてアンダークラウドを更新します。

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

コマンドの実行が完了するまで待ちます。

3. **stack** ユーザーを初期化し、コマンドラインツールを使用します。

```
[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 ~]$
```

4. director が新しい設定を適用していることを確認します。この例では、有効なハードウェア種別のリストを確認しています。

```
(undercloud) [stack@director ~]$ openstack baremetal driver list
+---------------------+---------------------+
| Supported driver(s) | Active host(s)      |
+---------------------+---------------------+
| idrac               | director.example.com |
| ipmi                | director.example.com |
| redfish             | director.example.com |
+---------------------+---------------------+
```

アンダークラウドの再設定が完了しました。

4.9. アンダークラウドのコンテナーレジストリー

Red Hat Enterprise Linux 9.2 には、Docker Registry v2 をインストールする **docker-distribution** パッケージが含まれなくなりました。互換性および同じ機能レベルを維持するために、director のインストールでは Apache Web サーバーおよび **image-serve** という仮想ホストが作成され、これによりレジストリーが提供されます。このレジストリーでも、SSL を無効にしたポート 8787/TCP が使用されます。Apache ベースのレジストリーはコンテナー化されていません。したがって、以下のコマンドを実行してレジストリーを再起動する必要があります。

```
$ sudo systemctl restart httpd
```

コンテナーレジストリーのログは、以下の場所に保存されます。

- /var/log/httpd/image_serve_access.log
- /var/log/httpd/image_serve_error.log.

イメージのコンテンツは、**/var/lib/image-serve** から提供されます。この場所では、レジストリー REST API のプル機能を実装するために、特定のディレクトリーレイアウトおよび **apache** 設定が使用されています。

Apache ベースのレジストリーでは、**podman push** コマンドも **buildah push** コマンドもサポートされません。つまり、従来の方法を使用してコンテナーイメージをプッシュすることはできません。デプロイ中にイメージを変更するには、**ContainerImagePrepare** パラメーターなどのコンテナー準備ワークフローを使用します。コンテナーイメージを管理するには、コンテナー管理コマンドを使用します。

openstack tripleo container image list
　レジストリーに保存されているすべてのイメージをリスト表示します。

openstack tripleo container image show
　レジストリーの特定イメージのメタデータを表示します。

openstack tripleo container image push
　イメージをリモートレジストリーからアンダークラウドレジストリーにプッシュします。

openstack tripleo container image delete
　レジストリーからイメージを削除します。

4.10. デフォルトのアンダークラウドディレクトリーの内容

RHOSP 17 では、すべての設定ファイルが 1 つのディレクトリーにあります。ディレクトリーの名前は、使用した openstack コマンドとスタックの名前を組み合わせたものです。ディレクトリーにはデフォルトの場所がありますが、**--working-dir** オプションを使用してデフォルトの場所を変更できます。このオプションは、デプロイメントで使用されるファイルの読み取りまたは作成を行う任意の **tripleoclient** コマンドで使用できます。

Default location	コマンド
$HOME/tripleo-deploy/undercloud	**tripleo deploy** に基づく **undercloud install**
$HOME/tripleo-deploy/<stack>	**tripleo deploy**、<stack> はデフォルトでは **standalone** です
$HOME/overcloud-deploy/<stack>	**overcloud deploy**、<stack> はデフォルトで **overcloud** です。

次の表は、**~/tripleo-deploy/undercloud** ディレクトリーに含まれるファイルとディレクトリーの詳細を示しています。

表4.1 アンダークラウドディレクトリーの内容

ディレクトリー	設定
heat-launcher	一時的な Heat 設定とデータベースのバックアップを含む一時的な Heat 作業ディレクトリー。

ディレクトリー	設定
install-undercloud.log	アンダークラウドのインストールおよびアップグレードのログ。
tripleo-ansible-inventory.yaml	オーバークラウドの Ansible インベントリー。
tripleo-config-generated-env-files	生成された環境ファイルが含まれます。
tripleo-undercloud-outputs.yaml	保存されたアンダークラウド出力が含まれます。
tripleo-undercloud-passwords.yaml	アンダークラウドのパスワードが含まれます。
undercloud-install-*.tar.bzip2	作業ディレクトリーの tarball (例: **undercloud-install-20220823210316.tar.bzip2**)。

第5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 (RHOSP) 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、ネットワークトポロジーのプランニング、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。

5.1. ノードロール

director には、オーバークラウドを作成するために、以下に示すデフォルトノード種別が含まれます。

Controller

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 (RHOSP) 環境に Controller ノードが 3 台必要です。

注記

1 台の Controller 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にのみ使用してください。2 台の Controller ノードまたは 4 台以上の Controller ノードで設定される環境はサポートされません。

Compute

ハイパーバイザーとして機能し、環境内で仮想マシンを実行するのに必要な処理機能を持つ物理サーバー。基本的な RHOSP 環境には少なくとも 1 つの Compute 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 Object Storage (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5.1 各種シナリオに使用するノードデプロイメントロール

	Controller	Compute	Ceph Storage	Swift Storage	Total
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

	Controller	Compute	Ceph Storage	Swift Storage	Total
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アーキテクチャーの詳細は、**Red Hat OpenStack Platform デプロイメントのカスタマイズ** ガイドの 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。

表5.2 概念検証用デプロイメントに使用するノードデプロイメントロール

	アンダークラウド	Controller	Compute	Ceph Storage	Total
概念実証	1	1	1	1	4

警告

Red Hat OpenStack Platform は、Day 2 操作中、稼働状態にある Ceph Storage クラスターを維持します。そのため、MON ノードまたはストレージノードの数が 3 未満のデプロイメントでは、一部の Day 2 操作 (Ceph Storage クラスターのアップグレードまたはマイナー更新等) を行うことができません。単一の Controller ノードまたは単一の Ceph Storage ノードを使用している場合は、Day 2 操作に失敗します。

5.2. オーバークラウドネットワーク

ロールとサービスを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トポロジー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 (RHOSP) 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 Openstack Networking (neutron) サービスを使用します。

デフォルトでは、director は接続に **プロビジョニング/コントロールプレーン** を使用するようにノードを設定します。ただし、ネットワークトラフィックを一連のコンポーザブルネットワークに分離し、カスタマイズしてサービスを割り当てることができます。

一般的な RHOSP 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ために、オーバークラウドは VLAN タグ付けを使用

して、それぞれ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独立したサブネットですが、一部のネットワークには、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。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 を提供する必要があります。

注記

VLAN を使用して、プロジェクト (tenant) ネットワークを作成します。プロジェクト VLAN を使用せずに、特殊用途ネットワーク用の Geneve トンネルを作成できます。Red Hat は、トンネリングを無効にして neutron VLAN モードでオーバークラウドをデプロイする場合でも、Geneve でトンネリングされたプロジェクトネットワークをデプロイすることを推奨します。Geneve でトンネリングされたプロジェクトネットワークをデプロイしても、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ように環境を更新できます。プロジェクト VLAN を使用してデプロイメントに Geneve 機能を追加することは可能ですが、中断を引き起こさずに既存のオーバークラウドにプロジェクト VLAN を追加することはできません。

director には、NIC を分離コンポーザブルネットワークと連携させるのに使用できるテンプレートセットも含まれています。デフォルトの設定は以下のとおりです。

- 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。
- ボンディングされた 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用に NIC を 1 つと、オーバークラウドネットワークの種別ごとのタグ付けされた VLAN 用にボンディング設定の 2 つの NIC。
- 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。

専用のテンプレートを作成して、特定の NIC 設定をマッピングすることもできます。

ネットワーク設定を検討する上で、以下の点を考慮することも重要です。

- 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。
- 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。
- director が各ノードの電源管理を制御できるように、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。
- 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。
- 外部のインターネットからインスタンスにアクセス可能でなければならない場合、パブリックネットワークから Floating IP アドレスを確保して、その Floating IP アドレスをインスタンスに割り当てることができます。インスタンスはプライベートの 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ます。したがって、そのテナントは必要に応じて Floating IP アドレスを特定のインスタンスに割り当てまたは割り当てを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るので、適切なセキュリティー確保手段に従う必要があります。

- あるブリッジのメンバーを単一のインターフェイスまたは単一のボンディングだけにすると、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。

- Red Hat では、オーバークラウドノードが Red Hat コンテンツ配信ネットワーク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、DNS によるホスト名解決を使用することを推奨します。

- Red Hat では、プロビジョニングインターフェイス、外部インターフェイス、Floating IP インターフェイスの MTU はデフォルトの 1500 のままにしておくことを推奨します。変更すると、接続性の問題が発生する可能性があります。これは、ルーターが通常レイヤー 3 の境界を超えてジャンボフレームでのデータを転送できないためです。

5.3. オーバークラウドのストレージ

オーバークラウド環境のバックエンドストレージとして Red Hat Ceph Storage ノードを使用できます。以下のタイプのストレージに Ceph ノードを使用するようにオーバークラウドを設定できます。

イメージ

Image サービス (glance) は、仮想マシンインスタンスの作成に使用されるイメージを管理します。イメージは不変のバイナリー BLOB です。Image サービスを使用して、イメージを Ceph Block Device に保存できます。サポートされているイメージ形式の詳細は、**イメージの作成と管理** の Image サービス (glance) を参照してください。

Volumes

Block Storage サービス (cinder) は、インスタンスの永続ストレージボリュームを管理します。Block Storage サービスボリュームはブロックデバイスです。ボリュームを使用してインスタンスを起動したり、実行中のインスタンスにボリュームをアタッチしたりできます。Block Storage サービスを使用して、イメージの Copy-on-Write クローンで仮想マシンをブートすることができます。

オブジェクト

Ceph Object Gateway (RGW) は、オーバークラウドストレージのバックエンドが Red Hat Ceph Storage である場合、Ceph クラスターにデフォルトのオーバークラウドオブジェクトストレージを提供します。オーバークラウドに Red Hat Ceph Storage がない場合、オーバークラウドは Object Storage サービス (swift) を使用して Object Storage を提供します。オーバークラウドノードを Object Storage サービス専用にすることができます。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。

ファイルシステム

Shared File Systems サービス (manila) は、共有ファイルシステムを管理します。Shared File Systems サービスを使用して、Ceph Storage ノード上のデータを使用して、CephFS ファイルシステムによってバックアップされた共有を管理できます。

インスタンスディスク

インスタンスを起動すると、インスタンスディスクはハイパーバイザーのインスタンスディレクトリーにファイルとして保存されます。デフォルトのファイルの場所は **/var/lib/nova/instances** です。

Ceph Storage の詳細は、Red Hat Ceph Storage アーキテクチャーガイド を参照してください。

5.3.1. オーバークラウドストレージの設定に関する考慮事項

ストレージ設定を計画するときは、次の問題を考慮してください。

インスタンスのセキュリティーとパフォーマンス

バックエンドの Block Storage ボリュームを使用するインスタンスで LVM を使用すると、パフォーマンス、ボリュームの可視性と可用性、およびデータの破損に関する問題が発生します。LVM フィルターを使用して問題を軽減します。詳細は、**永続ストレージの設定** の オーバークラウドノードでの LVM2 フィルターの有効化 および Red Hat ナレッジベースのソリューション Using LVM on a cinder volume exposes the data to the compute host を参照してください。

ローカルディスクのパーティションサイズ

デプロイメントのストレージと保持の要件を考慮して、次のデフォルトのディスクパーティションサイズが要件を満たしているかどうかを判断します。

パーティション	デフォルトのサイズ
/	8GB
/tmp	1 GB
/var/log	10 GB
/var/log/audit	2 GB
/home	1 GB
/var	ノードのロールに応じて: ● Object Storage ノード: 残りのディスクサイズの 10%。 ● コントローラーノード: 残りのディスクサイズの 90%。 ● Object Storage 以外のノード: 他のすべてのパーティションが割り当てられた後に、ディスクの残りのサイズが割り当てられます。
/srv/	Object Storage ノードの場合: 他のすべてのパーティションが割り当てられた後に、ディスクの残りのサイズが割り当てられます。

パーティションに割り当てられたディスクサイズを変更するには、**overcloud-baremetal-deploy.yaml** ノード定義ファイルの **ansible_playbooks** 定義内の **role_growvols_args** 追加の Ansible 変数を更新します。詳細は、Object Storage サービスのディスクパーティション全体の設定 を参照してください。

5.4. オーバークラウドのセキュリティー

OpenStack Platform の実装のセキュリティーレベルは、お使いの環境のセキュリティーレベルと同等でしかありません。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を正しく制御するようにします。

- 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およびパスワードの使用を徹底してください。
- 必ず SELinux を有効にします。

システムのセキュリティー保護の詳細は、以下の Red Hat ガイドを参照してください。

- Red Hat Enterprise Linux 9 の セキュリティーの強化
- Red Hat Enterprise Linux 9 の SELinux の使用

5.5. オーバークラウドの高可用性

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 Controller、Compute、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とサービスの詳細は、高可用性サービスの管理 を参照してください。

注記

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 を参照してください。

director を使用して、Compute 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。この高可用性のメカニズムにより、ノードで障害が発生すると Compute 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、環境をデプロイメント用に準備するために追加のステップを実施する必要があります。インスタンス HA とインストール手順の詳細は、インスタンスの高可用性の設定 ガイドを参照してください。

5.6. CONTROLLER ノードの要件

Controller ノードは、Red Hat OpenStack Platform 環境の中核となるサービス (例: Dashboard (horizon)、バックエンドのデータベースサーバー、Identity サービス (keystone) の認証、および高可用性サービスなど) をホストします。

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

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 (CPU コアの数をハイパースレッディングの値で乗算した数値に基づく) によって異なります。以下の計算により、RAM の要件を決定します。

- **Controller の最小メモリー容量の算出:**
  - 各仮想 CPU ごとに 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。
- **Controller の推奨メモリー容量の算出:**
  - 各仮想 CPU ごとに 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。

メモリーの要件の詳細は、Red Hat カスタマーポータルで Red Hat OpenStack Platform Hardware Requirements for Highly Available Controllers 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 Controller 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 および Object Storage サービスはいずれも Controller 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、市販の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。これらの環境は、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を使用すれば、最小限のプランニングでオーバークラウドをデプロイすることができますが、ワークロードキャパシティーとパフォーマンスの面ではあまり優れていません。

ただし、Telemetry が絶えずストレージにアクセスするため、エンタープライズ環境の場合、デフォルト設定では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 Controller 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。

ネットワークインターフェイスカード

最小 2 枚の 1 Gbps 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 Controller 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。

5.6.1. NUMA 使用時の制約

コンピュートサービス (nova) は、非均一メモリーアクセス (NUMA) トポロジーを持つすべての仮想マシン (VM) に対して厳密なメモリーアフィニティーを強制します。これは、NUMA 仮想マシンのメモリーが CPU と同じホスト NUMA ノードに関連付けられていることを意味します。NUMA と非 NUMA 仮想マシンを同じホスト上で実行しないでください。非 NUMA 仮想マシンがすでにホストを実行しており、そのホスト上で NUMA 仮想マシンがアフィニティーされて起動した場合、NUMA 仮想マシンはホストメモリーにアクセスできず、NUMA ノードに制限されるため、メモリー不足 (OOM) イベントが発生する可能性があります。OOM イベントを回避するには、すべての NUMA 関連インスタンスで NUMA 対応メモリートラッキングが有効になっていることを確認します。これを行うには、**hw:mem_page_size** フレーバーの追加仕様を設定します。

5.7. COMPUTE ノードの要件

Compute ノードは、仮想マシンインスタンスの起動後にそれを実行する役割を担います。Compute ノードには、ハードウェアの仮想化をサポートするベアメタルシステムが必要です。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。このプロセッサーには最小でも 4 つのコアが搭載されていることを推奨しています。

メモリー

ホストオペレーティングシステム用に最低 6GB の RAM と、次の考慮事項に対応するための追加メモリー。

- 仮想マシンインスタンスで使用できるようにするメモリーを追加します。
- メモリーを追加して、追加のカーネルモジュール、仮想スイッチ、モニターソリューション、その他の追加のバックグラウンドタスクなど、ホスト上で特別な機能や追加のリソースを実行します。
- Non-Uniform Memory Access (NUMA) を使用する場合、Red Hat は CPU ソケットノードあたり 8 GB、または 256 GB を超える物理 RAM がある場合はソケットノードあたり 16 GB を推奨します。
- 少なくとも 4 GB のスワップスペースを設定します。

ディスク容量

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 Compute 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。

5.8. RED HAT CEPH STORAGE ノードの要件

director を使用して Ceph Storage クラスターを作成するには、追加のノード要件があります。

- プロセッサー、メモリー、ネットワークインターフェイスカードの選択、ディスクレイアウトなどのハードウェア要件は、Red Hat Ceph Storage Hardware Guide で確認できます。
- 各 Ceph Storage ノードにも、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。
- 各 Ceph Storage ノードには、少なくとも 2 つのディスクが必要です。RHOSP director は **cephadm** を使用して Ceph Storage クラスターをデプロイします。cephadm 機能は、ノードのルートディスクへの Ceph OSD のインストールをサポートしていません。

5.8.1. Red Hat Ceph Storage ノードと RHEL の互換性

RHOSP 17.1 は RHEL 9.2 でサポートされています。ただし、Red Hat Ceph Storage ロールにマップされているホストは、最新のメジャー RHEL リリースに更新されます。アップグレードする前に、Red Hat ナレッジベースの記事 Red Hat Ceph Storage: Supported configurations を確認してください。

5.9. 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。Object Storage プロキシーは、Controller 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

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、最低でも 1 GB のメモリーを使用します。最適なパフォーマンスを得るには、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 (特に、ワークロードが 100 GB に満たないファイルの場合)。

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ー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ー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ーデータの 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

推奨されるノード設定には、以下の例に示す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**: コンテナー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

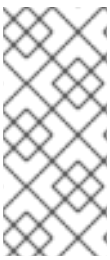
最小 2 枚の 1 Gbps ネットワークインターフェイスカード。タグ付けされた VLAN トラフィックを委譲する場合や、ボンディングインターフェイス向けには、追加のネットワークインターフェイスを使用します。

電源管理

各 Controller 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。

5.10. オーバークラウドのリポジトリー

Red Hat Enterprise Linux (RHEL) 9.2 で Red Hat OpenStack Platform (RHOSP) 17.1 を実行します。



注記

リポジトリーを Red Hat Satellite と同期する場合は、特定バージョンの Red Hat Enterprise Linux リポジトリーを有効にすることができます。ただし、選択したバージョンに関係なく、リポジトリーは同じままです。たとえば、BaseOS リポジトリーの 9.2 バージョンを有効にすることができますが、リポジトリー名は、選択した特定のバージョンに関係なく **rhel-9-for-x86_64-baseos-eus-rpms** のままになります。

警告

ここで指定されたもの以外のリポジトリーはサポートされていません。別途推奨されない限り、以下の表に記載されている以外の製品またはリポジトリーを有効にしないでください。有効にすると、パッケージの依存関係の問題が発生する可能性があります。Extra Packages for Enterprise Linux (EPEL) を有効にしないでください。

Controller ノード用リポジトリー

以下の表には、オーバークラウドの Controller ノード用コアリポジトリーをまとめています。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-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ー
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux の高可用性ツール。
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform のコアリポジトリー
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	OpenStack Platform 用 Open vSwitch (OVS) パッケージを提供します。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-rpms**	Red Hat Ceph Storage 6 for Red Hat Enterprise Linux 9 のツール

Compute ノードおよび ComputeHCI ノードのリポジトリー

以下の表に、オーバークラウド内の Compute ノードおよび ComputeHCI ノードのコアリポジトリーを示します。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-baseos-eus-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ー

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-eus-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 9 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-9-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux の高可用性ツール。
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	Red Hat OpenStack Platform のコアリポジトリー
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	OpenStack Platform 用 Open vSwitch (OVS) パッケージを提供します。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-rpms**	Red Hat Ceph Storage 6 for Red Hat Enterprise Linux 9 のツール

Ceph Storage ノード用リポジトリー

以下の表には、オーバークラウド用の Ceph Storage 関連のリポジトリーをまとめています。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)	**rhel-9-for-x86_64-baseos-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ー
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)	**rhel-9-for-x86_64-appstream-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat OpenStack Platform Deployment Tools for RHEL 9 x86_64 (RPMs)	**openstack-17.1-deployment-tools-for-rhel-9-x86_64-rpms**	director が Ceph Storage ノードを設定するのに役立つパッケージ。このリポジトリーは、スタンドアロンの Ceph Storage サブスクリプションに含まれています。OpenStack Platform と Ceph Storage サブスクリプションを組み合わせて使用する場合は、**openstack-17.1-for-rhel-9-x86_64-rpms** リポジトリーを使用します。

名前	リポジトリー	要件の説明
Red Hat OpenStack Platform for RHEL 9 x86_64 (RPMs)	**openstack-17.1-for-rhel-9-x86_64-rpms**	director が Ceph Storage ノードを設定するのに役立つパッケージ。このリポジトリーは、Red Hat OpenStack Platform と Red Hat Ceph Storage を組み合わせたサブスクリプションに含まれています。スタンドアロンの Red Hat Ceph Storage サブスクリプションを使用する場合は、**openstack-17.1-deployment-tools-for-rhel-9-x86_64-rpms** リポジトリーを使用します。
Red Hat Ceph Storage Tools 6 for RHEL 9 x86_64 (RPMs)	**rhceph-6-tools-for-rhel-9-x86_64-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。
Red Hat Fast Datapath for RHEL 9 (RPMS)	**fast-datapath-for-rhel-9-x86_64-rpms**	OpenStack Platform 用 Open vSwitch (OVS) パッケージを提供します。Ceph Storage ノードで OVS を使用している場合は、このリポジトリーをネットワークインターフェイス設定 (NIC) テンプレートに追加します。

5.11. ノードのプロビジョニングと設定

OpenStack Bare Metal (ironic) サービスまたは外部ツールを使用して、Red Hat OpenStack Platform (RHOSP) 環境のオーバークラウドノードをプロビジョニングします。ノードをプロビジョニングしたら、director を使用してノードを設定します。

OpenStack Bare Metal (ironic) サービスを使用したプロビジョニング

Bare Metal サービスを使用したオーバークラウドノードのプロビジョニングは、標準的なプロビジョニング方法です。詳細は、ベアメタルオーバークラウドノードのプロビジョニング を参照してください。

外部ツールを使用したプロビジョニング

Red Hat Satellite などの外部ツールを使用して、オーバークラウドノードをプロビジョニングできます。この方法は、電源管理制御を設定せずに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、あるいは **overcloud-hardened-uefi-full.qcow2** イメージに依存しないカスタムのパーティションレイアウトを持つノードを使用する場合に便利です。このプロビジョニング方法は、ノードの管理に OpenStack Bare Metal サービス (ironic) を使用しません。詳細は、Configuring a basic overcloud with pre-provisioned nodes を参照してください。

第6章 オーバークラウドネットワークの設定

オーバークラウドの物理ネットワークを設定するには、ネットワークデータスキーマで定義された構造に準拠する、ネットワーク設定ファイル **network_data.yaml** を作成します。

6.1. オーバークラウドネットワークの定義

Red Hat OpenStack Platform (RHOSP) は、特定のタイプのネットワークトラフィックを分離してホストするために使用できるデフォルトのオーバークラウドネットワークを提供します。ネットワークが分離設定されていない場合には、RHOSP はすべての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。

以下の分離されたオーバークラウドネットワークを定義できます。

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。

Provisioning

Director は、このネットワークをデプロイメントと管理に使用します。プロビジョニングネットワークは通常、専用インターフェイスで設定されます。最初のデプロイメントでは PXE で DHCP を使用し、ネットワークは静的 IP に変換されます。デフォルトでは、ネイティブ VLAN で PXE ブートを実行する必要がありますが、一部のシステムコントローラーでは VLAN からブートできます。デフォルトでは、コンピュートノードとストレージノードはプロビジョニングインターフェイスを DNS、NTP、およびシステムメンテナンス用のデフォルトゲートウェイとして使用します。
アンダークラウドは、デフォルトゲートウェイとして使用できます。ただし、すべてのトラフィックは IP マスカレード NAT (ネットワークアドレス変換) の背後にあり、残りの RHOSP ネットワークからは到達できません。アンダークラウドは、オーバークラウドのデフォルトルートの単一障害点でもあります。プロビジョニングネットワーク上のルーターデバイスに外部ゲートウェイが設定されている場合には、代わりにアンダークラウドの neutron DHCP サーバーがそのサービスを提供できます。

Internal API

Internal 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 RHOSP のサービス間の通信を行う際に使用します。

Tenant

Networking サービス (neutron) は、次のいずれかの方法を使用して、各テナント (プロジェクト) に独自のネットワークを提供します。

- 各テナントネットワークがネットワーク VLAN である、VLAN 分離。
- VXLAN または GRE を介したトンネリング。

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離されます。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。

ストレージ

Block Storage、NFS、iSCSI などに使用されるネットワーク。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。

Storage Management

Object Storage サービス (swift) は、このネットワークを使用して、参加してい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。プロキシーサービスは、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。プロキシーは、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。Red Hat Ceph Storage バックエンドを使用するサービスは、Red Hat Ceph Storage と直接対話せずにフロントエンドサービスを使用するため、Storage Management ネットワーク経由で接続します。RBD ドライバーは例外で、このトラフィックは Red Hat Ceph Storage に直接接続されます。

External

グラフィカルシステム管理用の Dashboard サービス (horizon)、RHOSP 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。

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。External 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 Networking Service (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、Networking Service (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。

注記

プロビジョニングネットワークはネイティブ VLAN である必要があり、他のネットワークはトランキングできます。

ロールごとに特定の分離ネットワークを定義する必要があります。

Role	ネットワーク
Controller	プロビジョニング、Internal API、Storage、Storage Management、Tenant、External
Compute	プロビジョニング、Internal API、Storage、Tenant
Ceph Storage	プロビジョニング、Internal API、Storage、Storage Management
Cinder Storage	プロビジョニング、Internal API、Storage、Storage Management
Swift Storage	プロビジョニング、Internal API、Storage、Storage Management

6.2. ネットワーク定義ファイルの作成

分離されたオーバークラウドネットワークを設定するためのネットワーク定義ファイルを作成します。

手順

1. 必要なサンプルネットワーク定義テンプレートを **/usr/share/openstack-tripleo-heat-templates/network-data-samples** から環境ファイルディレクトリーにコピーします。

```
$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/<sample_network_definition_file> /home/stack/templates/<networks_definition_file>
```

- **<sample_network_setting_file>** を、コピーするサンプルネットワーク定義ファイルの名前 (例: **default-network-isolation-ipv6.yaml)** に置き換えます。
- **<networks_definition_file>** は、ネットワーク定義ファイルの名前 (**network_data.yaml** など) に置き換えます。

2. オーバークラウドネットワーク環境の要件に合わせて、ローカルネットワーク定義ファイルのネットワークとネットワーク属性を更新します。ネットワーク定義ファイルでネットワーク属性を設定するために使用できるプロパティーについては、ネットワーク定義ファイルの設定オプション を参照してください。ノード定義ファイルの例は、ネットワーク定義ファイルの例 を参照してください。

3. オプション: ユーザーに表示されるデフォルトネットワークの名前を変更するには、**name_lower** フィールドをネットワークの新しい名前に変更し、**ServiceNetMap** を新しい名前で更新します。

```
- name: InternalApi
  name_lower: <custom_name>
  service_net_map_replace: <default_name>
```

- **name** フィールドは変更しないでください。
- **<custom_name>** を、ネットワークに割り当てる新しい名前 (**MyCustomInternalApi** など) に置き換えます。
- **<default_name>** を、**name_lower** パラメーターのデフォルト値 (**internal_api** など) に置き換えます。

4. オプション: カスタムネットワークをネットワーク定義ファイルに追加します。以下の例では、ストレージバックアップ用のネットワークを追加します。

```
- name: StorageBackup
  vip: false
  name_lower: storage_backup
  subnets:
    storage_backup_subnet:
      ip_subnet: 172.16.6.0/24
      allocation_pools:
        - start: 172.16.6.4
        - end: 172.16.6.250
      gateway_ip: 172.16.6.1
```

6.3. ネットワーク VIP 定義ファイルの作成

オーバークラウドのネットワーク VIP を設定するために、ネットワーク仮想 IP (VIP) 定義ファイルを作成します。

手順

1. 必要なサンプルネットワーク VIP 定義テンプレートを **/usr/share/openstack-tripleo-heat-templates/network-data-samples** から環境ファイルディレクトリーにコピーします。

```
$ cp /usr/share/openstack-tripleo-heat-templates/network-data-
samples/<sample_network_VIP_definition_file>
/home/stack/templates/<network_VIP_definition_file>
```

- **<sample_network_VIP_definition_file>** を、コピーするサンプルネットワーク仮想 IP 定義ファイルの名前 (**vip-data-default-network-isolation.yaml** など) に置き換えます。
- **<network_VIP_definition_file>** を、ネットワーク仮想 IP 定義ファイルの名前 (**vip_data.yaml** など) に置き換えます。

2. オプション: 環境に合わせてネットワーク仮想 IP 定義ファイルを設定します。

```
- network: <network>
  dns_name: <dns_name>
  name: <vip_name>
  ip_address: <vip_address>
  subnet: <subnet>
```

- **<network>** を、IP アドレスが割り当てられているネットワークの名前 (**internal_api** や **storage** など) に置き換えます。
- オプション: **<dns_name>** を、FQDN (完全修飾ドメイン名) (**overcloud** など) に置き換えます。
- オプション: **<vip_name>** を、仮想 IP 名に置き換えます。
- オプション: **<vip_address>** をネットワークの仮想 IP アドレスに置き換えます。
- オプション: **<subnet>** を、仮想 IP ポートの作成時に使用するサブネットの名前に置き換えます。ルーティングされたネットワークに必要です。
  たとえば、次の設定では、外部ネットワークとコントロールプレーン仮想 IP を定義します。

```
- network: external
  dns_name: overcloud
  ip_address: '1.2.3.4'
- network: ctlplane
  dns_name: overcloud
```

ネットワーク VIP の設定に使用できるプロパティーの詳細は、ネットワーク仮想 IP 属性のプロパティー を参照してください。

6.4. ネットワーク定義ファイルの設定オプション

次の表に記載されたプロパティーを使用して、**network_data.yaml** ファイル内のネットワーク属性を設定できます。

表6.1 ネットワーク定義のプロパティー

プロパティー	Value
name	ネットワークの名前。
name_lower	ネットワーク名の小文字バージョン。Director は、この名前を **roles_data.yaml** ファイル内のロールに割り当てられた各ネットワークにマップします。デフォルト値は **name.lower()** です。

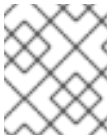
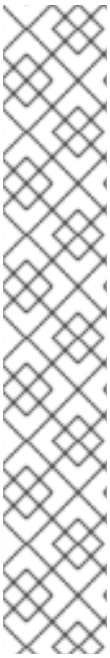
プロパティー	Value
dns_domain	ネットワークの DNS ドメイン名。アンダークラウドノードが複数のオーバークラウドをデプロイおよび管理する場合に限り、**dns_domain** を設定します。 **dns_domain** の値を確認するには、以下の手順を行います。 ● ネットワーク名を小文字に変換します。 ● ネットワーク名内のすべてのアンダースコア ("_") をハイフン ("-") に置き換えます。 ● ネットワーク名の末尾にドット (".") と **CloudDomain** の名前を追加します。 以下に例を示します。 ● **network.name** = **InternalApi** ● **CloudDomain** = **cloud.example.com.** ● **dns_domain** = **internalapi.cloud.example.com.**  **注記** 異なるネットワークに同じ **dns_domain** を使用することはできません。
mtu	最大伝送単位 (MTU)。デフォルト値は **1500** です。
ipv6	IPv6 の場合は **true** に設定します。デフォルト値は **false** です。
vip	ネットワーク上に VIP を作成するには、**true** に設定します。デフォルト値は **false** です。
subnets	ネットワークのサブネット定義が含まれています。

表6.2 サブネット定義のプロパティー

プロパティー	Value
subnet_name	サブネットの名前。
ip_subnet	CIDR ブロック表記の IPv4 サブネット。デフォルト値は **192.0.5.0/24** です。
ipv6_subnet	CIDR ブロック表記の IPv6 サブネット。デフォルト値は **2001:db6:fd00:1000::/64** です。

プロパティー	Value
gateway_ip	IPv4 ネットワークのゲートウェイアドレス。デフォルト値は **192.0.5.1** です。
gateway_ipv6	IPv6 ネットワークのゲートウェイ。
allocation_pools	IPv4 サブネットの IP 範囲。デフォルト値: start: 192.0.5.100 end: 192.0.5.150
ipv6_allocation_pools	IPv6 サブネットの IP 範囲。デフォルト値: start: 2001:db6:fd00:1000:100::1 end: 2001:db6:fd00:1000:150::1
routes	ネットワークゲートウェイ経由のルーティングが必要な IPv4 ネットワークのリスト。
routes_ipv6	ネットワークゲートウェイ経由のルーティングが必要な IPv6 ネットワークのリスト。
vlan	ネットワークの VLAN ID。



注記

routes および **routes_ipv6** オプションには、ルートのリストが含まれています。各ルートは、**destination** と **nexthop** キーを含むディクショナリーエントリーです。どちらのオプションも文字列型です。

```
routes:
  - destination: 198.51.100.0/24
    nexthop: 192.0.5.1
  - destination: 203.0.113.0/24
    nexthost: 192.0.5.1
```

```
routes:
  - destination: 2001:db6:fd00:2000::/64
    nexthop: 2001:db6:fd00:1000:100::1
  - destination: 2001:db6:fd00:3000::/64
    nexthost: 2001:db6:fd00:1000:100::1
```

6.5. ネットワーク VIP 属性のプロパティー

次のプロパティーを使用して、**network_data.yaml** ファイルでネットワーク VIP 属性を設定できます。

表6.3 ネットワーク VIP 属性のプロパティー

プロパティー	Value
name	仮想 IP 名。デフォルト値は **$network_name_virtual_ip** です。
network	(必須) ネットワーク名。
ip_address	VIP の固定 IP アドレス。
subnet	サブネット名。仮想 IP ポートのサブネットを指定します。ルーティングされたネットワークを使用するデプロイメントに必要です。
dns_name	FQDN (完全修飾ドメイン名)。デフォルト値は **overcloud** です。

6.6. ネットワーク定義ファイルの例

以下のネットワーク定義ファイルの例では、ストレージ、ストレージ管理、内部 API、テナント、および外部ネットワークを設定します。

```
- name: Storage
  name_lower: storage
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    storage_subnet:
      ipv6_subnet: fd00:fd00:fd00:3000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:3000::10
          end: fd00:fd00:fd00:3000:ffff:ffff:ffff:fffe
      vlan: 30
- name: StorageMgmt
  name_lower: storage_mgmt
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    storage_mgmt_subnet:
      ipv6_subnet: fd00:fd00:fd00:4000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:4000::10
          end: fd00:fd00:fd00:4000:ffff:ffff:ffff:fffe
      vlan: 40
- name: InternalApi
  name_lower: internal_api
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    internal_api_subnet:
      ipv6_subnet: fd00:fd00:fd00:2000::/64
      ipv6_allocation_pools:
```

```
      - start: fd00:fd00:fd00:2000::10
        end: fd00:fd00:fd00:2000:ffff:ffff:ffff:fffe
    vlan: 20
- name: Tenant
  name_lower: tenant
  vip: false # Tenant networks do not use VIPs
  ipv6: true
  mtu: 1500
  subnets:
    tenant_subnet:
      ipv6_subnet: fd00:fd00:fd00:5000::/64
      ipv6_allocation_pools:
        - start: fd00:fd00:fd00:5000::10
          end: fd00:fd00:fd00:5000:ffff:ffff:ffff:fffe
      vlan: 50
- name: External
  name_lower: external
  vip: true
  ipv6: true
  mtu: 1500
  subnets:
    external_subnet:
      ipv6_subnet: 2001:db8:fd00:1000::/64
      ipv6_allocation_pools:
        - start: 2001:db8:fd00:1000::10
          end: 2001:db8:fd00:1000:ffff:ffff:ffff:fffe
      gateway_ipv6: 2001:db8:fd00:1000::1
      vlan: 10
```

第7章 オーバークラウドのプロビジョニングとデプロイ

オーバークラウドを作成するには、以下のタスクを実行する必要があります。

1. 物理ネットワークのネットワークリソースをプロビジョニングします。
    a. ネットワークの分離またはカスタム設定可能ネットワークをデプロイする場合は、ネットワーク定義ファイルを YAML 形式で作成します。
    b. ネットワーク定義ファイルを含め、ネットワークプロビジョニングコマンドを実行します。
    c. YAML 形式でネットワーク仮想 IP (VIP) 定義ファイルを作成します。
    d. ネットワーク VIP 定義ファイルを含め、ネットワーク VIP プロビジョニングコマンドを実行します。
2. ベアメタルノードをプロビジョニングします。
    a. ノード定義ファイルを YAML 形式で作成します。
    b. ノード定義ファイルを含め、ベアメタルノードプロビジョニングコマンドを実行します。
3. オーバークラウドをデプロイします。
    a. プロビジョニングコマンドにより生成される heat 環境ファイルを指定して、デプロイメントコマンドを実行します。

7.1. オーバークラウドネットワークのプロビジョニング

Red Hat OpenStack Platform (RHOSP) 物理ネットワーク環境のネットワークリソースを設定するには、次のタスクを実行する必要があります。

1. オーバークラウドのネットワークリソースを設定およびプロビジョニングします。
2. オーバークラウドのネットワーク仮想 IP を設定およびプロビジョニングします。

7.1.1. オーバークラウドのネットワーク定義の設定とプロビジョニング

YAML 形式のネットワーク定義ファイルで、オーバークラウドの物理ネットワークを設定します。プロビジョニングプロセスでは、ネットワーク仕様を含むネットワーク定義ファイルから heat 環境ファイルが作成されます。オーバークラウドをデプロイするときに、デプロイメントコマンドにこの heat 環境ファイルを含めます。

前提条件

- アンダークラウドがインストールされます。詳細は、Installing director を参照してください。

手順

1. source コマンドで **stackrc** アンダークラウド認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

2. 必要なサンプルネットワーク定義テンプレートを **/usr/share/openstack-tripleo-heat-templates/network-data-samples** から環境ファイルディレクトリーにコピーします。

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/default-network-isolation.yaml /home/stack/templates/network_data.yaml
```

3. ネットワーク環境に合わせてネットワーク定義ファイルを設定します。たとえば、外部ネットワーク定義を更新できます。

```
- name: External
  name_lower: external
  vip: true
  mtu: 1500
  subnets:
    external_subnet:
      ip_subnet: 10.0.0.0/24
      allocation_pools:
        - start: 10.0.0.4
          end: 10.0.0.250
      gateway_ip: 10.0.0.1
      vlan: 10
```

4. 環境のその他のネットワークとネットワーク属性を設定します。ネットワーク定義ファイルでネットワーク属性の設定に使用できるプロパティーの詳細は、オーバークラウドネットワークの設定 を参照してください。

5. オーバークラウドネットワークをプロビジョニングします。

```
(undercloud)$ openstack overcloud network provision \
 [--templates <templates_directory> \]
 --output  <deployment_file> \
 /home/stack/templates/<networks_definition_file>
```

- オプション: **--templates** オプションを含めて、**/usr/share/openstack-tripleo-heat-templates** にあるデフォルトテンプレートの代わりに独自のテンプレートを使用します。**<templates_directory>** は、テンプレートを含むディレクトリーへのパスに置き換えます。

- **<deployment_file>** は、デプロイメントコマンドに追加するために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-networks-deployed.yaml)**。

- **<networks_definition_file>** は、ネットワーク定義ファイルの名前 (**network_data.yaml** など) に置き換えます。

6. ネットワークのプロビジョニングが完了したら、次のコマンドを使用して、作成されたネットワークとサブネットを確認できます。

```
(undercloud)$ openstack network list
(undercloud)$ openstack subnet list
(undercloud)$ openstack network show <network>
(undercloud)$ openstack subnet show <subnet>
```

- **<network>** は、確認するネットワークの名前または UUID に置き換えます。

- **<subnet>** は、確認するサブネットの名前または UUID に置き換えます。

次のステップ

- オーバークラウドのネットワーク VIP の設定とプロビジョニング

7.1.2. オーバークラウドのネットワーク VIP の設定とプロビジョニング

YAML 形式のネットワーク VIP 定義ファイルで、オーバークラウドのネットワーク仮想 IP (VIP) を設定します。プロビジョニングプロセスでは、VIP 仕様を含む VIP 定義ファイルから Heat 環境ファイルが作成されます。オーバークラウドをデプロイするときに、デプロイメントコマンドにこの heat 環境ファイルを含めます。

前提条件

- アンダークラウドがインストールされます。詳細は、Installing director を参照してください。
- オーバークラウドネットワークがプロビジョニングされます。詳細は、オーバークラウドのネットワーク定義の設定とプロビジョニング を参照してください。

手順

1. source コマンドで **stackrc** アンダークラウド認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

2. 必要なサンプルネットワーク VIP 定義テンプレートを **/usr/share/openstack-tripleo-heat-templates/network-data-samples** から環境ファイルディレクトリーにコピーします。

```
(undercloud)$ cp /usr/share/openstack-tripleo-heat-templates/network-data-samples/vip-data-default-network-isolation.yaml /home/stack/templates/vip_data.yaml
```

3. オプション: 環境に合わせて VIP 定義ファイルを設定します。たとえば、次の例では、外部ネットワークとコントロールプレーンの VIP を定義しています。

```
- name: external_vip
  network: external
  ip_address: 10.0.0.0
  subnet: external_vip_subnet
  dns_name: overcloud
- name: ctlplane_vip
  network: ctlplane
  ip_address: 192.168.122.0
  subnet: ctlplane_vip_subnet
  dns_name: overcloud
```

VIP 定義ファイルでネットワーク VIP 属性を設定するために使用できるプロパティーの詳細は、ネットワーク VIP 属性のプロパティー を参照してください。

4. ネットワーク VIP をプロビジョニングします。

```
(undercloud)$ openstack overcloud network vip provision \
 [--templates <templates_directory> \]
 --stack <stack> \
```

```
--output <deployment_file> \
/home/stack/templates/<vip_definition_file>
```

- オプション: **--templates** オプションを含めて、**/usr/share/openstack-tripleo-heat-templates** にあるデフォルトテンプレートの代わりに独自のテンプレートを使用します。**<templates_directory>** は、テンプレートを含むディレクトリーへのパスに置き換えます。
- **<stack>** は、ネットワーク VIP がプロビジョニングされているスタックの名前 (**overcloud** など) に置き換えます。
- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-vip-deployed.yaml)**。
- **<vip_definition_file>** は VIP 定義ファイルの名前 (**vip_data.yaml** など) に置き換えます。

5. ネットワーク VIP のプロビジョニングが完了したら、次のコマンドを使用して、作成された VIP を確認できます。

```
(undercloud)$ openstack port list
(undercloud)$ openstack port show <port>
```

- **<port>** は、確認するポートの名前または UUID に置き換えます。

次のステップ

- ベアメタルオーバークラウドノードのプロビジョニング

7.2. ベアメタルオーバークラウドノードのプロビジョニング

Red Hat OpenStack Platform (RHOSP) 環境を設定するには、次のタスクを実行する必要があります。

1. オーバークラウドのベアメタルノードを登録します。
2. ベアメタルノードのハードウェアのインベントリーをディレクターに提供します。
3. ノード定義ファイルで、ベアメタルノードの数量、属性、およびネットワークレイアウトを設定します。
4. ベアメタルノードに、指定のノードとこのノードを合致させるリソースクラスを割り当てます。

プロファイルを一致させてオーバークラウドノードを指定するなど、追加のオプションのタスクを実行することもできます。

7.2.1. オーバークラウドノードの登録

ディレクターには、ノードのハードウェアと電源管理の詳細を指定するノード定義テンプレートが必要です。このテンプレートは、JSON 形式の **nodes.json** または YAML 形式の **nodes.yaml** で作成できます。

手順

1. ノードをリスト表示する **nodes.json** または **nodes.yaml** という名前のテンプレートを作成します。以下の例に示す JSON および YAML テンプレートを使用して、ノード定義のテンプレートを設定する方法を説明します。

JSON テンプレートの例

```
{
  "nodes": [{
    "name": "node01",
    "ports": [{
      "address": "aa:aa:aa:aa:aa:aa",
      "physical_network": "ctlplane",
      "local_link_connection": {
        "switch_id": "52:54:00:00:00:00",
        "port_id": "p0"
      }
    }],
    "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"
  },
  {
    "name": "node02",
    "ports": [{
      "address": "bb:bb:bb:bb:bb:bb",
      "physical_network": "ctlplane",
      "local_link_connection": {
        "switch_id": "52:54:00:00:00:00",
        "port_id": "p0"
      }
    }],
    "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"
  }]
}
```

YAML テンプレートの例

```
nodes:
  - name: "node01"
    ports:
      - address: "aa:aa:aa:aa:aa:aa"
        physical_network: ctlplane
        local_link_connection:
```

```
        switch_id: "52:54:00:00:00:00"
        port_id: p0
    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"
  - name: "node02"
    ports:
      - address: "bb:bb:bb:bb:bb:bb"
        physical_network: ctlplane
        local_link_connection:
          switch_id: "52:54:00:00:00:00"
          port_id: p0
    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"
```

このテンプレートには、以下の属性が含まれます。

name

ノードの論理名

ports

特定の IPMI デバイスにアクセスするためのポート次の任意のポート属性を定義できます。

- **address**: 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。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。
- **physical_network**: プロビジョニング NIC に接続されている物理ネットワーク。
- **local_link_connection**: IPv6 プロビジョニングを使用し、イントロスペクション中に LLDP がローカルリンク接続を正しく反映しない場合は、**local_link_connection** パラメーターの **switch_id** および **port_id** フィールドにダミーのデータを含める必要があります。偽のデータを含める方法の詳細は、director イントロスペクションを使用したベアメタルノードのハードウェア情報の収集 を参照してください。

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

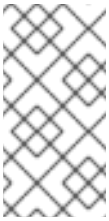
(オプション) ハードディスクのサイズ (GB 単位)

arch

(オプション) システムアーキテクチャー

pm_type

使用する電源管理ドライバー。この例では IPMI ドライバー (**ipmi**) を使用しています。



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。サポートされている電源管理の種類とそのオプションの詳細は、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

pm_addr

IPMI デバイスの IP アドレス

2. テンプレートのフォーマットと構文を確認します。

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

3. テンプレートファイルを**stack** ユーザーのホームディレクトリー (**/home/stack/nodes.json**) に保存します。

4. テンプレートを director にインポートして、各ノードをテンプレートから director に登録します。

```
(undercloud)$ openstack overcloud node import ~/nodes.json
```

5. ノードの登録および設定が完了するまで待ちます。完了したら、ノードが director に正しく登録されていることを確認します。

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

7.2.2. ベアメタルノードハードウェアのインベントリーの作成

ディレクターは、プロファイルのタグ付け、ベンチマーク、および手動のルートディスク割り当てのために、Red Hat OpenStack Platform (RHOSP) デプロイメント内のノードのハードウェアインベントリーを必要とします。

次のいずれかの方法を使用して、ハードウェアインベントリーをディレクターに提供できます。

- Automatic: 各ノードからハードウェア情報を収集するディレクターのイントロスペクションプロセスを使用できます。このプロセスは、各ノードでイントロスペクションエージェントを起動します。イントロスペクションエージェントは、ノードからハードウェアのデータを収集し、そのデータを director に送り返します。Director は、ハードウェアデータを OpenStack 内部データベースに保存します。

- Manual: 各ベアメタルマシンの基本的なハードウェアインベントリーを手動で設定できます。このインベントリーは、ベアメタルプロビジョニングサービス (ironic) に保存され、ベアメタルマシンの管理とデプロイに使用されます。

ディレクターの自動イントロスペクションプロセスには、ベアメタルプロビジョニングサービスポートを手動で設定する方法に比べて、次の利点があります。

- イントロスペクションは、接続されているすべてのポートをハードウェア情報に記録します。これには、**nodes.yaml** でまだ設定されていない場合に PXE ブートに使用するポートも含まれます。
- イントロスペクションは、属性が LLDP を使用して検出可能である場合、各ポートの **local_link_connection** 属性を設定します。手動による方法を使用する場合は、ノードを登録するときに各ポートに **local_link_connection** を設定する必要があります。
- イントロスペクションは、スパイン/リーフ型または DCN のアーキテクチャーをデプロイするときに、ベアメタルプロビジョニングサービスポートの **physical_network** 属性を設定します。

7.2.2.1. ディレクターのイントロスペクションを使用してベアメタルノードのハードウェア情報を収集する

物理マシンをベアメタルノードとして登録した後、ディレクターイントロスペクションを使用して、ハードウェアの詳細を自動的に追加し、イーサネット MAC アドレスごとにポートを作成できます。

ヒント

自動イントロスペクションの代わりに、ベアメタルノードのハードウェア情報をディレクターに手動で提供できます。詳細は、ベアメタルノードのハードウェア情報の手動設定 を参照してください。

前提条件

- オーバークラウドのベアメタルノードを登録しました。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. pre-introspection 検証グループを実行して、プリイントロスペクションの要件を確認します。

```
(undercloud)$ validation run --group pre-introspection \
 --inventory <inventory_file>
```

   - **<inventory_file>** ファイルを Ansible インベントリーファイルの名前および場所に置き換えます (例: **~/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml**)。

注記

検証を実行すると、出力の **Reasons** 列は 79 文字に制限されます。検証結果を完全に表示するには、検証ログファイルを表示します。

4. 検証レポートの結果を確認します。
5. オプション: 特定の検証からの詳細な出力を確認します。

```
(undercloud)$ validation history get --full <UUID>
```

- **<UUID>** は、確認するレポートの特定の検証の UUID に置き換えます。

重要

検証結果が **FAILED** であっても、Red Hat OpenStack Platform のデプロイや実行が妨げられることはありません。ただし、**FAILED** の検証結果は、実稼働環境で問題が発生する可能性があることを意味します。

6. 各ノードのハードウェア属性を検証します。すべてのノードまたは特定のノードのハードウェア属性を検査できます。
   - すべてのノードのハードウェア属性を検査します。

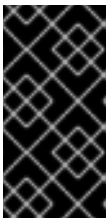
     ```
     (undercloud)$ openstack overcloud node introspect --all-manageable --provide
     ```

     - **--all-manageable** オプションを使用して、管理状態にあるノードのみをイントロスペクションします。ここでは、すべてのノードが管理状態にあります。
     - **--provide** オプションを使用して、イントロスペクション後に全ノードを **available** の状態に再設定します。
   - 特定のノードのハードウェア属性を検査します。

     ```
     (undercloud)$ openstack overcloud node introspect --provide <node1> [node2] [noden]
     ```

     - **--provide** オプションを使用して、イントロスペクション後に指定されたすべてのノードを **available** 状態にリセットします。
     - **<node1>**、**[node2]**、および **[noden]** までのすべてのノードを、イントロスペクションする各ノードの UUID に置き換えます。
7. 別のターミナルウィンドウで、イントロスペクションの進捗ログを監視します。

   ```
   (undercloud)$ sudo tail -f /var/log/containers/ironic-inspector/ironic-inspector.log
   ```



重要

イントロスペクションプロセスが完了するまで実行されていることを確認します。イントロスペクションは通常、ベアメタルノードの場合 15 分かかります。ただし、イントロスペクションネットワークのサイズが正しくないと、時間がかかる可能性があり、イントロスペクションが失敗する可能性があります。

8. オプション: IPv6 を介したベアメタルプロビジョニング用にアンダークラウドを設定した場合は、LLDP がベアメタルプロビジョニングサービス (ironic) ポートの **local_link_connection** を設定していることも確認する必要があります。

   ```
   $ openstack baremetal port list --long -c UUID -c "Node UUID" -c "Local Link Connection"
   ```

   - ベアメタルノードのポートに対して Local Link Connection フィールドが空の場合、**local_link_connection** 値に偽のデータを手動で入力する必要があります。次の例では、偽のスイッチ ID を **52:54:00:00:00:00** に設定し、偽のポート ID を **p0** に設定します。

```
$ openstack baremetal port set <port_uuid> \
--local-link-connection switch_id=52:54:00:00:00:00 \
--local-link-connection port_id=p0
```

- ローカルリンク接続フィールドにダミーのデータが含まれていることを確認します。

```
$ openstack baremetal port list --long -c UUID -c "Node UUID" -c "Local Link Connection"
```

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。

7.2.2.2. ベアメタルノードのハードウェア情報を手動で設定する

物理マシンをベアメタルノードとして登録した後、ハードウェアの詳細を手動で追加し、イーサネット MAC アドレスごとにベアメタルポートを作成できます。オーバークラウドをデプロイする前に、少なくとも 1 つのベアメタルポートを作成する必要があります。

ヒント

手動イントロスペクションの代わりに、自動ディレクターイントロスペクションプロセスを使用して、ベアメタルノードのハードウェア情報を収集できます。詳細は、director イントロスペクションを使用したベアメタルノードのハードウェア情報の収集 を参照してください。

前提条件

- オーバークラウドのベアメタルノードを登録しました。
- **nodes.json** の登録済みノードの各ポートに **local_link_connection** を設定しました。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。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

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

```
(undercloud)$ openstack baremetal node set <node> \
  --driver-info deploy_kernel=<kernel_file> \
  --driver-info deploy_ramdisk=<initramfs_file>
```

- **<node>** をベアメタルノードの ID に置き換えてください。
- **<kernel_file>** を **.kernel** イメージへのパス (例: **file:///var/lib/ironic/httpboot/agent.kernel**) に置き換えます。
- **<initramfs_file>** は、**.initramfs** イメージへのパス (例: **file:///var/lib/ironic/httpboot/agent.ramdisk**) に置き換えます。

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

```
(undercloud)$ openstack baremetal node set <node> \
```

```
--property cpus=<cpu> \
--property memory_mb=<ram> \
--property local_gb=<disk> \
--property cpu_arch=<arch>
```

- **<node>** をベアメタルノードの ID に置き換えてください。
- **<cpu>** は、CPU の数に置き換えます。
- **<ram>** を MB 単位の RAM に置き換えます。
- **<disk>** を GB 単位のディスクサイズに置き換えます。
- **<arch>** は、アーキテクチャータイプに置き換えます。

5. オプション: 各ノードの IPMI 暗号スイートを指定します。

```
(undercloud)$ openstack baremetal node set <node> \
 --driver-info ipmi_cipher_suite=<version>
```

- **<node>** をベアメタルノードの ID に置き換えてください。
- **<version>** は、ノードで使用する暗号スイートのバージョンに置き換えます。以下の有効な値のいずれかに設定します。
  - **3** - ノードは SHA1 暗号スイートで AES-128 を使用します。
  - **17** - ノードは SHA256 暗号スイートで AES-128 を使用します。

6. オプション: 複数のディスクがある場合は、ルートデバイスのヒントを設定して、デプロイメントに使用するディスクをデプロイ ramdisk に通知します。

```
(undercloud)$ openstack baremetal node set <node> \
 --property root_device='{"<property>": "<value>"}'
```

- **<node>** をベアメタルノードの ID に置き換えてください。
- **<property>** と **<value>** は、デプロイメントに使用するディスクの詳細に置き換えます (例: **root_device='{"size": "128"}'**)。
  RHOSP は、次のプロパティーをサポートしています。
  - **model** (文字列): デバイスの ID
  - **vendor** (文字列): デバイスのベンダー
  - **serial** (文字列): ディスクのシリアル番号
  - **hctl** (文字列): SCSI のホスト、チャンネル、ターゲット、Lun
  - **size** (整数): デバイスのサイズ (GB 単位)
  - **wwn** (文字列): 一意のストレージ ID
  - **wwn_with_extension** (文字列): ベンダー拡張子を追加した一意のストレージ ID
  - **wwn_vendor_extension** (文字列): 一意のベンダーストレージ ID

- **rotational** (ブール値): 回転式デバイス (HDD) には true、そうでない場合 (SSD) には false
- **name** (文字列): デバイス名 (例: /dev/sdb1)。このプロパティーは、永続デバイス名が付いたデバイスにのみ使用してください。

注記

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

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

```
(undercloud)$ openstack baremetal port create --node <node_uuid> <mac_address>
```

- **<node_uuid>** をベアメタルノードの一意の ID に置き換えます。
- **<mac_address>** は、PXE ブートに使用する NIC の MAC アドレスに置き換えます。

8. ノードの設定を検証します。

```
(undercloud)$ openstack baremetal node validate <node>
+------------+--------+--------------------------------------+
| Interface  | Result | Reason                               |
+------------+--------+--------------------------------------+
| bios       | True   |                                      |
| boot       | True   |                                      |
| console    | True   |                                      |
| deploy     | False  | Node 229f0c3d-354a-4dab-9a88-ebd318249ad6 |
|            |        | failed to validate deploy image info.     |
|            |        | Some parameters were missing. Missing are:|
|            |        | [instance_info.image_source]              |
| inspect    | True   |                                      |
| management | True   |                                      |
| network    | True   |                                      |
| power      | True   |                                      |
| raid       | True   |                                      |
| rescue     | True   |                                      |
| storage    | True   |                                      |
+------------+--------+--------------------------------------+
```

有効出力の **Result** は、次のことを示しています。

- **False**: インターフェイスは検証に失敗しました。**instance_info.image_source** パラメーターが欠落していることが理由に含まれている場合には、プロビジョニング中に入力されたことが原因である可能性があり、この時点では設定されていません。ディスクイメージ全体を使用している場合は、検証にパスするために **image_source** を設定するだけでよい場合があります。
- **True**: インターフェイスは検証にパスしました。
- **None**: インターフェイスはドライバーでサポートされていません。

7.2.3. オーバークラウドのベアメタルノードのプロビジョニング

ベアメタルノードをプロビジョニングするには、デプロイするベアメタルノードの数と属性を YAML 形式のノード定義ファイルで定義し、これらのノードにオーバークラウドロールを割り当てます。ノードのネットワークレイアウトも定義します。

プロビジョニングプロセスにより、ノード定義ファイルから Heat 環境ファイルが作成されます。この Heat 環境ファイルには、ノード数、予測ノード配置、カスタムイメージ、カスタム NIC など、ノード定義ファイルで設定したノード仕様が含まれています。オーバークラウドをデプロイする際に、このファイルをデプロイメントコマンドに追加します。プロビジョニングプロセスでは、ノード定義ファイル内の各ノードまたはロールに対して定義されたすべてのネットワークのポートリソースもプロビジョニングされます。

前提条件

- アンダークラウドがインストールされます。詳細は、Installing director を参照してください。
- ベアメタルノードは登録とイントロスペクトが行われ、プロビジョニングとデプロイメントに使用できます。詳細は、Registering nodes for the overcloud と ベアメタルノードハードウェアのインベントリー作成 を参照してください。

手順

1. source コマンドで **stackrc** アンダークラウド認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

2. **overcloud-baremetal-deploy.yaml** ノード定義ファイルを作成し、プロビジョニングするロールごとにノード数を定義します。たとえば、3 つのコントローラーノードと 3 つのコンピュートノードをプロビジョニングするには、以下の設定を **overcloud-baremetal-deploy.yaml** ファイルに追加します。

```
- name: Controller
  count: 3
- name: Compute
  count: 3
```

3. オプション: 予測ノード配置を設定します。たとえば、以下の設定を使用すると、3 つのコントローラーノードがノード **node00**、**node01**、および **node02** に、3 つのコンピュートノードがノード **node04**、**node05**、および **node06** に、それぞれプロビジョニングされます。

```
- name: Controller
  count: 3
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
- name: Compute
  count: 3
  instances:
  - hostname: overcloud-novacompute-0
    name: node04
  - hostname: overcloud-novacompute-1
```

```
    name: node05
  - hostname: overcloud-novacompute-2
    name: node06
```

4. オプション: デフォルトでは、プロビジョニングプロセスは **overcloud-hardened-uefi-full.qcow2** イメージを使用します。イメージのローカル URL またはリモート URL を指定することで、特定のノードで使用されるイメージ、またはロールのすべてのノードで使用されるイメージを変更できます。次の例では、イメージをローカル QCOW2 イメージに変更します。

   特定のノード

```
- name: Controller
  count: 3
  instances:
  - hostname: overcloud-controller-0
    name: node00
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
  - hostname: overcloud-controller-1
    name: node01
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
  - hostname: overcloud-controller-2
    name: node02
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
```

   ロールのすべてのノード

```
- name: Controller
  count: 3
  defaults:
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
```

5. ロールのすべてのノードのネットワークレイアウト、または特定のノードのネットワークレイアウトを定義します。

   特定のノード

   次の例では、特定のコントローラーノードのネットワークをプロビジョニングし、予測可能な IP を Internal API ネットワークのノードに割り当てます。

```
- name: Controller
  count: 3
  defaults:
    network_config:
      template: /home/stack/templates/nic-config/myController.j2
```

```
      default_route_network:
      - external
  instances:
  - hostname: overcloud-controller-0
    name: node00
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
      fixed_ip: 172.21.11.100
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
```

ロールのすべてのノード

次の例では、コントローラーロールとコンピューティングロールのネットワークをプロビジョニングします。

```
- name: Controller
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: /home/stack/templates/nic-config/myController.j2 (1)
      default_route_network:
      - external
- name: Compute
  count: 3
  defaults:
    networks:
    - network: ctlplane
      vif: true
    - network: internal_api
      subnet: internal_api_subnet02
    - network: tenant
      subnet: tenant_subnet02
    - network: storage
```

```
    subnet: storage_subnet02
  network_config:
    template: /home/stack/templates/nic-config/myCompute.j2
```

1. **/usr/share/ansible/roles/tripleo_network_config/templates** にあるサンプル NIC テンプレートを使用して、ローカル環境ファイルディレクトリーに独自の NIC テンプレートを作成できます。

6. オプション: デフォルトのディスクパーティションサイズが要件を満たさない場合は、ディスクパーティションサイズの割り当てを設定します。たとえば、**/var/log** パーティションのデフォルトのパーティションサイズは 10 GB です。ログストレージおよび保存要件を考慮して、10 GB が要件を満たすかどうかを判断してください。ログストレージ用に割り当てられたディスクサイズを増やす必要がある場合は、次の設定をノード定義ファイルに追加して、デフォルトをオーバーライドします。

```
- name: Compute
  count: 3
  defaults:
    ...
  ansible_playbooks:
    - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
      extra_vars:
        role_growvols_args:
          default:
            /=8GB
            /tmp=1GB
            /var/log=<log_size>GB
            /var/log/audit=2GB
            /home=1GB
            /var=100%
```

- **<log_size>** を、ログファイルに割り当てるディスクのサイズに置き換えます。

7. Object Storage サービス (swift) とディスク全体のオーバークラウドイメージ **overcloud-hardened-uefi-full** を使用する場合に、ディスクのサイズと **/var** および **/srv** のストレージ要件に基づいて **/srv** パーティションのサイズを設定する必要があります。詳細は、Object Storage サービスのディスクパーティション全体の設定 を参照してください。

8. オプション: カスタムリソースクラスまたはプロファイル機能を使用して、オーバークラウドノードを特定のロールに指定します。詳細は、リソースクラスを一致させることによるオーバークラウドノードのロールの指定 および プロファイルを一致させることによるオーバークラウドノードのロールの指定 を参照してください。

9. ノードに割り当てるその他の属性を定義します。ノード定義ファイルでノード属性を設定するために使用できるプロパティーについて詳しくは、ベアメタルノードのプロビジョニング属性 を参照してください。ノード定義ファイルの例は、ノード定義ファイルの例 を参照してください。

10. オーバークラウドノードをプロビジョニングします。

```
(undercloud)$ openstack overcloud node provision \
 [--templates <templates_directory> \]
 --stack <stack> \
```

```
--network-config \
--output <deployment_file> \
/home/stack/templates/<node_definition_file>
```

- オプション: **--templates** オプションを含めて、**/usr/share/openstack-tripleo-heat-templates** にあるデフォルトテンプレートの代わりに独自のテンプレートを使用します。**<templates_directory>** は、テンプレートを含むディレクトリーへのパスに置き換えます。
- **<stack>** を、ベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。指定しない場合、デフォルトは **overcloud** です。
- **--network-config** オプションの引数を含めて、**cli-overcloud-node-network-config.yaml** Ansible Playbook にネットワーク定義を提供します。Playbook **cli-overcloud-node-network-config.yaml** は、**os-net-config** ツールを使用して、デプロイされたノードにネットワーク設定を適用します。**--network-config** を使用してネットワークを定義をしない場合は、**network-environment.yaml** ファイルで **{{role.name}}NetworkConfigTemplate** パラメーターを設定する必要があります。設定しない場合は、デフォルトのネットワーク定義が使用されます。
- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml)**。
- **<node_definition_file>** は、ノード定義ファイルの名前 (**overcloud-baremetal-deploy.yaml** など) に置き換えます。

11. 別のターミナルでプロビジョニングの進捗を監視します。

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

- プロビジョニングが成功すると、ノードの状態が **available** から **active** に変わります。
- ノードのハードウェアまたはネットワーク設定の障害が原因でノードのプロビジョニングが失敗した場合は、プロビジョニング手順を再度実行する前に、失敗したノードを削除できます。詳細は、障害が発生したベアメタルノードをノード定義ファイルから削除する を参照してください。

12. **metalsmith** ツールを使用して、割り当てやポートなどを含むノードの統合ビューを取得します。

```
(undercloud)$ metalsmith list
```

13. ノードとホスト名の関連付けを確認します。

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

次のステップ

- オーバークラウドの設定およびデプロイ

7.2.4. ベアメタルノードのプロビジョニング属性

次の表を使用して、ノード属性を設定するために使用できるプロパティーと、**openstack baremetal node provision** コマンドでベアメタルノードをプロビジョニングするときに使用できる値を理解してください。

- ロールのプロパティー: ロールのプロパティーを使用して、各ロールを定義します。
- 各ロールのデフォルトプロパティーとインスタンスプロパティー: デフォルトプロパティーまたはインスタンスプロパティーを使用して、使用可能なノードのプールからノードを割り当てるための選択基準を指定し、ベアメタルノードの属性とネットワーク設定プロパティーを設定します。

ベアメタル定義ファイルの作成に関する詳細は、オーバークラウドのベアメタルノードのプロビジョニング を参照してください。

表7.1 ロールのプロパティー

プロパティー	Value
name	ロール名 (必須)
count	このロール用にプロビジョニングするノード数。デフォルト値は **1** です。
defaults	**instances** エントリープロパティーのデフォルト値のディクショナリー。**instances** エントリーのプロパティーは、**defaults** パラメーターで指定したデフォルトを上書きします。
instances	特定のノードの属性を指定するために使用可能な値のディクショナリー。**インスタンス** パラメーターでサポートされているプロパティーの詳細は、**defaults** と **instances** プロパティー を参照してください。定義されるノードの数は、**count** パラメーターの値を超えてはなりません。
hostname_format	このロールのデフォルトのホスト名形式を上書きします。デフォルトで生成されるホスト名は、オーバークラウドのスタック名、ロール、および増分インデックス (すべて小文字) から派生します。たとえば、Controller ロールのデフォルト形式は、**%stackname%-controller-%index%** です。Compute ロールだけは、ロール名のルールに従いません。Compute のデフォルトの形式は、**%stackname%-novacompute-%index%** です。
ansible_playbooks	Ansible Playbook と Ansible 変数の値のディクショナリー。Playbook は、ノードをプロビジョニングしてから、かつノードのネットワークを設定する前に、ロールインスタンスに対して実行されます。Ansible Playbook の指定の詳細は、**ansible_playbooks** プロパティー を参照してください。

表7.2 defaults と instances のプロパティー

プロパティー	Value
hostname	(**instances** のみ) **instance** プロパティーが適用されるノードのホスト名を指定します。ホスト名は、**hostname_format** プロパティーから派生します。カスタムホスト名を使用できます。
name	(**instances** のみ) プロビジョニングするノードの名前。

プロパティー	Value
image	ノードにプロビジョニングするイメージの詳細。サポート対象の **image** プロパティーについては、**image** プロパティー を参照してください。
capabilities	ノードのケイパビリティーを照合する際の選択基準
config_drive	ノードに渡された設定ドライブにデータと初回起動コマンドを追加します。詳細は、**config_drive** プロパティー を参照してください。 **注記** 初回起動時に実行する必要がある設定には、**config_drive** のみを使用してください。他のすべてのカスタム設定については、Ansible Playbook を作成し、**ansible_playbooks** プロパティーを使用して、ノードのプロビジョニング後にロールインスタンスに対して Playbook を実行します。
管理	インスタンスを metalsmith でプロビジョニングするには、**true** (デフォルト) に設定します。インスタンスを事前プロビジョニング済みとして処理するには、**false** に設定します。
networks	インスタンスネットワークを表すディクショナリーのリスト。ネットワーク属性の設定の詳細は、**network** プロパティー を参照してください。
network_config	ロールまたはインスタンスのネットワーク設定ファイルへのリンク。ネットワーク設定ファイルへのリンクの設定の詳細は、**network_config** プロパティー を参照してください。
profile	プロファイルマッチングの選択基準。詳細は、プロファイルを照合することによるオーバークラウドノードへのロール指定 を参照してください。
provisioned	ノードをプロビジョニングするには、**true** (デフォルト) に設定します。ノードのプロビジョニングを解除するには、**false** に設定します。詳細は、ベアメタルノードのスケールダウン を参照してください。
resource_class	ノードのリソースクラスを照合する際の選択基準。デフォルト値は **baremetal** です。詳細は、リソースクラスを一致させることによるオーバークラウドノードのロールの指定 を参照してください。
root_size_gb	ルートパーティションのサイズ (GiB 単位)。デフォルト値は **49** です。
swap_size_mb	スワップパーティションのサイズ (MiB 単位)
traits	ノード特性を照合する際の選択基準としての特性のリスト

表7.3 **image** プロパティー

プロパティー	Value
href	ノードにプロビジョニングするルートパーティションまたはディスクイメージ全体の URL を指定します。サポートされている URL スキーム: **file://**、**http://**、および **https://**。  **注記** **file://** URL スキームを使用してイメージのローカル URL を指定する場合、イメージパスは **/var/lib/ironic/images/** ディレクトリーを指している必要があります。これは、**/var/lib/ironic/images** がアンダークラウドから **ironic-conductor** コンテナーにバインドマウントされ、明示的にイメージを提供するためです。
checksum	ルートパーティションまたはディスクイメージ全体の MD5 チェックサムを指定します。**href** が URL の場合は必須です。
kernel	カーネルイメージのイメージ参照または URL を指定します。パーティションイメージに対してのみ、この属性を使用します。
ramdisk	RAM ディスクイメージのイメージ参照または URL を指定します。パーティションイメージに対してのみ、この属性を使用します。

表7.4 network プロパティー

プロパティー	Value
fixed_ip	このネットワークに使用する特定の IP アドレス
network	ネットワークポートを作成するネットワーク。
subnet	ネットワークポートを作成するサブネット。
ポート	新しいポートを作成する代わりに使用する既存のポート。
vif	プロビジョニングネットワーク (**ctlplane**) で **true** に設定して、ネットワークを仮想インターフェイス (VIF) として接続します。VIF 添付ファイルなしでネットワーキングサービス API リソースを作成するには、**false** に設定します。

表7.5 network_config プロパティー

プロパティー	Value
template	ノードのネットワーク設定を適用するときに使用する Ansible J2 NIC 設定テンプレートを指定します。NIC テンプレートの設定については、オーバークラウドネットワークの設定 を参照してください。

プロパティー	Value
physical_bridge_name	外部ネットワークにアクセスするために作成する OVS ブリッジの名前。デフォルトのブリッジ名は **br-ex** です。
public_interface_name	パブリックブリッジに追加するインターフェイスの名前を指定します。デフォルトのインターフェイスは **nic1** です。
network_config_update	更新時にネットワーク設定の変更を適用するには、**true** に設定します。デフォルトでは無効になっています。
net_config_data_lookup	各ノードまたはノードグループの NIC マッピング設定 **os-net-config** を指定します。
default_route_network	デフォルトルートに使用するネットワーク。デフォルトのルートネットワークは **ctlplane** です。
networks_skip_config	ノードのネットワークを設定するときにスキップするネットワークのリスト。
dns_search_domains	**resolv.conf** に追加する DNS 検索ドメインのリスト (優先順位順)。
bond_interface_ovs_options	結合インターフェイスに使用する OVS オプションまたは結合オプション。たとえば、OVS のボンディングの場合は **lacp=active** および **bond_mode=balance-slb**、Linux のボンディングの場合は **mode=4** です。
num_dpdk_interface_rx_queues	DPDK ボンドまたは DPDK ポートに必要な RX キューの数を指定します。

表7.6 config_drive プロパティー

プロパティー	Value

プロパティー	Value

cloud_config	ノードの起動時に実行するタスクの **cloud-init** クラウド設定データのディクショナリー。たとえば、初回起動時にカスタムネームサーバーを **resolve.conf** ファイルに書き込むには、以下の **cloud_config** を **config_drive** プロパティーに追加します。
	
	config_drive:
	  cloud_config:
	  manage_resolv_conf: true
	  resolv_conf:
	  nameservers:
	  - 8.8.8.8
	  - 8.8.4.4
	  searchdomains:
	  - abc.example.com
	  - xyz.example.com
	  domain: example.com
	  sortlist:
	  - 10.0.0.1/255
	  - 10.0.0.2
	  options:
	  rotate: true
	  timeout: 1

meta_data	config-drive **cloud-init** メタデータに含める追加のメタデータ。メタデータは、ロール名 (**public_keys**、**uuid**、**name**、**hostname**、および **instance-type**) で生成されたメタデータセットに追加されます。**Cloud-init** は、このメタデータをインスタンスデータとして利用できるようにします。

表7.7 ansible_playbooks プロパティー

プロパティー	Value
Playbook	ロール定義 YAML ファイルに相対的な、Ansible Playbook へのパス。

プロパティー	Value
extra_vars	Playbook の実行時に設定する追加の Ansible 変数。次の構文を使用して、追加の変数を指定します。 ansible_playbooks: - playbook: a_playbook.yaml extra_vars: param1: value1 param2: value2 たとえば、ディスク全体のオーバークラウドイメージ **overcloud-hardened-uefi-full.qcow2** でデプロイされた任意のノードの LVM ボリュームを拡張するには、以下の別の変数を Playbook プロパティーに追加します。 ansible_playbooks: - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml extra_vars: role_growvols_args: default: /=8GB /tmp=1GB /var/log=10GB /var/log/audit=2GB /home=1GB /var=100% Controller: /=8GB /tmp=1GB /var/log=10GB /var/log/audit=2GB /home=1GB /srv=50GB /var=100%

7.2.5. 障害が発生したベアメタルノードをノード定義ファイルから削除する

ノードのハードウェアまたはネットワーク設定の障害が原因でノードのプロビジョニングが失敗した場合は、プロビジョニング手順を再度実行する前に、失敗したノードを削除できます。プロビジョニング中に失敗したベアメタルノードを削除するには、ノード定義ファイルでスタックから削除するノードにタグを付け、動作中のベアメタルノードをプロビジョニングする前にノードのプロビジョニングを解除します。

前提条件

- アンダークラウドがインストールされます。詳細は、Installing director を参照してください。
- ノードのハードウェア障害が原因で、ベアメタルノードのプロビジョニングが失敗しました。

手順

1. source コマンドで **stackrc** アンダークラウド認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

2. **overcloud-baremetal-deploy.yaml** ノード定義ファイルを開きます。

3. ノードが割り当てられているロールの **count** パラメーターを減らします。たとえば、以下の設定は、**ObjectStorage** ロールのカウントパラメーターを更新して、**ObjectStorage** 専用のノード数が 3 に減ったことを反映します。

```
- name: ObjectStorage
  count: 3
```

4. ロールの **instances** 属性でまだ定義されていない場合は、スタックから削除するノードの **hostname** と **name** を定義します。

5. 削除するノードに属性 **provisioned: false** を追加します。たとえば、スタックからノード **overcloud-objectstorage-1** を削除するには、**overcloud-baremetal-deploy.yaml** ファイルに以下のスニペットを追加します。

```
- name: ObjectStorage
  count: 3
  instances:
  - hostname: overcloud-objectstorage-0
    name: node00
  - hostname: overcloud-objectstorage-1
    name: node01
    # Removed from cluster due to disk failure
    provisioned: false
  - hostname: overcloud-objectstorage-2
    name: node02
  - hostname: overcloud-objectstorage-3
    name: node03
```

6. ベアメタルノードのプロビジョニングを解除します。

```
(undercloud)$ openstack overcloud node unprovision \
 --stack <stack> \
 --network-ports \
 /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- **<stack>** を、ベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。指定しない場合、デフォルトは **overcloud** です。

7. オーバークラウドノードをプロビジョニングして、デプロイメントコマンドに含める heat 環境ファイルを更新して生成します。

```
(undercloud)$ openstack overcloud node provision \
--stack <stack> \
--output <deployment_file> \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイ

- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する Heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml**)。

7.2.6. リソースクラスを一致させることによるオーバークラウドノードのロールの指定

カスタムリソースクラスを使用して、オーバークラウドノードを特定のロールに指定できます。リソースクラスは、ノードとデプロイロールを照合します。デフォルトでは、すべてのノードに **baremetal** のリソースクラスが割り当てられます。

注記

ノードのプロビジョニング後にノードに割り当てられたリソースクラスを変更するには、スケールダウン手順を使用してノードのプロビジョニングを解除してから、スケールアップ手順を使用して、新しいリソースクラスの割り当てでノードを再プロビジョニングする必要があります。詳細は、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。

前提条件

- 現在、オーバークラウド用のベアメタルノードの初期プロビジョニング手順を行っている。

手順

1. カスタムリソースクラスを持つロールに指定する各ベアメタルノードを割り当てます。

```
(undercloud)$ openstack baremetal node set \
 --resource-class <resource_class> <node>
```

   - **<resource_class>** は、リソースクラスの名前 (**baremetal.CPU-PINNING** など) に置き換えます。
   - **<node>** をベアメタルノードの ID に置き換えてください。

2. まだ定義されていない場合は、**overcloud-baremetal-deploy.yaml** ファイルにロールを追加します。
3. ロールのノードに割り当てるリソースクラスを指定します。

```
- name: <role>
  count: 1
  defaults:
    resource_class: <resource_class>
```

   - **<role>** をロールの名前に置き換えます。
   - **<resource_class>** は、手順 1 で指定したリソースクラスの名前に置き換えます。

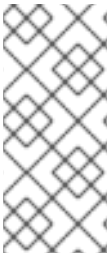
4. オーバークラウドのベアメタルノードのプロビジョニング に戻り、プロビジョニングプロセスを完了します。

7.2.7. プロファイルを一致させることによるオーバークラウドノードのロールの指定

プロファイル機能を使用して、オーバークラウドノードを特定のロールに指定できます。プロファイルは、ノードの機能をデプロイメントロールと照合します。

ヒント

イントロスペクションルールを使用して、自動プロファイル割り当てを実行することもできます。詳しくは、プロファイルの自動タグ付けの設定 を参照してください。



注記

ノードのプロビジョニング後にノードに割り当てられたプロファイルを変更するには、スケールダウン手順を使用してノードのプロビジョニングを解除してから、スケールアップ手順を使用して、新しいプロファイルの割り当てでノードを再プロビジョニングする必要があります。詳細は、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。

前提条件

- 現在、オーバークラウド用のベアメタルノードの初期プロビジョニング手順を行っている。

手順

1. 新しいノード機能を追加するたびに、既存のノード機能が上書きされます。したがって、再設定するには、登録済みの各ノードの既存の機能を取得する必要があります。

```
$ openstack baremetal node show <node> \
 -f json -c properties | jq -r .properties.capabilities
```

2. ノードの既存の機能に **profile:<profile>** を追加して、ロールプロファイルに一致させる各ベアメタルノードにプロファイル機能を割り当てます。

```
(undercloud)$ openstack baremetal node set  <node> \
 --property capabilities="profile:<profile>,<capability_1>,...,<capability_n>"
```

   - **<node>** は、ベアメタルノードの名前または ID に置き換えます。
   - **<profile>** は、ロールプロファイルに一致するプロファイルの名前に置き換えます。
   - **<capability_1>** および **<capability_n>** までのすべての機能は、手順 1 で取得した各機能に置き換えます。

3. まだ定義されていない場合は、**overcloud-baremetal-deploy.yaml** ファイルにロールを追加します。

4. ロールのノードに割り当てるプロファイルを定義します。

```
- name: <role>
  count: 1
  defaults:
    profile: <profile>
```

   - **<role>** をロールの名前に置き換えます。
   - **<profile>** は、ノードの機能に一致するプロファイルの名前に置き換えます。

5. オーバークラウドのベアメタルノードのプロビジョニング に戻り、プロビジョニングプロセスを完了します。

7.2.8. Object Storage サービスのディスクパーティション全体の設定

ディスクイメージ全体である **overcloud-hardened-uefi-full** は、個別のボリュームに分割されます。デフォルトでは、ディスク全体のオーバークラウドイメージでデプロイされたノードの **/var** パーティションは、ディスクが完全に割り当てられるまで自動的に増加します。Object Storage サービス (swift) を使用する場合は、ディスクのサイズと **/var** および **/srv** のストレージ要件に基づいて **/srv** パーティションのサイズを設定します。

前提条件

- 現在、オーバークラウド用のベアメタルノードの初期プロビジョニング手順を行っている。

手順

1. **overcloud-baremetal-deploy.yaml** ノード定義ファイルの Ansible playbook 定義で、追加の Ansible 変数として **role_growvols_args** を使用して、**/srv** および **/var** パーティションを設定します。**/srv** または **/var** のいずれかを GB 単位の絶対サイズに設定し、もう一方を 100% に設定して、残りのディスク領域を消費します。

   - 次の設定例では、**/srv** をコントローラーノードにデプロイされた Object Storage サービスの絶対サイズに、**/var** を 100% に設定して、残りのディスク領域を消費します。

     ```
     ansible_playbooks:
       - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
         extra_vars:
           role_growvols_args:
             default:
               /=8GB
               /tmp=1GB
               /var/log=10GB
               /var/log/audit=2GB
               /home=1GB
               /var=100%
             Controller:
               /=8GB
               /tmp=1GB
               /var/log=10GB
               /var/log/audit=2GB
               /home=1GB
               /srv=50GB
               /var=100%
     ```

   - 以下の設定例では、**/var** を絶対サイズに、**/srv** を 100% に設定して、Object Storage サービスの Object Storage ノードの残りのディスクスペースを消費します。

     ```
     ansible_playbooks:
       - playbook: /usr/share/ansible/tripleo-playbooks/cli-overcloud-node-growvols.yaml
         extra_vars:
           role_growvols_args:
             default:
               /=8GB
               /tmp=1GB
               /var/log=10GB
               /var/log/audit=2GB
               /home=1GB
     ```

```
    /var=100%
  ObjectStorage:
    /=8GB
    /tmp=1GB
    /var/log=10GB
    /var/log/audit=2GB
    /home=1GB
    /var=10GB
    /srv=100%
```

2. オーバークラウドのベアメタルノードのプロビジョニング に戻り、プロビジョニングプロセスを完了します。

7.2.9. ノード定義ファイルの例

次のノード定義ファイルの例では、3 つのコントローラーノードと 3 つのコンピュートノードの予測ノード配置と、それらが使用するデフォルトネットワークを定義しています。この例は、リソースクラスまたはノード機能プロファイルの照合に基づいて指定されたノードが含まれるカスタムロールを定義する方法も示しています。

```
- name: Controller
  count: 3
  defaults:
    image:
      href: file:///var/lib/ironic/images/overcloud-custom.qcow2
    networks:
    - network: ctlplane
      vif: true
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storagemgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
        template: /home/stack/templates/nic-config/myController.j2
        default_route_network:
        - external
    profile: nodeCapability
  instances:
  - hostname: overcloud-controller-0
    name: node00
  - hostname: overcloud-controller-1
    name: node01
  - hostname: overcloud-controller-2
    name: node02
- name: Compute
  count: 3
  defaults:
    networks:
    - network: ctlplane
```

```
    vif: true
  - network: internal_api
    subnet: internal_api_subnet02
  - network: tenant
    subnet: tenant_subnet02
  - network: storage
    subnet: storage_subnet02
  network_config:
    template: /home/stack/templates/nic-config/myCompute.j2
  resource_class: baremetal.COMPUTE
instances:
- hostname: overcloud-novacompute-0
  name: node04
- hostname: overcloud-novacompute-1
  name: node05
- hostname: overcloud-novacompute-2
  name: node06
```

7.2.10. 仮想メディアブートの有効化

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す。実稼働環境にはデプロイしないでください。テクノロジープレビュー機能の詳細は、対象範囲の詳細 を参照してください。

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

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

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

前提条件

- **undercloud.conf** ファイルの **enabled_hardware_types** パラメーターで、Redfish ドライバーが有効化されている。
- ベアメタルノードが登録されている。
- Image サービス (glance) に IPA およびインスタンスイメージがある。
- UEFI ノードの場合、EFI システムパーティション (ESP) イメージも Image サービス (glance) で利用可能でなければなりません。

- クリーニングおよびプロビジョニング用ネットワーク

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. Bare Metal Provisioning サービスの起動インターフェイスを **redfish-virtual-media** に設定します。

```
(undercloud)$ openstack baremetal node set --boot-interface redfish-virtual-media <node>
```

- **<node>** はノード名に置き換えてください。

4. ESP イメージを定義します。

```
(undercloud)$ openstack baremetal node set --driver-info bootloader=<esp> <node>
```

- **<esp>** を Image サービス (glance) イメージの UUID または ESP イメージの URL に置き換えます。
- **<node>** はノード名に置き換えてください。

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

```
(undercloud)$ openstack baremetal port create --pxe-enabled True \
 --node <node_uuid> <mac_address>
```

- **<node_uuid>** をベアメタルノードの UUID に置き換えます。
- **<mac_address>** をベアメタルノードの NIC の MAC アドレスに置き換えます。

7.2.11. マルチディスク Ceph クラスターのルートディスクの定義

Ceph Storage ノードは通常、複数のディスクを使用します。Director は、複数のディスク設定でルートディスクを識別する必要があります。オーバークラウドイメージは、プロビジョニングプロセス中にルートディスクに書き込まれます。

ハードウェアプロパティーは、ルートディスクを識別するために使用されます。ルートディスクの識別に使用できるプロパティーの詳細は、ルートディスクを識別するプロパティー を参照してください。

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。

```
(undercloud)$ openstack baremetal introspection data save <node_uuid> --file
<output_file_name>
```

- **<node_uuid>** をノードの UUID に置き換えます。

- **<output_file_name>** を、ノードイントロスペクションの出力を含むファイルの名前に置き換えます。
  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. 一意のハードウェアプロパティーを使用して、ノードのルートディスクを設定します。
   (undercloud)$ openstack baremetal node set --property root_device='{<property_value>}' <node-uuid>

   - **<property_value>** を、ルートディスクの設定に使用するイントロスペクションデータからの一意のハードウェアプロパティー値に置き換えます。

   - **<node_uuid>** をノードの UUID に置き換えます。

注記

一意のハードウェアプロパティーは、ディスクを一意に識別するハードウェアイントロスペクションステップからの任意のプロパティーです。たとえば、次のコマンドは、ディスクのシリアル番号を使用してルートディスクを設定します。

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

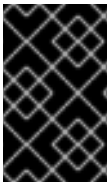
3. 最初にネットワークから起動し、次にルートディスクから起動するように、各ノードの BIOS を設定します。

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 node provision** 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。

7.2.12. ルートディスクを識別するプロパティー

以下の属性を定義すると、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)



重要

永続的な名前を持つデバイスには **name** プロパティーを使用します。ノードの起動時に値が変更される可能性があるため、**name** プロパティーを使用して永続的な名前を持たないデバイスのルートディスクを設定しないでください。

7.2.13. 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

Red Hat OpenStack Platform (RHOSP) デプロイメントのデフォルトイメージは **overcloud-hardened-uefi-full.qcow2** です。**overcloud-hardened-uefi-full.qcow2** イメージは、有効な Red Hat OpenStack Platform (RHOSP) サブスクリプションを使用します。サブスクリプションのエンタイトルメントを消費したくない場合は **overcloud-minimal** イメージを使用して、有償の Red Hat サブスクリプションが

限度に達するのを回避できます。これは、たとえば Ceph デーモンのみでノードをプロビジョニングする場合や、他の OpenStack サービスを実行したくないベアオペレーティングシステム (OS) をプロビジョニングする場合に役立ちます。**overcloud-minimal** イメージの取得方法に関する詳細は、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。

注記

overcloud-minimal イメージでは、標準の Linux ブリッジのみサポートされます。**overcloud-minimal** イメージでは、Open vSwitch (OVS) はサポートされません。OVS は、Red Hat OpenStack Platform サブスクリプションエンタイトルメントを必要とする OpenStack サービスだからです。Ceph Storage ノードをデプロイするのに OVS は必要ありません。**ovs_bond** を使用してボンディングを定義する代わりに、**linux_bond** を使用します。**linux_bond** の詳細は、Linux ボンディングの作成 を参照してください。

手順

1. **overcloud-minimal** イメージと関連するカーネルおよび RAM ディスクをアンダークラウドの **/var/lib/ironic/images/** にアップロードします。
以下はファイルのアップロードの例です。

```
(undercloud) [stack@undercloud ~]$ openstack overcloud image upload --image-path
./images/ --os-image-name overcloud-minimal.qcow2 --update-existing --image-type os
Image "file:///var/lib/ironic/images/overcloud-minimal.vmlinuz" was copied.
+---------------------------------------------------------+------------------+----------+
|                          Path                           |       Name       |   Size   |
+---------------------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.vmlinuz | overcloud-minimal | 12190232 |
+---------------------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.initrd" was copied.
+---------------------------------------------------------+------------------+----------+
|                          Path                           |       Name       |   Size   |
+---------------------------------------------------------+------------------+----------+
| file:///var/lib/ironic/images/overcloud-minimal.initrd | overcloud-minimal | 82284184 |
+---------------------------------------------------------+------------------+----------+
Image "file:///var/lib/ironic/images/overcloud-minimal.raw" was copied.
+-------------------------------------------------------+------------------+------------+
|                          Path                         |       Name       |    Size    |
+-------------------------------------------------------+------------------+------------+
| file:///var/lib/ironic/images/overcloud-minimal.raw | overcloud-minimal | 3242131456 |
+-------------------------------------------------------+------------------+------------+
```

2. **overcloud-baremetal-deploy.yaml** ファイルを開きます。
3. **overcloud-minimal** イメージを使用するノードの **image** プロパティーを追加または更新します。特定のノードまたはロールのすべてのノードでイメージを **overcloud-minimal** に設定できます。

特定のノード

```
- name: Ceph
  count: 3
  instances:
  - hostname: overcloud-ceph-0
    name: node00
```

```
    image:
      href: 'file:///var/lib/ironic/images/overcloud-minimal.raw'
      kernel: 'file:///var/lib/ironic/images/overcloud-minimal.vmlinuz'
      ramdisk: 'file:///var/lib/ironic/images/overcloud-minimal.initrd'
  - hostname: overcloud-ceph-1
    name: node01
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
  - hostname: overcloud-ceph-2
    name: node02
    image:
      href: file:///var/lib/ironic/images/overcloud-full-custom.qcow2
```

ロールのすべてのノード

```
- name: Ceph
  count: 3
  defaults:
    image:
      href: 'file:///var/lib/ironic/images/overcloud-minimal.raw'
      kernel: 'file:///var/lib/ironic/images/overcloud-minimal.vmlinuz'
      ramdisk: 'file:///var/lib/ironic/images/overcloud-minimal.initrd'
  instances:
  - hostname: overcloud-ceph-0
    name: node00
  - hostname: overcloud-ceph-1
    name: node01
  - hostname: overcloud-ceph-2
    name: node02
```

4. **roles_data.yaml** ロール定義ファイルで、**rhsm_enforce** パラメーターを **False** に設定します。

```
rhsm_enforce: False
```

5. プロビジョニングコマンドを実行します。

```
(undercloud)$ openstack overcloud node provision \
--stack stack \
--output /home/stack/templates/overcloud-baremetal-deployed.yaml \
/home/stack/templates/overcloud-baremetal-deploy.yaml
```

6. **overcloud-baremetal-deployed.yaml** 環境ファイルを **openstack overcloud deploy** コマンドに渡します。

7.3. オーバークラウドの設定およびデプロイ

オーバークラウドのネットワークリソースとベアメタルノードをプロビジョニングしたら、director のインストールで提供される未編集の heat テンプレートファイルと、作成したカスタム環境ファイルを使用して、オーバークラウドを設定できます。オーバークラウドの設定が完了したら、オーバークラウド環境をデプロイできます。

重要

基本的なオーバークラウドでは、Block Storage 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。Red Hat では、Block Storage に Red Hat Ceph Storage などの外部ストレージソリューションを使用することを推奨しています。

7.3.1. 前提条件

- オーバークラウドに必要なネットワークリソースとベアメタルノードをプロビジョニングしている。

7.3.2. 環境ファイルを使用したオーバークラウドの設定

アンダークラウドには、オーバークラウドの作成プランを形作るさまざまな 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用して、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。環境ファイルの拡張子は **.yaml** または **.template** にする必要があります。

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

-e オプションを使用して、環境ファイルをオーバークラウドデプロイメントに含めます。**-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。後で指定する環境ファイルで定義され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。

初回のデプロイメント後にオーバークラウド設定を変更するには、以下のアクションを実行します。

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

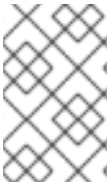
オーバークラウドの設定は直接編集しないでください。手動で設定しても、オーバークラウドスタックの更新時に、director が設定を上書きするためです。

注記

Open Virtual Networking (OVN) は、Red Hat OpenStack Platform 17.1 におけるデフォルトのネットワークメカニズムドライバーです。分散仮想ルーター (DVR) で OVN を使用する場合には、**openstack overcloud deploy** コマンドに **environments/services/neutron-ovn-dvr-ha.yaml** ファイルを追加する必要があります。DVR なしで OVN を使用する場合には、**openstack overcloud deploy** コマンドに **environments/services/neutron-ovn-ha.yaml** ファイルを追加する必要があります。

7.3.3. アンダークラウド CA を信頼するための環境ファイルの作成

アンダークラウドで TLS を使用され認証局 (CA) が一般に信頼できない場合には、SSL エンドポイント暗号化にアンダークラウドが運用する CA を使用することができます。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。

アンダークラウドで使用することの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**: 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。**undercloud.conf** ファイルの **generate_service_certificate** オプションを使用して、証明書の自動生成を有効にします。この場合、director は CA 証明書 **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** を生成し、アンダークラウドの HAProxy インスタンスがサーバー証明書を使用するように設定します。この CA 証明書を OpenStack Platform に配置するには、証明書を **inject-trust-anchor-hiera.yaml** ファイルに追加します。

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** を使用してください。

手順

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-----
MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECgwH
UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
-----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:
    undercloud-ca:
      content: |
        -----BEGIN CERTIFICATE-----
        MIIDITCCAn2gAwIBAgIJAOnPtx2hHEhrMA0GCSqGSIb3DQEBCwUAMGExCzAJBgNV

BAYTAlVTMQswCQYDVQQIDAJOQzEQMA4GA1UEBwwHUmFsZWlnaDEQMA4GA1UECgwH

UmVkIEhhdDELMAkGA1UECwwCUUUxFDASBgNVBAMMCzE5Mi4xNjguMC4yMB4XDTE3
        -----END CERTIFICATE-----
```

注記

- 証明書の文字列は、PEM の形式に従う必要があります。
- **CAMap** パラメーターには、SSL/TLS 設定に関連する他の証明書が含まれる場合があります。

3. **/home/stack/inject-trust-anchor-hiera.yaml** ファイルをデプロイコマンドに追加します。director は、オーバークラウドのデプロイメント時に CA 証明書をそれぞれのオーバークラウドノードにコピーします。これにより、それぞれのノードは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。

7.3.4. 新規デプロイメントでの TSX の無効化

Red Hat Enterprise Linux 8.3 以降、カーネルは、デフォルトで Intel Transactional Synchronization Extensions (TSX) 機能のサポートを無効にします。

ワークロードまたはサードパーティーベンダー用に厳密に要求しない限り、新しいオーバークラウドで TSX を明示的に無効にする必要があります。

環境ファイルで **KernelArgs** heat パラメーターを設定します。

```
parameter_defaults:
  ComputeParameters:
    KernelArgs: "tsx=off"
```

openstack overcloud deploy コマンドを実行する際に、環境ファイルを指定します。

関連情報

- Guidance on Intel TSX impact on OpenStack guests (applies for RHEL 8.3 and above)

7.3.5. オーバークラウド設定の検証

オーバークラウドをデプロイする前に、heat テンプレートと環境ファイルを検証します。

重要

- 17.0 で API が変更されたため、現在、次の検証は不安定になっています。
  - switch-vlans
  - network-environment
  - dhcp-provisioning
- 検証結果が **FAILED** であっても、Red Hat OpenStack Platform のデプロイや実行が妨げられることはありません。ただし、**FAILED** の検証結果は、実稼働環境で問題が発生する可能性があることを意味します。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. デプロイメントに必要なすべての環境ファイルを使用してオーバークラウドスタックを更新します。

```
$ openstack overcloud deploy --templates \
  -e environment-file1.yaml \
  -e environment-file2.yaml \
  ...
  --stack-only
```

4. **オプション**: 検証実行から除外する検証を含む YAML ファイルを作成します。

```
<validation_name>:
  hosts: <targeted_hostname>
```

- **<validation_name>** を、検証実行から除外する検証の名前に置き換えます。
- **<targeted_hostname>** を、検証が含まれるホストの名前に置き換えます。

5. オーバークラウドスタックを検証します。

```
$ validation run \
  --group pre-deployment \
  --inventory <inventory_file>
  [--skiplist <validation_skips>]
```

- **<inventory_file>** ファイルを Ansible インベントリーファイルの名前および場所に置き換えます (例: ~/**tripleo-deploy**/**undercloud**/**tripleo-ansible-inventory.yaml**)。
- **<validation_skips>** を、検証実行から除外する検証のリストを含む YAML ファイルの名前と場所に置き換えます。
  注記:
- **validation run --group pre-deployment** コマンドには、**node-disks** 検証が含まれます。既知の問題により、この検証は現在失敗します。この問題を回避するには、**node-disks** 検証を含む **yaml** ファイルを含む **--skiplist** 引数を、**validation run --group pre-deployment** コマンドに追加します。
- 検証を実行すると、出力の **Reasons** 列は 79 文字に制限されます。検証結果を完全に表示するには、検証ログファイルを表示します。

6. 検証レポートの結果を確認します。

```
$ validation history get [--full] [--validation-log-dir <log_dir>] <uuid>
```

- オプション: **--full** オプションを使用して、検証実行からの詳細な出力を表示します。
- オプション: **--validation-log-dir** オプションを使用して、検証実行の出力を検証ログに書き込みます。
- **<uuid>** は、検証実行の UUID に置き換えます。

7.3.6. オーバークラウドの作成

Red Hat OpenStack Platform (RHOSP) オーバークラウド環境を作成する最後の段階は、**openstack overcloud deploy** コマンドを実行してオーバークラウドを作成することです。**openstack overcloud deploy** コマンドで使用できるオプションについては、デプロイメントコマンドのオプション を参照してください。

手順

1. オーバークラウド環境に必要な環境ファイルと設定ファイル (director のインストールで提供される未編集の heat テンプレートファイルと作成したカスタム環境ファイルの両方) を照合します。これには、次のファイルが含まれている必要があります。
   - **overcloud-baremetal-deployed.yaml** ノード定義ファイル。
   - **overcloud-networks-deployed.yaml** ネットワーク定義ファイル。
   - **overcloud-vip-deployed.yaml** ネットワーク VIP 定義ファイル。
   - コンテナー化された OpenStack サービスのコンテナーイメージの場所。
   - Red Hat CDN または Satellite 登録用の環境ファイル
   - その他のカスタム環境ファイル
2. 環境ファイルと設定ファイルを優先順位に従って整理し、編集されていない Heat テンプレートファイル、その次にデフォルトプロパティーのオーバーライドなどのカスタム設定を含む環境ファイルをリストします。
3. 以下の例のように、設定ファイルとテンプレートを必要な順序で指定して、**openstack overcloud deploy** コマンドを作成します。

```
(undercloud) $ openstack overcloud deploy --templates \
 [-n /home/stack/templates/network_data.yaml \ ]
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml\
  -e /home/stack/templates/overcloud-networks-deployed.yaml\
  -e /home/stack/templates/overcloud-vip-deployed.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /home/stack/inject-trust-anchor-hiera.yaml \
 [-r /home/stack/templates/roles_data.yaml ]
```

-n /home/stack/templates/network_data.yaml

カスタムネットワーク設定を指定します。ネットワーク分離またはカスタム設定可能なネットワークを使用する場合に必要です。オーバークラウドネットワークの設定は、オーバークラウドネットワークの設定 を参照してください。

-e /home/stack/containers-prepare-parameter.yaml

コンテナーイメージ準備の環境ファイルを追加します。このファイルはアンダークラウドのインストール時に生成したもので、オーバークラウドの作成に同じファイルを使用することができます。

-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。

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

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ークラウドを有効にする場合に生成されるロールデータ。

4. オーバークラウドの作成が完了すると、オーバークラウドを設定するために実施された Ansible のプレイの概要が director により提示されます。

```
PLAY RECAP *********************************************************************
overcloud-compute-0     : ok=160  changed=67   unreachable=0    failed=0
overcloud-controller-0  : ok=210  changed=93   unreachable=0    failed=0
undercloud              : ok=10   changed=7    unreachable=0    failed=0

Tuesday 15 October 2018  18:30:57 +1000 (0:00:00.107) 1:06:37.514 ******
===============================================================================
```

5. オーバークラウドの作成が完了すると、director はオーバークラウドにアクセスするための詳細を提供します。

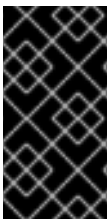
```
Ansible passed.
Overcloud configuration completed.
Overcloud Endpoint: http://192.168.24.113:5000
Overcloud Horizon Dashboard URL: http://192.168.24.113:80/dashboard
Overcloud rc file: /home/stack/overcloudrc
Overcloud Deployed
```

ヒント

新しい環境ファイルで設定を更新するたびに追加するファイルに、デプロイメントコマンドを格納できます。

7.3.7. デプロイメントコマンドのオプション

以下の表には、**openstack overcloud deploy** コマンドの追加パラメーターをまとめています。



重要

一部のオプションは、本リリースでは **テクノロジープレビュー** として提供されているため、Red Hat では全面的にはサポートしていません。これらはテスト目的にのみご利用いただく機能で、実稼働環境で使用すべきではありません。テクノロジープレビュー機能の詳細は、対象範囲の詳細 を参照してください。

表7.8 デプロイメントコマンドのオプション

パラメーター	説明
--templates [TEMPLATES]	デプロイする heat テンプレートが含まれるディレクトリー。空欄にした場合には、デプロイメント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時間 (分単位)
--libvirt-type [LIBVIRT_TYPE]	ハイパーバイザーに使用する仮想化タイプ

パラメーター	説明
--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、Controller ノードが一貫して同じ時刻ソースを参照することが重要です。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、プロキシー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 **tripleo-admin** ユーザーを介して行われます。
--overcloud-ssh-key OVERCLOUD_SSH_KEY	オーバークラウドノードへの SSH アクセスに使用する鍵のパスを定義します。
--overcloud-ssh-network OVERCLOUD_SSH_NETWORK	オーバークラウドノードへの SSH アクセスに使用するネットワーク名を定義します。
-e [EXTRA HEAT TEMPLATE]、--environment-file [ENVIRONMENT FILE]	オーバークラウドのデプロイメントに渡す追加の環境ファイル。このオプションは複数回指定することができます。**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含まれるディレクトリー。デプロイメントコマンドでは、これら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。
-r ROLES_FILE	ロールファイルを定義し、**--templates** ディレクトリーのデフォルトの **roles_data.yaml** を上書きします。ファイルの場所は、絶対パスまたは **--templates** に対する相対パスになります。
-n NETWORKS_FILE	ネットワークファイルを定義し、**--templates** ディレクトリーのデフォルトの network_data.yaml を上書きします。ファイルの場所は、絶対パスまたは **--templates** に対する相対パスになります。

パラメーター	説明
-p PLAN_ENVIRONMENT_FILE	プラン環境ファイルを定義し、**--templates** ディレクトリーのデフォルトの **plan-environment.yaml** を上書きします。ファイルの場所は、絶対パスまたは **--templates** に対する相対パスになります。
--no-cleanup	デプロイメント後に一時ファイルを削除せず、それらの場所をログに記録するには、このオプションを使用します。
--update-plan-only	実際のデプロイメントを実行せずにプランを更新するには、このオプションを使用します。
--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	オーバークラウドを作成せずにオーバークラウドで検証チェックを実行するには、このオプションを使用します。
--run-validations	**openstack-tripleo-validations** パッケージで提供される外部検証を実行するには、このオプションを使用します。
--skip-postconfig	オーバークラウドデプロイ後の設定を省略するには、このオプションを使用します。
--force-postconfig	オーバークラウドデプロイ後の設定を強制的に行うには、このオプションを使用します。
--skip-deploy-identifier	デプロイメントコマンドで **DeployIdentifier** パラメーターの一意の ID を生成するのを希望しない場合は、このオプションを使用します。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。

パラメーター	説明
--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
--disable-password-generation	オーバークラウドサービスのパスワード生成を無効にする場合は、このオプションを使用します。
--deployed-server	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をデプロイする場合は、このオプションを使用します。**--disable-validations** と併用されます。
--no-config-download, --stack-only	**config-download** ワークフローを無効にして、スタックおよび関連する OpenStack リソースだけを作成する場合は、このオプションを使用します。このコマンドによってオーバークラウドにソフトウェア設定が適用されることはありません。
--config-download-only	オーバークラウドスタックの作成を無効にして、ソフトウェア設定を適用する **config-download** ワークフローだけを実行する場合は、このオプションを使用します。
--output-dir OUTPUT_DIR	保存した **config-download** の出力に使用するディレクトリー。ディレクトリーは mistral ユーザーが書き込み可能でなければなりません。指定しない場合、director はデフォルトの **/var/lib/mistral/overcloud** を使用します。
--override-ansible-cfg OVERRIDE_ANSIBLE_CFG	Ansible 設定ファイルへのパス。このファイルの設定は、**config-download** がデフォルトで生成する設定を上書きします。
--config-download-timeout CONFIG_DOWNLOAD_TIMEOUT	**config-download** のステップに使用するタイムアウト時間 (分単位)。設定しなければ、スタックデプロイメント操作後の **--timeout** パラメーターの残り時間にかかわらず、director はデフォルトをその時間に設定します。
--limit NODE1,NODE2	**(テクノロジープレビュー)** config-download Playbook の実行を特定のノードまたはノードセットに制限する場合は、このオプションを使用してノードのコンマ区切りリストを指定します。たとえば、**--limit** オプションは、スケールアップ操作時に新規ノード上でのみ config-download を実行する場合に役立ちます。この引数により、ホスト間のインスタンスのライブマイグレーションが失敗する可能性があります。ansible-playbook-command.sh スクリプトを使用した config-download の実行 を参照してください。

パラメーター	説明
--tags TAG1,TAG2	**(テクノロジープレビュー)** config-download の特定のタスクセットでデプロイメントを実施する場合は、このオプションを使用して config-download Playbook からのタグのコンマ区切りリストを指定します。
--skip-tags TAG1,TAG2	**(テクノロジープレビュー)** config-download Playbook のタグの一部を省略する場合は、このオプションを使用して省略するタグのコンマ区切りリストを指定します。

オプションの全リストを表示するには、以下のコマンドを実行します。

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

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比しています。

表7.9 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
--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
--disable-validations	デプロイメント前の検証を完全に無効にします。これらの検証は、デプロイメント前の検証として組み込まれていましたが、**openstack-tripleo-validations** パッケージで提供される外部検証に置き換えられています。	パラメーターのマッピングなし
--config-download	**config-download** のメカニズムを使用してデプロイメントを実行します。これは現在のデフォルトであり、この CLI のオプションは今後廃止される可能性があります。	パラメーターのマッピングなし

パラメーター	説明	heat テンプレートのパラメーター
--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する場合は、このオプションを使用します。	**RhsmVars**
--reg-method	このオプションを使用して、オーバークラウドノードの登録方法を定義します。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。	**RhsmVars**
--reg-org [REG_ORG]	登録に使用する組織。	**RhsmVars**
--reg-force	すでに登録済みでもシステムを登録する場合は、このオプションを使用します。	**RhsmVars**
--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。サーバーが 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。サーバーが 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 **rhnreg_ks** に登録します。	**RhsmVars**
--reg-activation-key [REG_ACTIVATION_KEY]	登録に使用するアクティベーションキーを定義する場合は、このオプションを使用します。	**RhsmVars**

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。

7.3.8. デフォルトのオーバークラウドディレクトリーの内容

RHOSP 17 では、すべての設定ファイルが 1 つのディレクトリーにあります。ディレクトリーの名前は、使用した openstack コマンドとスタックの名前を組み合わせたものです。ディレクトリーにはデフォルトの場所がありますが、**--working-dir** オプションを使用してデフォルトの場所を変更できます。このオプションは、デプロイメントで使用されるファイルの読み取りまたは作成を行う任意の **tripleoclient** コマンドで使用できます。

Default location	コマンド
$HOME/tripleo-deploy/undercloud	**tripleo deploy** に基づく **undercloud install**
$HOME/tripleo-deploy/<stack>	**tripleo deploy**、<stack> はデフォルトでは **standalone** です
$HOME/overcloud-deploy/<stack>	**overcloud deploy**、<stack> はデフォルトで **overcloud** です。

次の表は、~/**overcloud-deploy**/**overcloud** ディレクトリーに含まれるファイルとディレクトリーの詳細を示しています。

表7.10 オーバークラウドディレクトリーの内容

ディレクトリー	設定
cli-*`	**ansible-runner** が CLI ansible ベースのワークフローに使用するディレクトリー。 cli-config-download cli-enable-ssh-admin cli-grant-local-access cli-undercloud-get-horizon-url
config-download	**config-download** ディレクトリー。以前の Red Hat OpenStack Platform (RHOSP) リリースでは、このディレクトリーの名前は ~/**config-download** または /**var**/**lib**/**mistral**/**<stack>** でした。
environment	**openstack stack environment show <stack>** コマンドで生成された保存済みスタック環境が含まれています。
heat-launcher	一時的な Heat 設定とデータベースのバックアップを含む一時的な Heat 作業ディレクトリー。
outputs	**openstack stack output show <stack> <output>** コマンドで生成された保存済みスタック出力が含まれます。

ディレクトリー	設定
<stack>-deployment_status.yaml	保存されたスタックステータスが含まれます。**overcloud** がスタックの名前である場合、このファイルの名前は **overcloud-deployment_status.yaml** です。
<stack>-export.yaml	**openstack overcloud export --stack <stack>** コマンドで生成されたスタックのエクスポート情報が含まれます。**overcloud** がスタックの名前である場合、このファイルの名前は **overcloud-export.yaml** です。
<stack>-install-*.tar.bzip2	作業ディレクトリーの tarball (例: **overcloud-install-20220823213643.tar.bzip2)**。
<stack>-passwords.yaml	スタックのパスワードが含まれています。**overcloud** がスタックの名前である場合、このファイルの名前は **overcloud-passwords.yaml** です。
<stack>rc	オーバークラウド API を使用するために必要な認証情報ファイル (例: **overcloudrc)**。
tempest-deployer-input.conf	Tempest 設定が含まれています。
tripleo-ansible-inventory.yaml	オーバークラウドの Ansible インベントリー。
tripleo-heat-templates	レンダリングされた jinja2 テンプレートのコピーが含まれています。ソーステンプレートは **/usr/share/openstack-tripleo-heat-templates** にあります。または、CLI で **--templates** オプションを使用してテンプレートを指定できます。
tripleo-overcloud-baremetal-deployment.yaml	オーバークラウドノードをプロビジョニングするためのベアメタルデプロイメントの入力。
tripleo-overcloud-network-data.yaml	オーバークラウドネットワークをプロビジョニングするためのネットワークデプロイメントの入力。
tripleo-overcloud-roles-data.yaml	CLI の **-r** オプションで指定されたロールデータ。
tripleo-overcloud-virtual-ips.yaml	オーバークラウドネットワーク VIP をプロビジョニングするための VIP デプロイメントの入力。

7.3.9. オーバークラウドのデプロイメントの検証

デプロイされたオーバークラウドを検証します。

前提条件

- オーバークラウドをデプロイしている。

手順

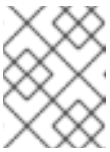
1. source コマンドで **stackrc** 認証情報ファイルを読み込みます。

```
$ source ~/stackrc
```

2. オーバークラウドのデプロイメントを検証します。

```
$ validation run \
  --group post-deployment \
  [--inventory <inventory_file>]
```

- **<inventory_file>** は ansible インベントリーファイルの名前に置き換えます。デフォルトでは、動的インベントリーは **tripleo-ansible-inventory** と呼ばれます。



注記

検証を実行すると、出力の **Reasons** 列は 79 文字に制限されます。検証結果を完全に表示するには、検証ログファイルを表示します。

3. 検証レポートの結果を確認します。

```
$ validation history get --full <UUID>
```

- **<UUID>** は、検証実行の UUID に置き換えます。
- オプション: **--full** オプションを使用して、検証実行からの詳細な出力を表示します。

重要

検証結果が **FAILED** であっても、Red Hat OpenStack Platform のデプロイや実行が妨げられることはありません。ただし、**FAILED** の検証結果は、実稼働環境で問題が発生する可能性があることを意味します。

7.3.10. オーバークラウドへのアクセス

director は、アンダークラウドからのオーバークラウドの操作に必要な認証情報を含めて認証情報ファイルを生成します。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。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

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

オーバークラウドにアクセスしていることを示すコマンドプロンプトに切り替わります。

```
(overcloud)$
```

2. アンダークラウドの操作に戻るには、**stackrc** ファイルを読み込みます。

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

アンダークラウドにアクセスしていることを示すコマンドプロンプトに切り替わります。

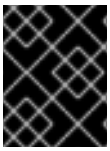
```
(undercloud)$
```

7.4. 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

この章では、事前にプロビジョニングされたノードを使用して Red Hat OpenStack Platform (RHOSP) 環境を設定するのに使用できる基本的な設定手順を説明します。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、電源管理制御を設定せずに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、ノードを管理するのに 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では、QCOW2 overcloud-full イメージに依存しないカスタムパーティションレイアウトを使用することができます。

このシナリオには、カスタム機能を持たない基本的な設定のみが含まれています。



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を組み合わせることはできません。

7.4.1. 事前にプロビジョニングされたノードの要件

事前にプロビジョニングされたノードを使用してオーバークラウドのデプロイメントを開始する前に、以下の項目が環境に存在していること確認してください。

- *4章 アンダークラウドへの director 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。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 9.2 がインストールされている必要があります。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 へのレイヤー 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7.11 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。このシナリオの要件の詳細は、「事前にプロビジョニングされたノードへの別ネットワークの使用」を参照してください。

- 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。

7.4.2. 事前にプロビジョニングされたノードでのユーザーの作成

事前にプロビジョニングされたノードを使用してオーバークラウドを設定する場合、director はオーバークラウドノードに SSH アクセス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で SSH 鍵認証のユーザーを作成し、そのユーザーに対してパスワード不要の sudo アクセス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でユーザーを作成したら、**openstack overcloud deploy** コマンドで **--overcloud-ssh-user** および **--overcloud-ssh-key** オプションを使用して、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作成することができます。

デフォルトでは、オーバークラウドの SSH ユーザーおよびオーバークラウドの SSH 鍵の値は、それぞれ **stack** ユーザーおよび **~/.ssh/id_rsa** です。**stack** ユーザーを作成するには、以下の手順を実施します。

手順

1. 各オーバークラウドノードで、**stack** 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。Controller ノードで、以下の例に示すコマンドを実行します。

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

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

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

3. 事前にプロビジョニングされた全ノードで **stack** ユーザーの作成と設定を行った後に、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。director の公開 SSH 鍵を Controller ノードにコピーするには、以下の例に示すコマンドを実行します。

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

重要

SSH 鍵をコピーするには、各オーバークラウドノードの SSH 設定で一時的に **PasswordAuthentication Yes** を設定しなければならない場合があります。SSH 鍵をコピーした後に **PasswordAuthentication No** を設定し、今後は SSH 鍵を使用して認証を行います。

7.4.3. 事前にプロビジョニングされた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。ノードを Red Hat コンテンツ配信ネットワークに登録するには、各ノードで以下の手順を実施します。**root** ユーザーとして、または **sudo** 権限を持つユーザーとしてコマンドを実行します。

重要

記載されたリポジトリーのみを有効にします。追加のリポジトリーを使用すると、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。他のリポジトリーは有効にしないでください。

手順

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の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 (RHOSP) 17.1 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. 前の手順で特定したプール ID を使用して、RHOSP 17.1 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ー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 (RHEL) リポジトリーを有効にします。

```
[root@controller-0 ~]# sudo subscription-manager repos \
 --enable=rhel-9-for-x86_64-baseos-eus-rpms \
 --enable=rhel-9-for-x86_64-appstream-eus-rpms \
 --enable=rhel-9-for-x86_64-highavailability-eus-rpms \
 --enable=openstack-17.1-for-rhel-9-x86_64-rpms \
 --enable=fast-datapath-for-rhel-9-x86_64-rpms
```

6. オーバークラウドで Red Hat Ceph Storage ノードが使用されている場合は、関連する Red Hat Ceph Storage リポジトリーを有効にします。

```
[root@cephstorage-0 ~]# sudo subscription-manager repos \
 --enable=rhel-9-for-x86_64-baseos-rpms \
 --enable=rhel-9-for-x86_64-appstream-rpms \
 --enable=openstack-deployment-tools-for-rhel-9-x86_64-rpms
```

7. Red Hat Ceph Storage ノードを除くすべてのオーバークラウドノードで RHEL バージョンをロックします。

```
[root@controller-0 ~]# subscription-manager release --set=9.2
```

8. システムを更新して、ベースシステムパッケージが最新の状態になるようにします。

```
[root@controller-0 ~]# sudo dnf 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。

7.4.4. 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操作を実施します。

手順

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/** ディレクトリー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
```

この手順により、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。

7.4.5. 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するサブネットです。オーバークラウドノードには、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** パラメーターで指定する director のパブリック API 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、SSL/TLS 通信を使用する場合に、このオプションを利用することができます。パブリック API エンドポイントを使用するオーバークラウドノードの設定の詳細は、「事前にプロビジョニングされたノードへの別ネットワークの使用」 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 と事前にプロビジョニングされたノード間の通信に対応するために、ネットワーク設定を変更しなければならない場合があります。

ネットワーク分離の使用

ネットワーク分離を使用することで、サービスをグループ化してコントロールプレーンなど特定のネットワークを使用することができます。コントロールプレーン上のノードに特定の IP アドレスを定義することも可能です。

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**) 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 OpenStack Networking (neutron) ポートを作成し、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ます。ただし、これにより、各ノードに手動で設定する 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。

ネットワーク分離を使用している場合は、予測可能な IP 戦略を実装するために、カスタム環境ファイル **deploy-ports.yaml** を作成します。次のカスタム環境ファイルの例 **deploy-ports.yaml** は、一連のリソースレジストリーマッピングとパラメーターを director に渡し、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。**NodePortMap**、**ControlPlaneVipData**、および **VipPortMap** 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 アドレスとサブネット CIDR を定義します。

```
resource_registry:
  # Deployed Virtual IP port resources
  OS::TripleO::Network::Ports::ExternalVipPort: /usr/share/openstack-tripleo-heat-
```

```
templates/network/ports/deployed_vip_external.yaml
  OS::TripleO::Network::Ports::InternalApiVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_internal_api.yaml
  OS::TripleO::Network::Ports::StorageVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_storage.yaml
  OS::TripleO::Network::Ports::StorageMgmtVipPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_vip_storage_mgmt.yaml

  # Deployed ControlPlane port resource
  OS::TripleO::DeployedServer::ControlPlanePort: /usr/share/openstack-tripleo-heat-
templates/deployed-server/deployed-neutron-port.yaml

  # Controller role port resources
  OS::TripleO::Controller::Ports::ExternalPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_external.yaml
  OS::TripleO::Controller::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_internal_api.yaml
  OS::TripleO::Controller::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage_mgmt.yaml
  OS::TripleO::Controller::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml
  OS::TripleO::Controller::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_tenant.yaml

  # Compute role port resources
  OS::TripleO::Compute::Ports::InternalApiPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_internal_api.yaml
  OS::TripleO::Compute::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml
  OS::TripleO::Compute::Ports::TenantPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_tenant.yaml

  # CephStorage role port resources
  OS::TripleO::CephStorage::Ports::StorageMgmtPort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage_mgmt.yaml
  OS::TripleO::CephStorage::Ports::StoragePort: /usr/share/openstack-tripleo-heat-
templates/network/ports/deployed_storage.yaml

parameter_defaults:
  NodePortMap: 1
    # Controller node parameters
    controller-00-rack01: 2
      ctlplane: 3
        ip_address: 192.168.24.201
        ip_address_uri: 192.168.24.201
        ip_subnet: 192.168.24.0/24
      external:
        ip_address: 10.0.0.201
        ip_address_uri: 10.0.0.201
        ip_subnet: 10.0.0.10/24
      internal_api:
        ip_address: 172.16.2.201
        ip_address_uri: 172.16.2.201
        ip_subnet: 172.16.2.10/24
      management:
        ip_address: 192.168.1.201
```

```
    ip_address_uri: 192.168.1.201
    ip_subnet: 192.168.1.10/24
  storage:
    ip_address: 172.16.1.201
    ip_address_uri: 172.16.1.201
    ip_subnet: 172.16.1.10/24
  storage_mgmt:
    ip_address: 172.16.3.201
    ip_address_uri: 172.16.3.201
    ip_subnet: 172.16.3.10/24
  tenant:
    ip_address: 172.16.0.201
    ip_address_uri: 172.16.0.201
    ip_subnet: 172.16.0.10/24
...

# Compute node parameters
compute-00-rack01:
  ctlplane:
    ip_address: 192.168.24.11
    ip_address_uri: 192.168.24.11
    ip_subnet: 192.168.24.0/24
  internal_api:
    ip_address: 172.16.2.11
    ip_address_uri: 172.16.2.11
    ip_subnet: 172.16.2.10/24
  storage:
    ip_address: 172.16.1.11
    ip_address_uri: 172.16.1.11
    ip_subnet: 172.16.1.10/24
  tenant:
    ip_address: 172.16.0.11
    ip_address_uri: 172.16.0.11
    ip_subnet: 172.16.0.10/24
...

# Ceph node parameters
ceph-00-rack01:
  ctlplane:
    ip_address: 192.168.24.101
    ip_address_uri: 192.168.24.101
    ip_subnet: 192.168.24.0/24
  storage:
    ip_address: 172.16.1.101
    ip_address_uri: 172.16.1.101
    ip_subnet: 172.16.1.10/24
  storage_mgmt:
    ip_address: 172.16.3.101
    ip_address_uri: 172.16.3.101
    ip_subnet: 172.16.3.10/24
...

# Virtual IP address parameters
ControlPlaneVipData:
  fixed_ips:
  - ip_address: 192.168.24.5
```

```
  name: control_virtual_ip
  network:
    tags: [192.168.24.0/24]
    subnets:
    - ip_version: 4
VipPortMap
  external:
    ip_address: 10.0.0.100
    ip_address_uri: 10.0.0.100
    ip_subnet: 10.0.0.100/24
  internal_api:
    ip_address: 172.16.2.100
    ip_address_uri: 172.16.2.100
    ip_subnet: 172.16.2.100/24
  storage:
    ip_address: 172.16.1.100
    ip_address_uri: 172.16.1.100
    ip_subnet: 172.16.1.100/24
  storage_mgmt:
    ip_address: 172.16.3.100
    ip_address_uri: 172.16.3.100
    ip_subnet: 172.16.3.100/24

RedisVirtualFixedIPs:
  - ip_address: 192.168.24.6
      use_neutron: false
```

1. **NodePortMap** マッピングは、ノード割り当ての名前を定義します。
2. **<node_hostname>** の形式に従う、ノードの短いホスト名。
3. ノードのネットワーク定義と IP 割り当て。ネットワークには、**ctlplane**、**external**、**internal_api**、**management**、**storage**、**storage_mgmt**、および **tenant** が含まれます。IP 割り当てには、**ip_address**、**ip_address_uri**、および **ip_subnet** が含まれます。
    - IPv4: **ip_address** と **ip_address_uri** は同じ値に設定する必要があります。
    - IPv6:
        - **ip_address**: 括弧なしの IPv6 アドレスに設定します。
        - **ip_address_uri**: **[2001:0db8:85a3:0000:0000:8a2e:0370:7334]** のように、角括弧で囲まれた IPv6 アドレスに設定します。

7.4.6. 事前にプロビジョニングされたノードへの別ネットワークの使用

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 Internal 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。詳細は、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。
- 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。

このセクションに記載する例では、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7.12 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 (Internal 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

以下のセクションでは、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を説明します。

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されたサーバーからルーティングできない場合があるため、**NodePortMap**、**ControlPlaneVipData**、および **VipPortMap** パラメーターを使用して、選択したオーバークラウドノードサブネットから IP アドレス (コントロールプレーンにアクセスするための仮想 IP アドレスを含む) を割り当てることができます。次の例は、「コントロールプレーンのネットワークの設定」 からの **deployed-ports.yaml** カスタム環境ファイルの修正版です。このネットワークアーキテクチャーに対応します。

```
parameter_defaults:
  NodePortMap:
    controller-00-rack01
      ctlplane
        ip_address: 192.168.100.2
        ip_address_uri: 192.168.100.2
        ip_subnet: 192.168.100.0/24
...
    compute-00-rack01:
      ctlplane
        ip_address: 192.168.100.3
        ip_address_uri: 192.168.100.3
```

```
      ip_subnet: 192.168.100.0/24
...
  ControlPlaneVipData:
    fixed_ips:
    - ip_address: 192.168.100.1
    name: control_virtual_ip
    network:
      tags: [192.168.100.0/24]
    subnets:
    - ip_version: 4
  VipPortMap:
    external:
      ip_address: 10.0.0.100
      ip_address_uri: 10.0.0.100
      ip_subnet: 10.0.0.100/24
....
  RedisVirtualFixedIPs:(1)
    - ip_address: 192.168.100.10
      use_neutron: false
```

1. **RedisVipPort** リソースは、**network/ports/noop.yaml** にマッピングされます。デフォルトの Redis 仮想 IP 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ので、このマッピングが必要です。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。

7.4.7. 事前にプロビジョニングされたノードのホスト名のマッピング

事前にプロビジョニングされたノードを設定する場合には、heat ベースのホスト名をそれらの実際のホスト名にマッピングして、**ansible-playbook** が解決されたホストに到達できるようにする必要があります。それらの値は、**HostnameMap** を使用してマッピングします。

手順

1. 環境ファイル (たとえば **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。

```
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** ファイルを保存します。

7.4.8. 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

すでにデプロイされているノードに Ceph を設定するには、アンダークラウドホストで以下の手順を実施します。

手順

1. アンダークラウドホストで環境変数 **OVERCLOUD_HOSTS** を作成し、変数に 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 アドレスのスペース区切りリストを設定します。

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

2. デフォルトのオーバークラウドプランの名前は **overcloud** です。別の名前を使用する場合は、環境変数 **OVERCLOUD_PLAN** を作成してカスタムの名前を保存します。

```
export OVERCLOUD_PLAN="<custom-stack-name>"
```

   - **<custom-stack-name>** を実際のスタックの名前に置き換えます。

3. **enable-ssh-admin.sh** スクリプトを実行して、Ansible が Ceph クライアントの設定に使用することのできるオーバークラウドノードのユーザーを設定します。

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

openstack overcloud deploy コマンドを実行すると、Ansible は **OVERCLOUD_HOSTS** 変数で Ceph クライアントとして定義したホストを設定します。

7.4.9. 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、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**: このオプションを使用して、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 検証を無効にします。これらの検証を無効にしないと、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、設定するには、この環境ファイルを追加します。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。

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

```
$ source ~/stackrc
(undercloud)$ openstack overcloud deploy \
  --disable-validations \
  -e /usr/share/openstack-tripleo-heat-templates/environments/deployed-server-environment.yaml \
  -e /home/stack/templates/deployed-ports.yaml \
  -e /home/stack/templates/hostname-map.yaml \
```

```
--overcloud-ssh-user stack \
--overcloud-ssh-key ~/.ssh/id_rsa \
<OTHER OPTIONS>
```

--overcloud-ssh-user および **--overcloud-ssh-key** オプションは、設定ステージ中に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、**--overcloud-ssh-user** および **--overcloud-ssh-key** (**~/.ssh/id_rsa** がデフォルト) を使用して、初回 SSH 接続用の認証情報を指定します。**--overcloud-ssh-key** オプションで指定する秘密鍵の公開を制限するために、director は Heat などのどの API サービスにもこの鍵を渡さず、director の **openstack overcloud deploy** コマンドだけ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。

7.4.10. オーバークラウドへのアクセス

director は、アンダークラウドからのオーバークラウドの操作に必要な認証情報を含めて認証情報ファイルを生成します。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。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

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

オーバークラウドにアクセスしていることを示すコマンドプロンプトに切り替わります。

```
(overcloud)$
```

2. アンダークラウドの操作に戻るには、**stackrc** ファイルを読み込みます。

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

アンダークラウドにアクセスしていることを示すコマンドプロンプトに切り替わります。

```
(undercloud)$
```

7.4.11. 事前にプロビジョニングされたノードのスケーリング

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、オーバークラウドノードのスケーリング に記載されている標準のスケーリング手順に似ています。ただし、事前プロビジョニングされたノードは、Bare Metal Provisioning サービス (ironic) および Compute サービス (nova) の標準の登録および管理プロセスを使用しないため、事前プロビジョニングされた新規ノードを追加するプロセスは異なります。

7.4.11.1. 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。

手順

1. 事前プロビジョニングされた新規ノードを準備します。詳細は、事前プロビジョニングされたノードの要件 を参照してください。

2. ノードをスケールアップします。詳細は、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。

7.4.11.2. 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場合は、オーバークラウドノードのスケーリング に記載されているスケールダウン手順に従ってください。

スケールダウン操作では、Red Hat OpenStack Platform (RHOSP) によりプロビジョニングされたノードまたは事前にプロビジョニングされたノードの両方にホスト名を使用できます。RHOSP によりプロビジョニングされたノードに UUID を使用することもできます。ただし、事前にプロビジョニングされたノードには UUID がないため、常にホスト名を使用します。

手順

1. 削除するノードの名前を取得します。

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

2. ノードを削除します。

```
$ openstack overcloud node delete --stack <overcloud> <node> [... <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、手順 1 で返された **stack_name** 列から取得した削除するノードのホスト名に置き換えます。

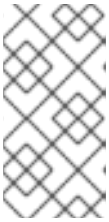
3. ノードが削除されていることを確認します。

```
$ openstack stack list
```

   削除の操作が完了すると、**オーバークラウド** スタックのステータスは **UPDATE_COMPLETE** と表示されます。

4. 削除された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 上のベアメタルサービスが、削除されたノードの電源をオフにします。事前にプロビジョニングされたノードでは、削除された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。

5. 削除されたノードをベースのオペレーティングシステム設定に再プロビジョニングして、今後、意図せずにオーバークラウドに参加しないようにします。



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。

7.4.12. 事前にプロビジョニングされたオーバークラウドの削除

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するには、「オーバークラウドスタックの削除」 で標準のオーバークラウド削除手順を参照してください。オーバークラウドを削除した後に、全ノードの電源をオフにしてからベースオペレーティングシステムの設定に再プロビジョニングします。

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。

第8章 オーバークラウドのインストール後タスクの実施

この章では、オーバークラウドを作成したすぐ後に実施するタスクを説明します。これらのタスクにより、オーバークラウドを使用可能な状態にすることができます。

8.1. オーバークラウドデプロイメントステータスの確認

オーバークラウドのデプロイメントステータスを確認するには、**openstack overcloud status** コマンドを使用します。このコマンドにより、すべてのデプロイメントステップの結果が返されます。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. デプロイメントステータスの確認コマンドを実行します。

```
$ openstack overcloud status
```

このコマンドの出力に、オーバークラウドのステータスが表示されます。

```
+-----------+---------------------+---------------------+-------------------+
| Plan Name |       Created       |       Updated       | Deployment Status |
+-----------+---------------------+---------------------+-------------------+
| overcloud | 2018-05-03 21:24:50 | 2018-05-03 21:27:59 |   DEPLOY_SUCCESS  |
+-----------+---------------------+---------------------+-------------------+
```

実際のオーバークラウドに別の名前が使用されている場合には、**--stack** 引数を使用してその名前のオーバークラウドを選択します。

```
$ openstack overcloud status --stack <overcloud_name>
```

- **<overcloud_name>** は、実際のオーバークラウドの名前に置き換えてください。

8.2. 基本的なオーバークラウドフレーバーの作成

この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、以下の手順を実施して、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。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source ~/overcloudrc
```

2. **openstack flavor create** コマンドを実行してフレーバーを作成します。以下のオプションを使用して、各フレーバーのハードウェア要件を指定します。

--disk

仮想マシンのボリュームのハードディスク容量を定義します。

--ram

仮想マシンに必要な RAM を定義します。

--vcpus

仮想マシンの仮想 CPU 数を定義します。

3. デフォルトのオーバークラウドフレーバー作成の例を以下に示します。

```
$ 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
```

注記

openstack flavor create コマンドの詳細は、**$ openstack flavor create --help** で確認してください。

8.3. デフォルトのテナントネットワークの作成

仮想マシンが内部で通信できるように、オーバークラウドにはデフォルトの Tenant ネットワークが必要です。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source ~/overcloudrc
```

2. デフォルトの Tenant ネットワークを作成します。

```
(overcloud) $ openstack network create default
```

3. ネットワーク上にサブネットを作成します。

```
(overcloud) $ openstack subnet create default --network default --gateway 172.20.1.1 --
subnet-range 172.20.0.0/16
```

4. 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 |
+----------------------+-------------+--------------------------------------+
```

これらのコマンドにより、**default** という名前の基本的な Networking サービス (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て、このネットワークから仮想マシンに IP アドレスを自動的に割り当てます。

8.4. デフォルトの FLOATING IP ネットワークの作成

オーバークラウドの外部から仮想マシンにアクセスするためには、仮想マシンに Floating IP アドレスを提供する外部ネットワークを設定する必要があります。

ここでは、2 つの手順例を示します。実際の環境に最も適した例を使用してください。

- ネイティブ VLAN (フラットネットワーク)
- 非ネイティブ VLAN (VLAN ネットワーク)

これらの例の両方で、**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。この名前は、「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。

デフォルトでは、OpenStack Networking (neutron) は、**datacentre** という物理ネットワーク名をホストノード上の **br-ex** ブリッジにマッピングします。**public** オーバークラウドネットワークを物理ネットワーク **datacentre** に接続し、これにより **br-ex** ブリッジを通じてゲートウェイが提供されます。

前提条件

- Floating IP 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 VLAN

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source ~/overcloudrc
```

2. **public** ネットワークを作成します。
   - ネイティブ VLAN 接続用に **flat** ネットワークを作成します。

```
(overcloud) $ openstack network create public --external --provider-network-type flat --provider-physical-network datacentre
```

   - 非ネイティブ VLAN 接続用に **vlan** ネットワークを作成します。

```
(overcloud) $ openstack network create public --external --provider-network-type vlan --provider-physical-network datacentre --provider-segment 201
```

     --provider-segment オプションを使用して、使用する VLAN を定義します。この例では、VLAN は **201** です。

3. Floating IP アドレスの割り当てプールを使用してサブネットを作成します。以下の例では、IP 範囲は **10.1.1.51** から **10.1.1.250** までです。

```
(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
```

   この範囲が、外部ネットワークの他の IP アドレスと競合しないようにしてください。

8.5. デフォルト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別の種別の外部ネットワーク接続で、トラフィックをプライベートテナントネットワークから External インフラストラクチャーネットワークにルーティングします。プロバイダーネットワークは Floating IP ネットワークと類似していますが、プライベートネットワークをプロバイダーネットワークに接続するのに、論理ルーターが使用されます。

ここでは、2 つの手順例を示します。実際の環境に最も適した例を使用してください。

- ネイティブ VLAN (フラットネットワーク)
- 非ネイティブ VLAN (VLAN ネットワーク)

デフォルトでは、OpenStack Networking (neutron) は、**datacentre** という物理ネットワーク名をホストノード上の **br-ex** ブリッジにマッピングします。**public** オーバークラウドネットワークを物理ネットワーク **datacentre** に接続し、これにより **br-ex** ブリッジを通じてゲートウェイが提供されます。

手順

1. source コマンドで **overcloudrc** ファイルを読み込みます。

```
$ source ~/overcloudrc
```

2. **provider** ネットワークを作成します。
   - ネイティブ VLAN 接続用に **flat** ネットワークを作成します。

```
(overcloud) $ openstack network create provider --external --provider-network-type flat --provider-physical-network datacentre --share
```

   - 非ネイティブ VLAN 接続用に **vlan** ネットワークを作成します。

```
(overcloud) $ openstack network create provider --external --provider-network-type vlan --provider-physical-network datacentre --provider-segment 201 --share
```

     - **--provider-segment** オプションを使用して、使用する VLAN を定義します。この例では、VLAN は **201** です。
     - **--share** オプションを使用して、共有ネットワークを作成します。または、テナントのみが新しいネットワークにアクセスできるように、**--share** を指定する代わりにテナントを指定します。
     - **--external** オプションを使用して、プロバイダーネットワークを外部としてマークします。Operator のみが外部ネットワークにポートを作成できます。

3. **provider** ネットワークにサブネットを追加して、DHCP サービスを提供します。

```
(overcloud) $ openstack subnet create provider-subnet --network  provider --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
```

4. 他のネットワークがプロバイダーネットワークを通じてトラフィックをルーティングできるように、ルーターを作成します。

```
(overcloud) $ openstack router create external
```

5. ルーターの外部ゲートウェイを **provider** ネットワークに設定します。

```
(overcloud) $ openstack router set --external-gateway provider external
```

6. 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、以下のコマンドを実行してサブネット **subnet1** をルーターに割り当てます。

```
(overcloud) $ openstack router add subnet external subnet1
```

このコマンドにより、**subnet1** がルーティングテーブルに追加され、**subnet1** を使用する仮想マシンからの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。

8.6. 新たなブリッジマッピングの作成

デプロイメント時に追加のブリッジをマッピングすれば、Floating IP ネットワークは **br-ex** だけでなく任意のブリッジを使用することができます。

手順

1. **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、環境ファイルに **NeutronBridgeMappings** パラメーターを追加します。

```
parameter_defaults:
  NeutronBridgeMappings: "datacentre:br-ex,floating:br-floating"
```

2. この手法により、オーバークラウドの作成後に独立した外部ネットワークを作成することができます。たとえば、**floating** 物理ネットワークにマッピングする Floating IP ネットワークを作成するには、以下のコマンドを実行します。

```
$ source ~/overcloudrc
(overcloud) $ openstack network create public --external --provider-physical-network floating
--provider-network-type vlan --provider-segment 105
(overcloud) $ openstack subnet create public --network public --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
```

8.7. 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。このセクションでは、統合テストを実施するための準備を説明します。OpenStack Integration Test Suite の使用方法の詳細は、Red Hat OpenStack Platform Integration Test Suite を使用したクラウドの検証 を参照してください。

Integration Test Suite では、テストを成功させるために、いくつかのインストール後手順が必要になります。

手順

1. 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 Internal 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
```

2. Red Hat OpenStack Platform Integration Test Suite を使用したクラウドの検証 の説明に従って統合テストを実行します。

3. 検証が完了したら、オーバークラウドの Internal 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
```

8.8. オーバークラウドの削除防止

Orchestration サービス (heat) のカスタムポリシーを設定して、オーバークラウドが削除されないようにすることができます。

スタックの削除を再度有効にするには、**custom_env_files** パラメーターから **prevent-stack-delete.yaml** ファイルを削除し、**openstack undercloud install** コマンドを実行します。

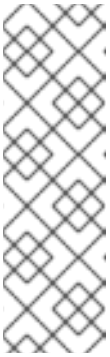
手順

1. **prevent-stack-delete.yaml** という名前の環境ファイルを作成します。

2. **HeatApiPolicies** パラメーターを設定します。

```
parameter_defaults:
  HeatApiPolicies:
    heat-deny-action:
      key: 'actions:action'
      value: 'rule:deny_everybody'
    heat-protect-overcloud:
      key: 'stacks:delete'
      value: 'rule:deny_everybody'
```

   - **heat-deny-action** は、アンダークラウドのインストールに追加する必要のあるデフォルトポリシーです。
   - **heat-protect-overcloud** ポリシーを **rule:deny_everybody** に設定して、すべてのユーザーがオーバークラウド内のスタックを削除できないようにします。



注記

オーバークラウドの保護を **rule:deny_everybody** に設定すると、以下の機能を実行できなくなることを意味します。

- オーバークラウドの削除
- 個々の Compute ノードまたは Storage ノードの削除
- Controller ノードの置き換え

3. **undercloud.conf** ファイルの **custom_env_files** パラメーターに、**prevent-stack-delete.yaml** 環境ファイルを追加します。

```
custom_env_files = prevent-stack-delete.yaml
```

4. アンダークラウドのインストールコマンドを実行して設定をリフレッシュします。

```
$ openstack undercloud install
```

第9章 基本的なオーバークラウド管理タスクの実施

この章では、オーバークラウドのライフサイクル期間中に実行しなければならない可能性がある、基本的なタスクを説明します。

9.1. SSH を使用したオーバークラウドノードへのアクセス

SSH プロトコルを使用して、各オーバークラウドノードにアクセスすることができます。

- 各オーバークラウドノードには、以前は **heat-admin** ユーザーと呼ばれていた **tripleo-admin** ユーザーが含まれます。
- アンダークラウドの **stack** ユーザーは、各オーバークラウドノードの **tripleo-admin** ユーザーに鍵ベースの SSH アクセスを行うことができます。
- すべてのオーバークラウドノードは短縮ホスト名を持ち、アンダークラウドはこのホスト名をコントロールプレーンネットワーク上の IP アドレスに解決します。それぞれの短縮ホスト名には、**.ctlplane** 接尾辞が使用されます。たとえば、**overcloud-controller-0** の短縮名は **overcloud-controller-0.ctlplane** です。

前提条件

- 稼動状態にあるコントロールプレーンネットワークと共にデプロイされたオーバークラウド

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. アクセスするノードの名前を確認します。

```
(undercloud)$ metalsmith list
```

3. **tripleo-admin** ユーザーとしてノードに接続します。

```
(undercloud)$ ssh tripleo-admin@overcloud-controller-0
```

9.2. コンテナー化されたサービスの管理

Red Hat OpenStack Platform (RHOSP) では、アンダークラウドおよびオーバークラウドノード上のコンテナー内でサービスが実行されます。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。このセクションでは、コンテナー化されたサービスを管理するためにノード上で実行することのできる、一般的なコマンドを説明します。

コンテナーとイメージのリスト表示

実行中のコンテナーをリスト表示するには、以下のコマンドを実行します。

```
$ sudo podman ps
```

コマンド出力に停止中またはエラーの発生したコンテナーを含めるには、コマンドに **--all** オプションを追加します。

```
$ sudo podman ps --all
```

コンテナーイメージをリスト表示するには、以下のコマンドを実行します。

```
$ sudo podman images
```

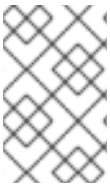
コンテナーの属性の確認

コンテナーまたはコンテナーイメージのプロパティーを表示するには、**podman inspect** コマンドを使用します。たとえば、**keystone** コンテナーを検査するには、以下のコマンドを実行します。

```
$ sudo podman inspect keystone
```

Systemd サービスを使用したコンテナーの管理

以前のバージョンの OpenStack Platform では、コンテナーは Docker およびそのデーモンで管理されていました。これで、Systemd サービスインターフェイスがコンテナーのライフサイクルを管理します。それぞれのコンテナーはサービスであり、Systemd コマンドを実行して各コンテナーに関する特定の操作を実施します。



注記

Systemd は再起動ポリシーを適用するため、Podman CLI を使用してコンテナーを停止、起動、および再起動することは推奨されません。その代わりに、Systemd サービスコマンドを使用してください。

コンテナーのステータスを確認するには、**systemctl status** コマンドを実行します。

```
$ sudo systemctl status tripleo_keystone
● tripleo_keystone.service - keystone container
   Loaded: loaded (/etc/systemd/system/tripleo_keystone.service; enabled; vendor preset: disabled)
   Active: active (running) since Fri 2019-02-15 23:53:18 UTC; 2 days ago
 Main PID: 29012 (podman)
   CGroup: /system.slice/tripleo_keystone.service
           └─29012 /usr/bin/podman start -a keystone
```

コンテナーを停止するには、**systemctl stop** コマンドを実行します。

```
$ sudo systemctl stop tripleo_keystone
```

コンテナーを起動するには、**systemctl start** コマンドを実行します。

```
$ sudo systemctl start tripleo_keystone
```

コンテナーを再起動するには、**systemctl restart** コマンドを実行します。

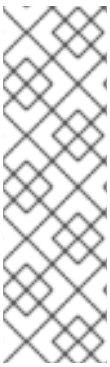
```
$ sudo systemctl restart tripleo_keystone
```

コンテナーステータスを監視するデーモンはないので、以下の状況では Systemd はほとんどのコンテナーを自動的に再起動します。

- **podman stop** コマンドの実行など、明瞭な終了コードまたはシグナル
- 起動後に podman コンテナーがクラッシュするなど、不明瞭な終了コード

- 不明瞭なシグナル
- コンテナーの起動に 1 分 30 秒以上かかった場合のタイムアウト

Systemd サービスの詳細情報は、**systemd.service のドキュメント** を参照してください。



注記

コンテナー内のサービス設定ファイルに加えた変更は、コンテナーの再起動後には元に戻ります。これは、コンテナー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** コンテナー内の **/etc/keystone/keystone.conf** を編集してコンテナーを再起動すると、そのコンテナー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ー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。

Systemd タイマーを使用した podman コンテナーの監視

Systemd タイマーインターフェイスは、コンテナーのヘルスチェックを管理します。各コンテナーのタイマーがサービスユニットを実行し、そのユニットがヘルスチェックスクリプトを実行します。

すべての OpenStack Platform コンテナーのタイマーをリスト表示するには、**systemctl list-timers** コマンドを実行し、出力を **tripleo** が含まれる行に限定します。

```
$ sudo systemctl list-timers | grep tripleo
Mon 2019-02-18 20:18:30 UTC  1s left       Mon 2019-02-18 20:17:26 UTC  1min 2s ago
tripleo_nova_metadata_healthcheck.timer           tripleo_nova_metadata_healthcheck.service
Mon 2019-02-18 20:18:34 UTC  5s left       Mon 2019-02-18 20:17:23 UTC  1min 5s ago
tripleo_keystone_healthcheck.timer                tripleo_keystone_healthcheck.service
Mon 2019-02-18 20:18:35 UTC  6s left       Mon 2019-02-18 20:17:13 UTC  1min 15s ago
tripleo_memcached_healthcheck.timer               tripleo_memcached_healthcheck.service
(...)
```

特定のコンテナータイマーのステータスを確認するには、healthcheck サービスに対して **systemctl status** コマンドを実行します。

```
$ sudo systemctl status tripleo_keystone_healthcheck.service
● tripleo_keystone_healthcheck.service - keystone healthcheck
   Loaded: loaded (/etc/systemd/system/tripleo_keystone_healthcheck.service; disabled; vendor
preset: disabled)
   Active: inactive (dead) since Mon 2019-02-18 20:22:46 UTC; 22s ago
  Process: 115581 ExecStart=/usr/bin/podman exec keystone /openstack/healthcheck (code=exited,
status=0/SUCCESS)
 Main PID: 115581 (code=exited, status=0/SUCCESS)

Feb 18 20:22:46 undercloud.localdomain systemd[1]: Starting keystone healthcheck...
Feb 18 20:22:46 undercloud.localdomain podman[115581]: {"versions": {"values": [{"status": "stable",
"updated": "2019-01-22T00:00:00Z", "..."}]}}
Feb 18 20:22:46 undercloud.localdomain podman[115581]: 300 192.168.24.1:35357 0.012 seconds
Feb 18 20:22:46 undercloud.localdomain systemd[1]: Started keystone healthcheck.
```

コンテナータイマーを停止、起動、再起動、およびコンテナータイマーのステータスを表示するには、**.timer** Systemd リソースに対して該当する **systemctl** コマンドを実行します。たとえば、**tripleo_keystone_healthcheck.timer** リソースのステータスを確認するには、以下のコマンドを実

行します。

```
$ sudo systemctl status tripleo_keystone_healthcheck.timer
● tripleo_keystone_healthcheck.timer - keystone container healthcheck
   Loaded: loaded (/etc/systemd/system/tripleo_keystone_healthcheck.timer; enabled; vendor preset:
disabled)
   Active: active (waiting) since Fri 2019-02-15 23:53:18 UTC; 2 days ago
```

healthcheck サービスは無効だが、そのサービスのタイマーが存在し有効になっている場合には、チェックは現在タイムアウトしているが、タイマーに従って実行されることを意味します。チェックを手動で開始することもできます。

注記

podman ps コマンドは、コンテナーのヘルスステータスを表示しません。

コンテナーログの確認

Red Hat OpenStack Platform 17.1 は、すべてのコンテナーからのすべての標準出力 (stdout) をログに記録し、標準エラー (stderr) は **/var/log/containers/stdout 内** のコンテナーごとに 1 つのファイルに統合されます。

また、ホストはこのディレクトリーにログローテーションを適用し、大きな容量のファイルがディスク容量を消費する問題を防ぎます。

コンテナーが置き換えられた場合には、新しいコンテナーは同じログファイルにログを出力します。**podman** はコンテナー ID ではなくコンテナー名を使用するためです。

podman logs コマンドを使用して、コンテナー化されたサービスのログを確認することもできます。たとえば、**keystone** コンテナーのログを確認するには、以下のコマンドを実行します。

```
$ sudo podman logs keystone
```

コンテナーへのアクセス

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

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

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

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

コンテナーから出るには、以下のコマンドを実行します。

```
# exit
```

9.3. 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、新たな機能を追加したり、既存の操作を変更したりすることができます。

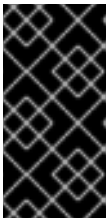
手順

手順

1. 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、「オーバークラウドの設定およびデプロイ」 に記載の手順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/overcloud-baremetal-deployed.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。director はオーバークラウドを再度作成せずに、既存のオーバークラウドに変更を加えます。



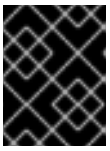
重要

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。

2. 新規環境ファイルを追加する場合には、`-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 ~/templates/network-environment.yaml \
  -e ~/templates/storage-environment.yaml \
  -e ~/templates/overcloud-baremetal-deployed.yaml \
  --ntp-server pool.ntp.org
```

このコマンドにより、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

オーバークラウドの設定に手動で変更を加えることは推奨されません。director によりこれらの変更が後で上書きされてしまう可能性があるためです。

9.4. 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 環境からご自分の Red Hat OpenStack Platform (RHOSP) 環境に仮想マシンを移行することができます。

手順

1. 既存の OpenStack 環境において、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。

```
$ openstack server image create --name <image_name> <instance_name>
$ openstack image save --file <exported_vm.qcow2> <image_name>
```

- **<instance_name>** は、インスタンスの名前に置き換えます。
- **<image_name>** は、新しいイメージの名前に置き換えます。
- **<exported_vm.qcow2>** は、エクスポートされた仮想マシンの名前に置き換えます。

2. エクスポートしたイメージをアンダークラウドノードにコピーします。

```
$ scp exported_vm.qcow2 stack@192.168.0.2:~/.
```

3. アンダークラウドに **stack** ユーザーとしてログインします。
4. **overcloudrc** 認証情報ファイルを入手します。

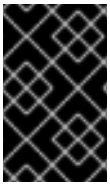
```
$ source ~/overcloudrc
```

5. 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ます。

```
(overcloud) $ openstack image create --disk-format qcow2  -file <exported_vm.qcow2> --
container-format bare <image_name>
```

6. 新規インスタンスを起動します。

```
(overcloud) $ openstack server create  --key-name default --flavor m1.demo --image
imported_image --nic net-id=net_id <instance_name>
```



重要

これらのコマンドを使用すると、各仮想マシンのディスクが既存の OpenStack 環境から新たな Red Hat OpenStack Platform にコピーできます。QCOW 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。

9.5. エフェメラル HEAT プロセスの開始

以前のバージョンの Red Hat OpenStack Platform (RHOSP) では、システムにインストールされた Heat プロセスを使用してオーバークラウドをインストールしていました。ここで、エフェメラル Heat を使用してオーバークラウドをインストールします。つまり、**heat-api** および **heat-engine** プロセスは、**deployment**、**update**、および **upgrade** コマンドによってオンデマンドで開始されます。

以前は、**openstack stack** コマンドを使用してスタックを作成および管理していました。このコマンドは、デフォルトでは使用できなくなりました。スタックに障害が発生した場合など、トラブルシューティングとデバッグの目的で、最初にエフェメラル Heat プロセスを起動して **openstack stack** コマンドを使用する必要があります。

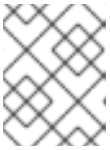
openstack overcloud tripleo launch heat コマンドを使用して、デプロイメントの外部でエフェメラル Heat を有効にします。

手順

1. エフェメラル Heat プロセスを開始します。

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-
deploy/<overcloud>/heat-launcher --restore-db
```

- **<overcloud>** を、使用するオーバークラウドの名前に置き換えます。



注記

このコマンドは、Heat プロセスを起動した後に終了します。この Heat プロセスは、引き続き podman Pod としてバックグラウンドで実行されます。

2. **ephemeral-heat** プロセスが実行されていることを確認します。

```
(undercloud)$ sudo podman pod ps
POD ID        NAME            STATUS      CREATED         INFRA ID      # OF CONTAINERS
958b141609b2  ephemeral-heat  Running     2 minutes ago  44447995dbcf  3
```

3. **OS_CLOUD** 環境をエクスポートします。

```
(undercloud)$ export OS_CLOUD=heat
```

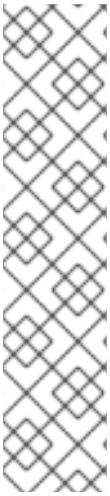
4. インストールされているスタックをリストします。

```
(undercloud)$ openstack stack list
+--------------------------------------+------------+---------+-----------------+----------------------+------
--------+
| ID                                   | Stack Name | Project | Stack Status    | Creation Time        |
Updated Time |
+--------------------------------------+------------+---------+-----------------+----------------------+------
--------+
| 761e2a54-c6f9-4e0f-abe6-c8e0ad51a76c | overcloud  | admin   | CREATE_COMPLETE |
2022-08-29T20:48:37Z | None         |
+--------------------------------------+------------+---------+-----------------+----------------------+------
--------+
```

openstack stack environment show や **openstack stack resource list** などのコマンドでデバッグできます。

5. デバッグが終了したら、エフェメラル Heat プロセスを停止します。

```
(undercloud)$ openstack tripleo launch heat --kill
```



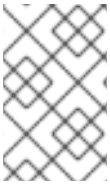
注記

Heat 環境のエクスポートが失敗することがあります。これは、**overcloudrc** などの他の資格情報が使用されている場合に発生する可能性があります。この場合、既存の環境を設定解除し、Heat 環境を取り込みます。

```
(overcloud)$ unset OS_CLOUD
(overcloud)$ unset OS_PROJECT_NAME
(overcloud)$ unset OS_PROJECT_DOMAIN_NAME
(overcloud)$ unset OS_USER_DOMAIN_NAME
(overcloud)$ OS_AUTH_TYPE=none
(overcloud)$ OS_ENDPOINT=http://127.0.0.1:8006/v1/admin
(overcloud)$ export OS_CLOUD=heat
```

9.6. 動的インベントリースクリプトの実行

Ansible ベースの自動化をご自分の Red Hat OpenStack Platform (RHOSP) 環境で実行することができます。**/home/stack/overcloud-deploy/<stack>** ディレクトリーにある **tripleo-ansible-inventory.yaml** インベントリーファイルを使用して、Ansible Play またはアドホックコマンドを実行します。



注記

アンダークラウドで Ansible Playbook または Ansible アドホックコマンドを実行する場合は、**/home/stack/tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml** インベントリーファイルを使用する必要があります。

手順

1. ノードのインベントリーを表示するには、次の Ansible アドホックコマンドを実行します。

```
(undercloud) $ ansible -i ./overcloud-deploy/<stack>/tripleo-ansible-inventory.yaml all --list
```

注記

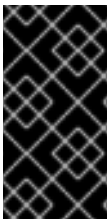
stack をデプロイされたオーバークラウドスタックの名前に置き換えます。

2. 環境上で Ansible Playbook 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インベントリーファイルへのフルパスを含めます。以下に例を示します。

```
(undercloud) $ ansible <hosts> -i ./overcloud-deploy/tripleo-ansible-inventory.yaml
<playbook> <options>
```

- **<hosts>** を、使用するホストのタイプに置き換えます。
  - 全 Controller ノードの場合には **controller**
  - 全 Compute ノードの場合には **compute**
  - オーバークラウドの全子ノードの場合には **overcloud**(たとえば、**controller** ノードおよび **compute** ノードの場合)
  - 全ノードの場合には **"*"**

- **<options>** を追加の Ansible オプションに置き換えます。
  - ホストキーの確認を省略するには、**--ssh-extra-args='-o StrictHostKeyChecking=no'** オプションを使用します。
  - Ansible の自動化を実行する SSH ユーザーを変更するには、**-u [USER]** オプションを使用します。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。
  - 特定の Ansible モジュールを使用するには、**-m [MODULE]** オプションを使用します。デフォルトは **command** で Linux コマンドを実行します。
  - 選択したモジュールの引数を定義するには、**-a [MODULE_ARGS]** オプションを使用します。

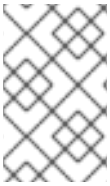


重要

オーバークラウドのカスタム Ansible 自動化は、標準のオーバークラウドスタックの一部ではありません。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。

9.7. オーバークラウドスタックの削除

オーバークラウドスタックを削除して、すべてのスタックノードのプロビジョニングを解除できます。



注記

オーバークラウドスタックを削除しても、すべてのオーバークラウドデータが消去されるわけではありません。オーバークラウドのデータをすべて消去する必要がある場合は、Red Hat サポートにお問い合わせください。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. スタック内のすべてのノードとその現在のステータスのリストを取得します。

```
(undercloud)$ openstack baremetal node list
+--------------------------------------+-------------+--------------------------------------+-------------+--
-----------------+-------------+
| UUID                                 | Name        | Instance UUID                        | Power State |
Provisioning State | Maintenance |
+--------------------------------------+-------------+--------------------------------------+-------------+--
-----------------+-------------+
| 92ae71b0-3c31-4ebb-b467-6b5f6b0caac7 | compute-0   | 059fb1a1-53ea-4060-9a47-
09813de28ea1 | power on    | active             | False       |
| 9d6f955e-3d98-4d1a-9611-468761cebabf | compute-1   | e73a4b50-9579-4fe1-bd1a-
556a2c8b504f | power on    | active             | False       |
| 8a686fc1-1381-4238-9bf3-3fb16eaec6ab | controller-0 | 6d69e48d-10b4-45dd-9776-
```

```
155a9b8ad575 | power on    | active            | False       |
| eb8083cc-5f8f-405f-9b0c-14b772ce4534 | controller-1 | 1f836ac0-a70d-4025-88a3-
bbe0583b4b8e | power on    | active            | False       |
| a6750f1f-8901-41d6-b9f1-f5d6a10a76c7 | controller-2 | e2edd028-cea6-4a98-955e-
5c392d91ed46 | power on    | active            | False       |
+--------------------------------------+-------------+--------------------------------------+-------------+--
-----------------+-------------+
```

4. オーバークラウドスタックを削除し、ノードとネットワークのプロビジョニングを解除します:

```
(undercloud)$ openstack overcloud delete -b <node_definition_file> \
 --networks-file <networks_definition_file> --network-ports <stack>
```

- **<node_definition_file>** は、ノード定義ファイルの名前 (**overcloud-baremetal-deploy.yaml** など) に置き換えます。
- **<networks_definition_file>** は、ネットワーク定義ファイルの名前 (**network_data_v2.yaml** など) に置き換えます。
- **<stack>** を、削除するスタックの名前に置き換えます。指定しない場合、デフォルトのスタックは **overcloud** です。

5. オーバークラウドを削除することを確認します。

```
Are you sure you want to delete this overcloud [y/N]?
```

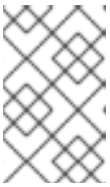
6. オーバークラウドが削除され、ノードとネットワークがプロビジョニング解除されるまで待ちます。

7. ベアメタルノードがプロビジョニング解除されていることを確認します。

```
(undercloud) [stack@undercloud-0 ~]$ openstack baremetal node list
+--------------------------------------+-------------+--------------+-------------+--------------------+----
---------+
| UUID                                 | Name        | Instance UUID | Power State | Provisioning State |
Maintenance |
+--------------------------------------+-------------+--------------+-------------+--------------------+----
---------+
| 92ae71b0-3c31-4ebb-b467-6b5f6b0caac7 | compute-0   | None          | power off   |
available         | False       |
| 9d6f955e-3d98-4d1a-9611-468761cebabf | compute-1   | None          | power off   |
available         | False       |
| 8a686fc1-1381-4238-9bf3-3fb16eaec6ab | controller-0 | None          | power off   | available
| False       |
| eb8083cc-5f8f-405f-9b0c-14b772ce4534 | controller-1 | None          | power off   | available
| False       |
| a6750f1f-8901-41d6-b9f1-f5d6a10a76c7 | controller-2 | None          | power off   | available
| False       |
+--------------------------------------+-------------+--------------+-------------+--------------------+----
---------+
```

8. スタックディレクトリーを削除します。

```
$ rm -rf ~/overcloud-deploy/<stack>
$ rm -rf ~/config-download/<stack>
```



注記

openstack overcloud deploy コマンドを使用してオーバークラウドをデプロイするときに **--output-dir** と **--working-dir** オプションを使用した場合、スタックのディレクトリーパスはデフォルトとは異なる可能性があります。

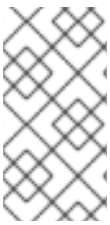
9.8. ローカルディスクのパーティションサイズの管理

パーティションサイズの設定を最適化した後もローカルディスクパーティションがいっぱいになる場合は、次のいずれかのタスクを実行します。

- 影響を受けるパーティションからファイルを手動で削除します。
- 新しい物理ディスクを追加し、LVM ボリュームグループに追加します。詳しくは、論理ボリュームの設定と管理 を参照してください。
- パーティションをオーバープロビジョニングして、残りのスペアディスク領域を使用します。このオプションが可能になるのは、デフォルトのディスク全体のオーバークラウドイメージ **overcloud-hardened-uefi-full.qcow2** がシンプールによってサポートされているためです。シンプロビジョニングされた論理ボリュームの詳細は、RHEL の **ローカルボリュームの設定と管理** ガイドの シンプロビジョニングボリューム (シンボリューム) の作成と管理 を参照してください。

警告

オーバープロビジョニングは、ファイルを手動で削除したり、新しい物理ディスクを追加したりできない場合にのみ使用してください。物理的な空き領域が不十分な場合、書き込み操作中にオーバープロビジョニングが失敗する可能性があります。



注記

新しいディスクを追加してパーティションをオーバープロビジョニングするには、サポート例外が必要です。サポートの例外、またはその他のオプションについては、Red Hat カスタマーエクスペリエンスおよびエンゲージメントチーム にお問い合わせください。

第10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。

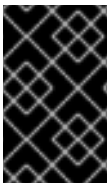
注記

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ます。

以下の表を使用して、各ノード種別のスケーリングに対するサポートを判断してください。

表10.1 各ノード種別のスケーリングサポート

ノード種別	スケールアップ	スケールダウン	備考
Controller	非対応	非対応	11章*Controller ノードの置き換え* に記載の手順を使用して、Controller 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。
Object Storage ノード	対応	対応	

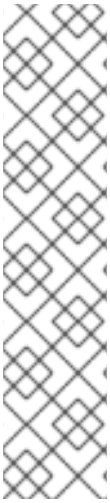


重要

オーバークラウドをスケーリングする前に、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。

10.1. オーバークラウドへのノード追加

オーバークラウドにノードを追加できます。



注記

Red Hat OpenStack Platform (RHOSP) の新規インストールには、セキュリティーエラータやバグ修正などの特定の更新が含まれていません。その結果、Red Hat Customer Portal または Red Hat Satellite Server を使用する接続環境をスケールアップすると、RPM 更新は新しいノードに適用されません。最新の更新をオーバークラウドノードに適用するには、以下のいずれかを実行する必要があります。

- スケールアウト操作後にノードのオーバークラウド更新を完了します。
- **virt-customize** ツールを使用して、スケールアウト操作の前にパッケージをベースのオーバークラウドイメージに変更します。詳細は、Red Hat ナレッジベースで Modifying the Red Hat Linux OpenStack Platform Overcloud Image with virt-customize のソリューションを参照してください。

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":"192.02.24.207"
  },
  {
     "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":"192.02.24.208"
  }
 ]
}
```

2. アンダークラウドホストに **stack** ユーザーとしてログインします。

3. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

4. 新しいノードを登録します。

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

5. 新しいノードごとにイントロスペクションプロセスを開始します。

```
$ openstack overcloud node introspect \
 --provide <node_1> [<node_2>] [<node_n>]
```

- **--provide** オプションを使用して、イントロスペクション後に指定されたすべてのノードを **available** 状態にリセットします。
- **<node_1>**、**<node_2>**、および **<node_n>** までの全ノードを、イントロスペクトする各ノードの UUID に置き換えます。

6. 新しいノードごとにイメージのプロパティーを設定します。

```
$ openstack overcloud node configure <node>
```

10.2. ベアメタルノードのスケールアップ

既存オーバークラウドのベアメタルノード数を増やすには、**overcloud-baremetal-deploy.yaml** ファイルのノード数を増やして、オーバークラウドを再デプロイします。

前提条件

- 新しいベアメタルノードが登録され、イントロスペクトされ、プロビジョニングとデプロイメントに使用できる。詳細は、オーバークラウドのノードの登録 と ベアメタルノードハードウェアのインベントリーの作成 を参照してください。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. ベアメタルノードのプロビジョニングに使用する **overcloud-baremetal-deploy.yaml** ノード定義ファイルを開きます。
4. スケールアップするロールの **count** パラメーターを増やします。たとえば、以下の設定では、Object Storage ノード数を 4 に増やします。

```
- name: Controller
  count: 3
- name: Compute
  count: 10
- name: ObjectStorage
  count: 4
```

5. オプション: 新規ノードに予測可能なノード配置を設定します。たとえば、以下の設定を使用して、**node03** に新しい Object Storage ノードをプロビジョニングします。

```
- name: ObjectStorage
  count: 4
  instances:
  - hostname: overcloud-objectstorage-0
    name: node00
  - hostname: overcloud-objectstorage-1
    name: node01
  - hostname: overcloud-objectstorage-2
    name: node02
  - hostname: overcloud-objectstorage-3
    name: node03
```

6. オプション: 新しいノードに割り当てるその他の属性を定義します。ノード定義ファイルでノード属性を設定するために使用できるプロパティーの詳細は、ベアメタルノードのプロビジョニング属性 を参照してください。

7. Object Storage サービス (swift) とディスク全体のオーバークラウドイメージ **overcloud-hardened-uefi-full** を使用する場合に、ディスクのサイズと **/var** および **/srv** のストレージ要件に基づいて **/srv** パーティションのサイズを設定します。詳細は、Object Storage サービスのディスクパーティション全体の設定 を参照してください。

8. オーバークラウドノードをプロビジョニングします。

```
$ openstack overcloud node provision \
  --stack <stack> \
  --network-config \
  --output <deployment_file> \
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```

   - **<stack>** を、ベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。指定しない場合、デフォルトは **overcloud** です。
   - **--network-config** 引数を含めて、**cli-overcloud-node-network-config.yaml** Ansible Playbook にネットワーク定義を提供します。
   - **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml)**。

注記

Red Hat OpenStack Platform 16.2 から 17.1 にアップグレードした場合は、**openstack overcloud node provision** コマンドでアップグレードプロセス中に作成または更新した YAML ファイルを含める必要があります。たとえば、**/home/stack/templates/overcloud-baremetal-deployed.yaml** ファイルの代わりに、**/home/stack/tripleo-[stack]-baremetal-deploy.yaml** ファイルを使用します。詳細は、**16.2 から 17.1 へのアップグレードフレームワーク** での オーバークラウドの導入と準備の実行 を参照してください。

9. 別のターミナルでプロビジョニングの進捗をモニタリングします。プロビジョニングが成功すると、ノードの状態が **available** から **active** に変わります。

```
$ watch openstack baremetal node list
```

10. 生成された **overcloud-baremetal-deployed.yaml** ファイルを他の環境ファイルと共にスタックに追加し、オーバークラウドをデプロイします。

```
$ openstack overcloud deploy --templates \
  -e [your environment files] \
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
  --deployed-server \
  --disable-validations \
  ...
```

10.3. ベアメタルノードのスケールダウン

オーバークラウド内のベアメタルノードの数を縮小するには、ノード定義ファイルでスタックから削除するノードにタグを付け、オーバークラウドを再デプロイしてから、オーバークラウドからベアメタルノードを削除します。

前提条件

- アンダークラウドの正常なインストール。詳細は、Installing director on the undercloud を参照してください。
- オーバークラウドの正常なデプロイメント。詳細は、プリ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 を参照してください。
- Object Storage ノードを置き換える場合は、削除するノードから新しい置き換えノードにデータを複製します。新しいノードでレプリケーションのパスが完了するまで待機します。**/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 (swift) は、以下の例のようにエントリーをログに追加します。

```
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
```

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

3. スケールダウンするロールについて、**overcloud-baremetal-deploy.yaml** ファイルの **count** パラメーターの数を減らします。
4. スタックから削除する各ノードの **hostname** と **name** を定義します (ロールの **instances** 属性で定義されていない場合)。
5. 削除するノードに属性 **provisioned: false** を追加します。たとえば、スタックからノード **overcloud-objectstorage-1** を削除するには、**overcloud-baremetal-deploy.yaml** ファイルに以下のスニペットを追加します。

```
- name: ObjectStorage
  count: 3
  instances:
  - hostname: overcloud-objectstorage-0
    name: node00
  - hostname: overcloud-objectstorage-1
    name: node01
    # Removed from cluster due to disk failure
    provisioned: false
  - hostname: overcloud-objectstorage-2
    name: node02
  - hostname: overcloud-objectstorage-3
    name: node03
```

オーバークラウドの再デプロイ後、**provisioned: false** 属性で定義したノードがスタックには存在しなくなります。ただし、これらのノードは provisioned の状態で稼働したままです。

注記

スタックから一時的にノードを削除するには、属性 **provisioned: false** でオーバークラウドをデプロイし、属性 **provisioned: true** でオーバークラウドを再デプロイして、ノードをスタックに戻します。

6. オーバークラウドからノードを削除します。

```
$ openstack overcloud node delete \
  --stack <stack> \
  --baremetal-deployment \
   /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- **<stack>** を、ベアメタルノードがプロビジョニングされるスタックの名前に置き換えます。指定しない場合、デフォルトは **overcloud** です。

注記

スタックから削除するノードを、**openstack overcloud node delete** コマンドのコマンド引数に含めないでください。

7. ironic ノードを削除します。

```
$ openstack baremetal node delete <IRONIC_NODE_UUID>
```

IRONIC_NODE_UUID は、ノードの UUID に置き換えます。

8. オーバークラウドノードをプロビジョニングして、デプロイメントコマンドに含める heat 環境ファイルを更新して生成します。

```
$ openstack overcloud node provision \
  --stack <stack> \
  --output <deployment_file> \
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```

- **<deployment_file>** は、デプロイメントコマンドに含めるために生成する heat 環境ファイルの名前に置き換えます (例 **:/home/stack/templates/overcloud-baremetal-deployed.yaml)**。

9. プロビジョニングコマンドによって生成された **overcloud-baremetal-deployed.yaml** ファイルを他の環境ファイルと共にスタックに追加し、オーバークラウドをデプロイします。

```
$ openstack overcloud deploy \
  ...
  -e /usr/share/openstack-tripleo-heat-templates/environments \
  -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
  --deployed-server \
  --disable-validations \
  ...
```

10.4. RED HAT 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 Red Hat Ceph Storage ノードを置き換えることができます。詳細は、director を使用した Red Hat Ceph Storage および Red Hat OpenStack Platform のデプロイ を参照してください。

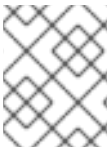
10.5. スキップデプロイ識別子の使用

スタック更新操作中に、puppet はデフォルトで、すべてのマニフェストを再適用します。その結果、必要のない操作に時間がかかることがあります。

デフォルトの操作をオーバーライドするには、**skip-deploy-identifier** オプションを使用します。

```
openstack overcloud deploy --skip-deploy-identifier
```

デプロイメントコマンドで **DeployIdentifier** パラメーターの一意の ID を生成するのを希望しない場合は、このオプションを使用します。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。



注記

puppet マニフェストまたは hierdata に変更がある場合、**--skip-deploy-identifier** が指定されている場合でも、puppet はすべてのマニフェストを再適用します。

10.6. 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、新規ノードをスケーリングする場合に、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するのに役立ちます。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には、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用すると、スタックがハングアップしたり、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、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ことがあります。

- 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。

- 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 Compute 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 Compute 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。Compute ノードのブラックリスト登録が解除される次回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 パラメーターを削除しないでください。パラメーターを削除すると、オーバークラウドの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11章 CONTROLLER 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 Controller 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 Controller ノードをクラスターから削除して新しい Controller 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、Controller ノードを置き換えます。Controller 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、Controller 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。Controller ノード 1 台の場合には、この手順は使用しないでください。

11.1. CONTROLLER 置き換えの準備

オーバークラウドの Controller ノードを置き換える前に、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。このチェックをしておくと、Controller 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、Controller 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
$ openstack overcloud status
```

overcloud スタックの deployment ステータスが **DEPLOY_SUCCESS** である場合にのみ続行します。

2. データベースクライアントツールをインストールします。

```
$ sudo dnf -y install mariadb
```

3. root ユーザーのデータベースへのアクセス権限を設定します。

```
$ sudo cp /var/lib/config-data/puppet-generated/mysql/root/.my.cnf /root/.
```

4. 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。

```
$ mkdir /home/stack/backup
$ sudo mysqldump --all-databases --quick --single-transaction | gzip >
/home/stack/backup/dump_db_undercloud.sql.gz
```

5. 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。

```
$ df -h
```

6. 新規 Controller ノード用に IP アドレスを再利用する場合は、古い Controller が使用したポートを削除するようにしてください。

```
$ openstack port delete <port>
```

7. Controller 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 Controller 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 のステータスを表示します。

```
$ ssh tripleo-admin@192.168.0.47 'sudo pcs status'
```

この出力には、既存のノードで動作中のサービスおよび障害の発生しているノードで停止しているサービスがすべて表示されます。

8. 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 Controller 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、Controller 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。

```
$ for i in 192.168.0.46 192.168.0.47 ; do echo "*** $i ***" ; ssh tripleo-admin@$i "sudo podman exec \$(sudo podman ps --filter name=galera-bundle -q) mysql -e \"SHOW STATUS LIKE 'wsrep_local_state_comment'; SHOW STATUS LIKE 'wsrep_cluster_size';\""; done
```

9. RabbitMQ のステータスをチェックします。たとえば、実行中の Controller 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 RabbitMQ のステータスを表示します。

```
$ ssh tripleo-admin@192.168.0.47 "sudo podman exec \$(sudo podman 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。

10. フェンシングが有効な場合は、無効にします。たとえば、実行中の Controller 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のステータスを確認します。

```
$ ssh tripleo-admin@192.168.0.47 "sudo pcs property show stonith-enabled"
```

フェンシングを無効にするには、以下のコマンドを実行します。

```
$ ssh tripleo-admin@192.168.0.47 "sudo pcs property set stonith-enabled=false"
```

11. 障害が発生したコントローラーノードにログインし、実行中のすべての **nova_*** コンテナーを停止します。

```
$ sudo systemctl stop tripleo_nova_api.service
$ sudo systemctl stop tripleo_nova_api_cron.service
$ sudo systemctl stop tripleo_nova_conductor.service
```

```
$ sudo systemctl stop tripleo_nova_metadata.service
$ sudo systemctl stop tripleo_nova_scheduler.service
```

12. オプション: virt ドライバーとして Bare Metal サービス (ironic) を使用している場合は、削除する Controller に **instances.host** が設定されているベアメタルインスタンスのセルデータベースのサービスエントリーを手動で更新する必要があります。レッドハットサポートにお問い合わせください。

注記

Bare Metal サービス (ironic) を virt ドライバーとして使用する場合のセルデータベースのこの手動更新は、BZ2017980 が完了するまで、ノードのリバランスを確保するための一時的な回避策です。

11.2. CEPH MONITOR デーモンの削除

Controller 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。

注記

クラスターに新しい Controller ノードを追加すると、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ノードに接続します。

```
$ ssh tripleo-admin@192.168.0.47
```

2. Ceph mon サービスを一覧表示します。

```
$ sudo systemctl --type=service | grep ceph
ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@crash.controller-0.service          loaded active
running Ceph crash.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-0.mufglq.service     loaded
active running Ceph mgr.controller-0.mufglq for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-0.service            loaded active
running Ceph mon.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@rgw.rgw.controller-0.ikaevh.service loaded
active running Ceph rgw.rgw.controller-0.ikaevh for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
```

3. Ceph mon サービスを停止します。

```
$ sudo systemtctl stop ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-
0.service
```

4. Ceph mon サービスを無効にします。

```
$ sudo systemctl disable ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mon.controller-
0.service
```

5. 置き換える Controller ノードとの接続を終了します。

6. SSH を使用して、同じクラスター内の別のコントローラーノードに接続します。

```
$ ssh tripleo-admin@192.168.0.46
```

7. Ceph 仕様ファイルは、この手順の後半で変更および適用します。ファイルを操作するには、エクスポートする必要があります。

```
$ sudo cephadm shell --ceph orch ls --export > spec.yaml
```

8. クラスターから monitor を削除します。

```
$ sudo cephadm shell -- ceph mon remove controller-0
  removing mon.controller-0 at [v2:172.23.3.153:3300/0,v1:172.23.3.153:6789/0], there will be
2 monitors
```

9. コントローラーノードから切断し、クラスターから削除するコントローラーノードに再度ログインします。

```
$ ssh tripleo-admin@192.168.0.47
```

10. Ceph mgr サービスを一覧表示します。

```
$ sudo systemctl --type=service | grep ceph
ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@crash.controller-0.service           loaded active
running Ceph crash.controller-0 for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-0.mufglq.service     loaded
active running Ceph mgr.controller-0.mufglq for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
  ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@rgw.rgw.controller-0.ikaevh.service loaded
active running Ceph rgw.rgw.controller-0.ikaevh for 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
```

11. Ceph mgr サービスを停止します。

```
$ sudo systemctl stop ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-
0.mufglq.service
```

12. Ceph mgr サービスを無効にします。

```
$ sudo systemctl disable ceph-4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31@mgr.controller-
0.mufglq.service
```

13. **cephadm** シェルを起動します。

```
$ sudo cephadm shell
```

14. コントローラーノードの Ceph mgr サービスがクラスターから削除されていることを確認します。

```
$ ceph -s
cluster:
    id:     b9b53581-d590-41ac-8463-2f50aa985001
    health: HEALTH_OK

  services:
```

```
    mon: 2 daemons, quorum controller-2,controller-1 (age 2h)
    mgr: controller-2(active, since 20h), standbys: controller1-1
    osd: 15 osds: 15 up (since 3h), 15 in (since 3h)

  data:
    pools:   3 pools, 384 pgs
    objects: 32 objects, 88 MiB
    usage:   16 GiB used, 734 GiB / 750 GiB avail
    pgs:     384 active+clean
```

Ceph mgr サービスが正常に削除された場合、ノードは一覧表示されません。

15. Red Hat Ceph Storage 仕様をエクスポートします。

```
$ ceph orch ls --export > spec.yaml
```

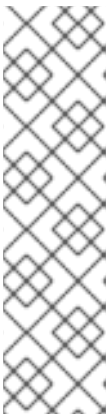
16. **spec.yaml** 仕様ファイルで、ホストのすべてのインスタンス (**controller-0** など) を **service_type: mon** および **service_type: mgr** から削除します。

17. Red Hat Ceph Storage 仕様を再適用します。

```
$ ceph orch apply -i spec.yaml
```

18. 削除されたホストに Ceph デーモンが残っていないことを確認します。

```
$ ceph orch ps controller-0
```



注記

デーモンが存在する場合は、次のコマンドを使用してそれらを削除します。

```
$ ceph orch host drain controller-0
```

ceph orch host drain コマンドを実行する前に、**/etc/ceph** の内容をバックアップします。**ceph orch host drain** コマンドを実行した後、内容を復元します。https://bugzilla.redhat.com/show_bug.cgi?id=2153827 が解決されるまで、**ceph orch host drain** コマンドを実行する前にバックアップする必要があります。

19. Red Hat Ceph Storage クラスターから **controller-0** ホストを削除します。

```
$ ceph orch host rm controller-0
  Removed host 'controller-0'
```

20. cephadm シェルを終了します。

```
$ exit
```

関連情報

- systemd を使用した Red Hat Ceph Storage サービスの制御の詳細は、Ceph のプロセス管理について を参照してください。

- Red Hat Ceph Storage 仕様ファイルの編集と適用の詳細は、サービス仕様を使用した Ceph モニターデーモンのデプロイ を参照してください。

11.3. CONTROLLER ノードを置き換えるためのクラスター準備

ノードを置き換える前に、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ようにしてください。

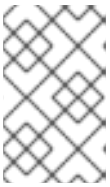
手順

1. Controller ノードの IP アドレスリストを表示するには、以下のコマンドを実行します。

```
(undercloud)$ metalsmith -c Hostname -c "IP Addresses" list
+------------------------+-----------------------+
| Hostname               | IP Addresses          |
+------------------------+-----------------------+
| 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. ノードにログインし、ペースメーカーのステータスを確認します。Pacemaker が実行中の場合は、**pcs cluster** コマンドを使用して Pacemaker を停止します。この例では、**overcloud-controller-0** で Pacemaker を停止します。

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs status | grep -w Online | grep -w overcloud-controller-0"
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster stop overcloud-controller-0"
```



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. ノードで Pacemaker を停止したら、Pacemaker クラスターからノードを削除します。以下に示すコマンドの例では、**overcloud-controller-1** にログインし、**overcloud-controller-0** を削除しています。

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster node remove overcloud-controller-0"
```

置き換えるノードにアクセスすることができない場合には (ハードウェア障害などの理由により)、**pcs** コマンドを実行する際にさらに **--skip-offline** および **--force** オプションを指定して、ノードをクラスターから強制的に削除します。

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs cluster node remove overcloud-controller-0 --skip-offline --force"
```

4. Pacemaker クラスターからノードを削除したら、Pacemaker の既知のホスト一覧からノードを削除します。

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs host deauth overcloud-controller-0"
```

ノードにアクセス可能かどうかにかかわらず、このコマンドを実行することができます。

5. 置き換え後に新しいコントローラーノードで正しい STONITH フェンシングデバイスが使用されるようにするには、以下のコマンドを入力してノードからデバイスを削除します。

```
(undercloud) $ ssh tripleo-admin@192.168.0.45 "sudo pcs stonith delete <stonith_resource_name>"
```

- **<stonith_resource_name>** は、ノードに対応する STONITH リソースの名前に置き換えます。リソース名には、**<resource_agent>-<host_mac>** という形式が使用されます。リソースエージェントおよびホストの MAC アドレスは、**fencing.yaml** ファイルの **FencingConfig** セクションに記載されています。

6. 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 Controller ノードを選択して、その Controller 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 tripleo-admin@192.168.0.45 "sudo pcs resource unmanage galera-bundle"
```

7. 置き換えられたコントローラーノードの OVN ノースバウンドデータベースサーバーをクラスターから削除します。

   a. 置き換える OVN ノースバウンドデータベースサーバーのサーバー ID を取得します。

   ```
   $ ssh tripleo-admin@<controller_ip> sudo podman exec ovn_cluster_north_db_server ovs-appctl -t /var/run/ovn/ovnnb_db.ctl cluster/status OVN_Northbound 2>/dev/null|grep -A4 Servers:
   ```

   <controller_ip> を、任意のアクティブなコントローラーノードの IP アドレスに置き換えます。

   以下のような出力が表示されるはずです。

   ```
   Servers:
   96da (96da at tcp:172.17.1.55:6643) (self) next_index=26063 match_index=26063 466b (466b at tcp:172.17.1.51:6643) next_index=26064 match_index=26063 last msg 2936 ms ago
   ba77 (ba77 at tcp:172.17.1.52:6643) next_index=26064 match_index=26063 last msg 2936 ms ago
   ```

   この例では、**172.17.1.55** が置き換えられるコントローラーノードの内部 IP アドレスであるため、ノースバウンドデータベースサーバー ID は **96da** になります。

   b. 前述の手順で取得したサーバー ID を使用し、次のコマンドを実行して OVN ノースバウンドデータベースサーバーを削除します。

   ```
   $ ssh tripleo-admin@172.17.1.52 sudo podman exec ovn_cluster_north_db_server ovs-appctl -t /var/run/ovn/ovnnb_db.ctl cluster/kick OVN_Northbound 96da
   ```

この例では、**172.17.1.52** を任意のアクティブなコントローラーノードの IP アドレスに置き換え、**96da** を OVN ノースバウンドデータベースサーバーのサーバー ID に置き換えます。

8. 置き換えられたコントローラーノードの OVN サウスバウンドデータベースサーバーをクラスターから削除します。

   a. 置き換える OVN サウスバウンドデータベースサーバーのサーバー ID を取得します。

   ```
   $ ssh tripleo-admin@<controller_ip> sudo podman exec ovn_cluster_south_db_server ovs-appctl -t /var/run/ovn/ovnsb_db.ctl cluster/status OVN_Southbound 2>/dev/null|grep -A4 Servers:
   ```

   <controller_ip> を、任意のアクティブなコントローラーノードの IP アドレスに置き換えます。

   以下のような出力が表示されるはずです。

   ```
   Servers:
   e544 (e544 at tcp:172.17.1.55:6644) last msg 42802690 ms ago
   17ca (17ca at tcp:172.17.1.51:6644) last msg 5281 ms ago
   6e52 (6e52 at tcp:172.17.1.52:6644) (self)
   ```

   この例では、**172.17.1.55** が置き換えられるコントローラーノードの内部 IP アドレスであるため、サウスバウンドデータベースサーバー ID は **e544** になります。

   b. 前述の手順で取得したサーバー ID を使用し、次のコマンドを実行して OVN サウスバウンドデータベースサーバーを削除します。

   ```
   $ ssh tripleo-admin@172.17.1.52 sudo podman exec ovn_cluster_south_db_server ovs-appctl -t /var/run/ovn/ovnsb_db.ctl cluster/kick OVN_Southbound e544
   ```

   この例では、**172.17.1.52** を任意のアクティブなコントローラーノードの IP アドレスに置き換え、**e544** を OVN サウスバウンドデータベースサーバーのサーバー ID に置き換えます。

9. クラスターの再参加を防ぐために、以下のクリーンアップコマンドを実行します。**<replaced_controller_ip>** を、置き換えるコントローラーノードの IP アドレスに置き換えます。

```
$ ssh tripleo-admin@<replaced_controller_ip> sudo systemctl disable --now tripleo_ovn_cluster_south_db_server.service tripleo_ovn_cluster_north_db_server.service

$ ssh tripleo-admin@<replaced_controller_ip> sudo rm -rfv /var/lib/openvswitch/ovn/.ovn* /var/lib/openvswitch/ovn/ovn*.db
```

11.4. IDM からコントローラーノードを削除する

ノードが TLSe で保護されている場合は、IdM (Identity Management) サーバーからホストと DNS エントリーを削除する必要があります。

1. IdM サーバーで、IDM からコントローラーノードのすべての DNS エントリーを削除します。

```
[root@server ~]# ipa dnsrecord-del
```

```
Record name: <host name> 1
Zone name: <domain name> 2
No option to delete specific record provided.
Delete all? Yes/No (default No): yes
------------------------
Deleted record "<host name>"
```

- **<host name>** をコントローラーのホスト名に置き換えます。
- **<domain name>** をコントローラーのドメイン名に置き換えます。

2. IdM サーバーで、IdM LDAP サーバーからクライアントホストエントリーを削除します。これにより、すべてのサービスが削除され、そのホストに発行されたすべての証明書が無効になります。

```
[root@server ~]# ipa host-del client.idm.example.com
```

11.5. ブートストラップ CONTROLLER ノードの置き換え

ブートストラップ操作に使用する Controller ノードを置き換え、ノード名を維持するには、以下の手順を実施して、置き換えプロセス後にブートストラップ Controller ノードの名前を設定します。

手順

1. 以下のコマンドを実行して、ブートストラップコントローラーノードの名前を確認します。

```
ssh tripleo-admin@<controller_ip> "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

- **<controller_ip>** を、任意のアクティブな Controller ノードの IP アドレスに置き換えます。

2. 環境ファイルに **ExtraConfig** パラメーターおよび **AllNodesExtraMapData** パラメーターが含まれているか確認します。パラメーターが設定されていない場合は、以下の環境ファイル **~/templates/bootstrap-controller.yaml** を作成し、以下の内容を追加します。

```
parameter_defaults:
  ExtraConfig:
    pacemaker_short_bootstrap_node_name: NODE_NAME
    mysql_short_bootstrap_node_name: NODE_NAME
  AllNodesExtraMapData:
    ovn_dbs_bootstrap_node_ip: NODE_IP
    ovn_dbs_short_bootstrap_node_name: NODE_NAME
```

- **NODE_NAME** を、置き換えプロセス後にブートストラップ操作に使用する既存のコントローラーノードの名前に置き換えます。
- **NODE_IP** を、**NODE_NAME** で指定された Controller にマップされた IP アドレスに置き換えます。名前を取得するには、次のコマンドを実行します。

```
sudo hiera -c /etc/puppet/hiera.yaml ovn_dbs_node_ips
```

環境ファイルに **ExtraConfig** パラメーターおよび **AllNodesExtraMapData** パラメーターがすでに含まれている場合は、この手順で示す行だけを追加します。

ブートストラップ Controller ノード置き換えのトラブルシューティングに関する情報は、アーティクル Replacement of the first controller node fails at step 1 if the same hostname is used for a new node を参照してください。

11.6. コントローラーノードのプロビジョニング解除と削除

Controller ノードは、プロビジョニングを解除して削除できます。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. **overcloud-controller-0** ノードの UUID を特定します。

```
(undercloud)$ NODE=$(metalsmith -c UUID -f value show overcloud-controller-0)
```

3. ノードをメンテナンスモードに切り替えます。

```
$ openstack baremetal node maintenance set $NODE
```

4. **overcloud-baremetal-deploy.yaml** ファイルをコピーします。

```
$ cp /home/stack/templates/overcloud-baremetal-deploy.yaml
/home/stack/templates/unprovision_controller-0.yaml
```

5. **unprovision_controller-0.yaml** ファイルで、コントローラー数を減らして、置き換えるコントローラーノードのプロビジョニングを解除します。この例では、数を **3** から **2** に減らします。**controller-0** ノードを **instances** ディクショナリーに移動し、**provisioned** パラメーターを **false** に設定します。

```
- name: Controller
  count: 2
  hostname_format: controller-%index%
  defaults:
    resource_class: BAREMETAL.controller
    networks:
      [ ... ]
  instances:
  - hostname: controller-0
    name: <IRONIC_NODE_UUID_or_NAME>
    provisioned: false
- name: Compute
  count: 2
  hostname_format: compute-%index%
  defaults:
    resource_class: BAREMETAL.compute
    networks:
      [ ... ]
```

6. **node unprovision** コマンドを実行します。

```
$ openstack overcloud node delete \
  --stack overcloud \
  --baremetal-deployment /home/stack/templates/unprovision_controller-0.yaml
```

```
The following nodes will be unprovisioned:
+--------------+-----------------------+--------------------------------------+
| hostname     | name                  | id                                   |
+--------------+-----------------------+--------------------------------------+
| controller-0 | baremetal-35400-leaf1-2 | b0d5abf7-df28-4ae7-b5da-9491e84c21ac |
+--------------+-----------------------+--------------------------------------+

Are you sure you want to unprovision these overcloud nodes and ports [y/N]?
```

7. オプション: ironic ノードを削除します。

```
$ openstack baremetal node delete <IRONIC_NODE_UUID>
```

- **IRONIC_NODE_UUID** は、ノードの UUID に置き換えます。

11.7. オーバークラウドへの新しいコントローラーノードのデプロイ

オーバークラウドに新しいコントローラーノードをデプロイするには、以下の手順を実行します。

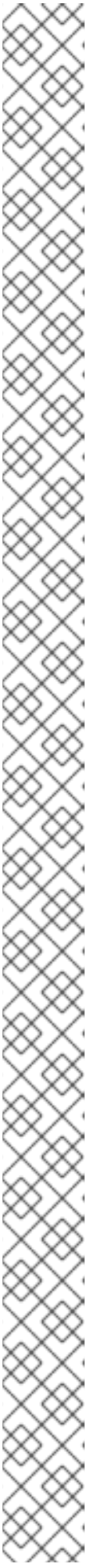
前提条件

- 新しいコントローラーノードが登録、検査、およびタグ付けされていて、プロビジョニングの準備ができている。詳細は、ベアメタルオーバークラウドノードのプロビジョニング を参照してください。

手順

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。
   $ source ~/stackrc
3. 元の **overcloud-baremetal-deploy.yaml** 環境ファイルを使用してオーバークラウドをプロビジョニングします。

```
$ openstack overcloud node provision
  --stack overcloud
  --network-config
  --output /home/stack/templates/overcloud-baremetal-deployed.yaml
  /home/stack/templates/overcloud-baremetal-deploy.yaml
```



注記

同じスケジューリング、配置、または IP アドレスを使用する場合は **overcloud-baremetal-deploy.yaml** 環境ファイルを編集できます。**instances** セクションで、新しい controller-0 インスタンスのホスト名、名前、およびネットワークを設定します。以下に例を示します。

```
- name: Controller
  count: 3
  hostname_format: controller-%index%
  defaults:
    resource_class: BAREMETAL.controller
    networks:
    - network: external
      subnet: external_subnet
    - network: internal_api
      subnet: internal_api_subnet01
    - network: storage
      subnet: storage_subnet01
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
    - network: tenant
      subnet: tenant_subnet01
    network_config:
      template: templates/multiple_nics/multiple_nics_dvr.j2
      default_route_network:
      - external
  instances:
  - hostname: controller-0
    name: baremetal-35400-leaf1-2
    networks:
    - network: external
      subnet: external_subnet
      fixed_ip: 10.0.0.224
    - network: internal_api
      subnet: internal_api_subnet01
      fixed_ip: 172.17.0.97
    - network: storage
      subnet: storage_subnet01
      fixed_ip: 172.18.0.24
    - network: storage_mgmt
      subnet: storage_mgmt_subnet01
      fixed_ip: 172.19.0.129
    - network: tenant
      subnet: tenant_subnet01
      fixed_ip: 172.16.0.11
- name: Compute
  count: 2
  hostname_format: compute-%index%
  defaults:
    [ ... ]
```

ノードがプロビジョニングされたら、**overcloud-baremetal-deploy.yaml** ファイルから **instance** セクションを削除します。

4. 新しいコントローラーノードで **cephadm** ユーザーを作成するには、新しいホストの情報を含む基本的な Ceph 仕様をエクスポートします。

```
$ openstack overcloud ceph spec --stack overcloud \
  /home/stack/templates/overcloud-baremetal-deployed.yaml \
  -o ceph_spec_host.yaml
```

注記

環境でカスタムロールを使用している場合は、**--roles-data** オプションを含めます。

5. **cephadm** ユーザーを新しいコントローラーノードに追加します。

```
$ openstack overcloud ceph user enable \
  --stack overcloud ceph_spec_host.yaml
```

6. Controller ノードにログインし、新規ロールを Ceph クラスターに追加します。

```
$ sudo cephadm shell \
  -- ceph orch host add controller-3 <IP_ADDRESS> <LABELS>
192.168.24.31 _admin mon mgr
Inferring fsid 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
Using recent ceph image undercloud-0.ctlplane.redhat.local:8787/rh-
osbs/rhceph@sha256:3075e8708792ebd527ca14849b6af4a11256a3f881ab09b837d7af0f8b2
102ea
Added host 'controller-3' with addr '192.168.24.31'
```

- <IP_ADDRESS> をコントローラーノードの IP アドレスに置き換えます。
- <LABELS> を必要な Ceph ラベルに置き換えます。

7. **openstack overcloud deploy** コマンドを再実行します。

```
$ openstack overcloud deploy --stack overcloud --templates \
    -n /home/stack/templates/network_data.yaml \
    -r /home/stack/templates/roles_data.yaml \
    -e /home/stack/templates/overcloud-baremetal-deployed.yaml \
    -e /home/stack/templates/overcloud-networks-deployed.yaml \
    -e /home/stack/templates/overcloud-vips-deployed.yaml \
    -e /home/stack/templates/bootstrap_node.yaml \
    -e [ ... ]
```

注記

置き換え用のコントローラーノードがブートストラップノードの場合には、**bootstrap_node.yaml** 環境ファイルを含めます。

11.8. 新しいコントローラーノードへの CEPH サービスのデプロイ

新しいコントローラーノードをプロビジョニングし、Ceph モニターサービスが実行されたら、コントローラーノードに **mgr**、**rgw**、および **osd** Ceph サービスをデプロイできます。

前提条件

- 新しいコントローラーノードがプロビジョニングされ、Ceph モニターサービスが実行されている。

手順

1. **spec.yml** 環境ファイルを変更し、以前のコントローラーノード名を新しいコントローラーノード名に置き換えます。

```
$ cephadm shell -- ceph orch ls --export > spec.yml
```

注記

基本的な Ceph 環境ファイル **ceph_spec_host.yaml** には、必要なすべてのクラスター情報が含まれていないため、使用しないでください。

2. 変更した Ceph 仕様ファイルを適用します。

```
$ cat spec.yml | sudo cephadm shell -- ceph orch apply -i -
Inferring fsid 4cf401f9-dd4c-5cda-9f0a-fa47fbf12b31
Using recent ceph image undercloud-0.ctlplane.redhat.local:8787/rh-
osbs/rhceph@sha256:3075e8708792ebd527ca14849b6af4a11256a3f881ab09b837d7af0f8b2
102ea
Scheduled crash update...
Scheduled mgr update...
Scheduled mon update...
Scheduled osd.default_drive_group update...
Scheduled rgw.rgw update...
```

3. 新しいモニターの可視性を確認します。

```
$ sudo cephadm --ceph status
```

.

11.9. CONTROLLER ノード置き換え後のクリーンアップ

ノードの交換が完了したら、コントローラークラスターを完成させることができます。

手順

1. Controller 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。

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

3. フェンシングの有効化:

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo pcs property set stonith-enabled=true
```

4. 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。

```
[tripleo-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。

5. director を終了します。

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

6. 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
```

7. 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
```

8. 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
```

9. 必要に応じて、新規ノード上の L3 エージェントホストに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 **r1** という名称のルーターを追加しています。

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

10. cinder サービスを掃除します。

    a. cinder サービスをリスト表示します。

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

    b. Controller ノードにログインし、**cinder-api** コンテナーに接続し、**cinder-manage service remove** コマンドを使用して、残っているサービスを削除します。

    ```
    [tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it cinder_api cinder-manage service remove cinder-backup <host>
    [tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it cinder_api cinder-manage service remove cinder-scheduler <host>
    ```

11. RabbitMQ クラスターをクリーンアップします。

    a. Controller ノードにログインします。

b. **podman exec** コマンドを使用して bash を起動し、RabbitMQ クラスターのステータスを確認します。

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo podman exec -it rabbitmq-bundle-podman-0 bash
[root@overcloud-controller-0 /]$ rabbitmqctl cluster_status
```

c. 置き換えられた Controller ノードをクリアするには、**rabbitmqctl** コマンドを使用します。

```
[root@controller-0 /]$ rabbitmqctl forget_cluster_node <node_name>
```

12. ブートストラップ Controller ノードを置き換えた場合は、置き換えプロセス後に環境ファイル **~/templates/bootstrap-controller.yaml** を削除するか、既存の環境ファイルから **pacemaker_short_bootstrap_node_name** および **mysql_short_bootstrap_node_name** パラメーターを削除する必要があります。このステップにより、これ以降の置き換えで director が Controller ノード名をオーバーライドしようとするのを防ぎます。詳細は、ブートストラップコントローラーノードの交換 を参照してください。

13. オーバークラウドで Object Storage サービス (swift) を使用している場合は、オーバークラウドノードを更新した後、swift リングを同期する必要があります。次の例のようなスクリプトを使用して、リングファイルを既存のコントローラーノード (この例ではコントローラーノード 0) からすべてのコントローラーノードに配布し、それらのノードで Object Storage サービスコンテナーを再起動します。

```
#!/bin/sh
set -xe

SRC="tripleo-admin@overcloud-controller-0.ctlplane"
ALL="tripleo-admin@overcloud-controller-0.ctlplane tripleo-admin@overcloud-controller-1.ctlplane tripleo-admin@overcloud-controller-2.ctlplane"
```

- リングファイルの現在のセットを取得します。

```
ssh "${SRC}" 'sudo tar -czvf - /var/lib/config-data/puppet-generated/swift_ringbuilder/etc/swift/{*.builder,*.ring.gz,backups/*.builder}' > swift-rings.tar.gz
```

- リングをすべてのノードにアップロードし、それらを正しい場所に配置して、swift サービスを再起動します。

```
for DST in ${ALL}; do
  cat swift-rings.tar.gz | ssh "${DST}" 'sudo tar -C / -xvzf -'
  ssh "${DST}" 'sudo podman restart swift_copy_rings'
  ssh "${DST}" 'sudo systemctl restart tripleo_swift*'
done
```

第12章 ノードの再起動

アンダークラウドおよびオーバークラウドで、ノードをリブートしなければならない場合があります。

注記

オーバークラウドでインスタンス HA (高可用性) を有効にし、コンピュートノードをシャットダウンまたはリブートする必要がある場合は、第 3 章 インスタンス HA を使用したアンダークラウドとオーバークラウドのメンテナンスを実行する を **インスタンスの高可用性の設定** で参照してください。

以下の手順を使用して、さまざまなノード種別をリブートする方法を説明します。

- 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。ロールの全ノードを同時に再起動すると、その操作中サービスにダウンタイムが生じる場合があります。
- OpenStack Platform 環境の全ノードをリブートする場合には、以下の順序でノードをリブートします。

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. Controller 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. Compute ノードのリブート

12.1. アンダークラウドノードのリブート

アンダークラウドノードをリブートするには、以下の手順を実施します。

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。

12.2. コントローラーノードおよびコンポーザブルノードの再起動

設定可能なロールに基づいて Controller ノードとスタンドアロンノードを再起動し、Compute ノードと Ceph ストレージノードを除外します。

手順

1. 再起動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。

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

3. ノードをリブートします。

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

4. ノードがブートするまで待ちます。

検証

1. サービスが有効になっていることを確認します。
   a. ノードが Pacemaker サービスを使用している場合は、ノードがクラスターに再度加わったか確認します。

```
[tripleo-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。

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

   c. ノードがコンテナー化されたサービスを使用している場合には、ノード上の全コンテナーがアクティブであることを確認します。

```
[tripleo-admin@overcloud-controller-0 ~]$ sudo podman ps
```

12.3. スタンドアロンの CEPH MON ノードのリブート

スタンドアロンの Ceph MON ノードをリブートするには、以下の手順を実施します。

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。

12.4. CEPH STORAGE (OSD) クラスターのリブート

Ceph Storage (OSD) ノードのクラスターを再起動するには、以下の手順を実施します。

前提条件

- **ceph-mon** サービスを実行している Ceph Monitor または Controller ノードで、Red Hat Ceph Storage クラスターのステータスが正常であり、pg ステータスが **active+clean** であることを確認する。

```
$ sudo cephadm -- shell ceph status
```

Ceph クラスターが正常な場合、**HEALTH_OK** のステータスが返されます。

Ceph クラスターのステータスが異常な場合、**HEALTH_WARN** または **HEALTH_ERR** のステータスが返されます。トラブルシューティングのガイダンスは、Red Hat Ceph Storage 5 トラブルシューティングガイド または Red Hat Ceph Storage 6 トラブルシューティングガイド を参照してください。

手順

1. **ceph-mon** サービスを実行している Ceph Monitor または Controller ノードにログインし、Ceph Storage クラスターのリバランスを一時的に無効にします。

```
$ sudo cephadm shell -- ceph osd set noout
$ sudo cephadm shell -- ceph osd set norebalance
```

注記

マルチスタックまたは分散コンピュートノード (DCN) アーキテクチャーを使用している場合は、**noout** フラグと **norebalance** フラグの設定時に Ceph クラスター名を指定する必要があります。例: **sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring**。

2. 再起動する最初の Ceph Storage ノードを選択し、そのノードにログインします。
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. ノードにログインし、Ceph クラスターのステータスを確認します。

```
$ sudo cephadm -- shell ceph status
```

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. ノードからログアウトして、次のノードを再起動し、ステータスを確認します。全 Ceph Storage ノードが再起動されるまで、このプロセスを繰り返します。
7. 完了したら、**ceph-mon** サービスを実行している Ceph Monitor または Controller ノードにログインし、クラスターのリバランスを有効にします。

```
$ sudo cephadm shell -- ceph osd unset noout
$ sudo cephadm shell -- ceph osd unset norebalance
```

注記

マルチスタックまたは分散コンピュートノード (DCN) アーキテクチャーを使用している場合は、**noout** フラグと **norebalance** フラグの設定解除時に Ceph クラスター名を指定する必要があります。例: **sudo cephadm shell -c /etc/ceph/<cluster>.conf -k /etc/ceph/<cluster>.client.keyring**。

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 cephadm shell ceph status
```

12.5. 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 Object Storage ノードに対して、次の手順を実行します。

手順

1. Object Storage 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 Object Storage ノードごとに再起動を繰り返します。

12.6. COMPUTE ノードの再起動

Red Hat OpenStack Platform (RHOSP) 環境でのインスタンスのダウンタイムを最小限に抑えるために、インスタンスの移行ワークフローでは、再起動するコンピュートノードからインスタンスを移行する時に必要な手順を概説します。

インスタンスの移行ワークフロー

1. Compute ノードを再起動する前に、インスタンスを別のノードに移行するか決定します。
2. 再起動する Compute ノードを選択して無効にし、新規インスタンスをプロビジョニングしないようにします。
3. インスタンスを別の Compute ノードに移行します。
4. 空の Compute ノードを再起動します。
5. 空の Compute ノードを有効にします。

前提条件

- Compute ノードを再起動する前に、ノードの再起動中にインスタンスを別の Compute ノードに移行するか決定する。

Compute ノード間で仮想マシンインスタンスを移行する際に発生する可能性のある移行の制約リストを確認する。詳細は、**インスタンス作成のための Compute サービスの設定** の 移行の制約 を参照してください。

注記

Multi-RHEL 環境があり、RHEL 9.2 を実行しているコンピュートノードから RHEL 8.4 を実行しているコンピュートノードに仮想マシンを移行する場合は、コールドマイグレーションのみがサポートされます。コールドマイグレーションの詳細は、**インスタンス作成のための Compute サービスの設定** の インスタンスのコールドマイグレーション を参照してください。

- インスタンスを移行できない場合は、以下のコアテンプレートパラメーターを設定して、Compute ノード再起動後のインスタンスの状態を制御する。

NovaResumeGuestsStateOnHostBoot

リブート後の Compute ノードで、インスタンスを同じ状態に戻すか定義します。**False** に設定すると、インスタンスは停止した状態を維持し、手動で起動する必要があります。デフォルト値は **False** です。

NovaResumeGuestsShutdownTimeout

再起動する前に、インスタンスのシャットダウンを待つ秒数。この値を **0** に設定することは推奨されません。デフォルト値は **300** です。
オーバークラウドパラメーターおよびその使用方法の詳細は、オーバークラウドのパラメーター を参照してください。

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. コンピュートノードのリストを取得して、再起動するノードのホスト名を特定します。

```
(undercloud)$ source ~/overcloudrc
(overcloud)$ openstack compute service list
```

再起動するコンピュートノードのホスト名を特定します。

3. 再起動するコンピュートノード上のコンピュートサービスを無効にします。

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

- **<hostname>** は、コンピュートノードのホスト名に置き換えます。

4. Compute ノード上の全インスタンスをリスト表示します。

```
(overcloud)$ openstack server list --host <hostname> --all-projects
```

5. オプション: インスタンスを別のコンピュートノードに移行するには、次の手順を実行します。
   a. インスタンスを別の Compute ノードに移行する場合は、以下のコマンドのいずれかを使用します。
      - インスタンスを別のホストに移行するには、次のコマンドを実行します。

```
(overcloud) $ openstack server migrate <instance_id> --live <target_host> --wait
```

- **<instance_id>** は、インスタンス ID に置き換えます。
- **<target_host>** は、インスタンスの移行先のホストに置き換えます。

- **nova-scheduler** がターゲットホストを自動的に選択できるようにします。

```
(overcloud) $ nova live-migration <instance_id>
```

- すべてのインスタンスを一度にライブマイグレーションします。

```
$ nova host-evacuate-live <hostname>
```

注記

nova コマンドで非推奨の警告が表示される可能性がありますが、無視しても問題ありません。

b. 移行が完了するまで待ちます。

c. 移行が正常に完了したことを確認します。

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

d. Compute ノードのインスタンスがなくなるまで、移行を続けます。

6. コンピュートノードにログインして、ノードをリブートします。

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

7. ノードがブートするまで待ちます。

8. Compute ノードを再度有効にします。

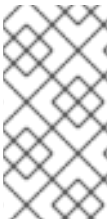
```
$ source ~/overcloudrc
(overcloud) $ openstack compute service set <hostname>  nova-compute --enable
```

9. Compute ノードが有効であることを確認します。

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

第13章 アンダークラウドおよびオーバークラウドのシャットダウンおよび起動

アンダークラウドおよびオーバークラウドでメンテナンスを実施する必要がある場合は、オーバークラウド起動時の問題を最小限に抑えるために、アンダークラウドおよびオーバークラウドノードを特定の順序でシャットダウンして起動する必要があります。



注記

オーバークラウドでインスタンス HA (高可用性) を有効にし、コンピュートノードをシャットダウンまたはリブートする必要がある場合は、第 3 章 インスタンス HA を使用したアンダークラウドとオーバークラウドのメンテナンスを実行する を **インスタンスの高可用性の設定** で参照してください。

前提条件

- 動作中のアンダークラウドおよびオーバークラウド

13.1. アンダークラウドおよびオーバークラウドのシャットダウン順序

Red Hat OpenStack Platform 環境をシャットダウンするには、オーバークラウドおよびアンダークラウドを以下の順序でシャットダウンする必要があります。

1. オーバークラウド Compute ノード上のインスタンスをシャットダウンします。
2. Compute ノードをシャットダウンします。
3. Controller ノードの高可用性サービスおよび OpenStack Platform のサービスをすべて停止します。
4. Ceph Storage ノードをシャットダウンします。
5. Controller ノードをシャットダウンします。
6. アンダークラウドをシャットダウンします。

13.2. オーバークラウド COMPUTE ノード上のインスタンスのシャットダウン

Red Hat OpenStack Platform 環境のシャットダウンのサブタスクとして、Compute ノードをシャットダウンする前に Compute ノード上のインスタンスをすべてシャットダウンします。

前提条件

- Compute サービスがアクティブなオーバークラウド

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. source コマンドでオーバークラウドの認証情報ファイルを読み込みます。

```
$ source ~/overcloudrc
```

3. オーバークラウドで実行中のインスタンスを表示します。

```
$ openstack server list --all-projects
```

4. オーバークラウドのそれぞれのインスタンスを停止します。

```
$ openstack server stop <INSTANCE>
```

オーバークラウド内のすべてのインスタンスを停止するまで、それぞれのインスタンスでこのステップを繰り返します。

13.3. COMPUTE ノードのシャットダウン

Red Hat OpenStack Platform 環境をシャットダウンする際のサブタスクとして、それぞれの Compute ノードにログインしてシャットダウンします。

前提条件

- Compute ノード上のすべてのインスタンスがシャットダウンされている。

手順

1. Compute ノードに **root** ユーザーとしてログインします。
2. ノードをシャットダウンします。

```
# shutdown -h now
```

3. すべての Compute ノードをシャットダウンするまで、それぞれの Compute ノードでこの手順を実施します。

13.4. CONTROLLER ノードのサービスの停止

Red Hat OpenStack Platform 環境のシャットダウンする際のサブタスクとして、Controller ノードをシャットダウンする前にノードのサービスを停止します。これには Pacemaker サービスおよび systemd サービスが含まれます。

前提条件

- Pacemaker サービスがアクティブなオーバークラウド

手順

1. Controller ノードに **root** ユーザーとしてログインします。
2. Pacemaker クラスターを停止します。

```
# pcs cluster stop --all
```

このコマンドにより、すべてのノード上のクラスターが停止します。

3. Pacemaker サービスが停止するまで待ち、サービスが停止したことを確認します。

a. Pacemaker のステータスを確認します。

```
# pcs status
```

b. Pacemaker サービスが実行されていないことを Podman で確認します。

```
# podman ps --filter "name=.*-bundle.*"
```

4. Red Hat OpenStack Platform のサービスを停止します。

```
# systemctl stop 'tripleo_*'
```

5. サービスが停止するまで待ち、サービスが実行されなくなったことを Podman で確認します。

```
# podman ps
```

13.5. CEPH STORAGE ノードのシャットダウン

Red Hat OpenStack Platform 環境のシャットダウンのサブタスクとして、Ceph Storage サービスを無効にし、続いてそれぞれの Ceph Storage ノードにログインしてシャットダウンします。

前提条件

- 正常な Ceph Storage クラスター。
- Ceph MON サービスがスタンドアロンの Ceph MON ノードまたは Controller ノードで動作している。

手順

1. Ceph MON サービスを実行するノード (例: Controller ノードまたはスタンドアロンの Ceph MON ノード) に **root** ユーザーとしてログインします。
2. クラスターが正常であることを確認します。以下の例の **podman** コマンドにより、Controller ノード上の Ceph MON コンテナー内でステータス確認が実施されます。

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

ステータスが **HEALTH_OK** であることを確認します。

3. クラスターの **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown**、および **pause** フラグを設定します。以下の例の **podman** コマンドにより、Controller ノード上の Ceph MON コンテナーを通じてこれらのフラグが設定されます。

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd set noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd set norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd set nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd set pause
```

4. それぞれの Ceph Storage ノードをシャットダウンします。

a. Ceph Storage ノードに **root** ユーザーとしてログインします。

b. ノードをシャットダウンします。

```
# shutdown -h now
```

c. すべての Ceph Storage ノードをシャットダウンするまで、それぞれの Ceph Storage ノードでこの手順を実施します。

5. スタンドアロンの Ceph MON ノードをすべてシャットダウンします。

a. スタンドアロンの Ceph MON ノードに **root** ユーザーとしてログインします。

b. ノードをシャットダウンします。

```
# shutdown -h now
```

c. スタンドアロンの Ceph MON ノードをすべてシャットダウンするまで、それぞれのスタンドアロンの Ceph MON ノードでこの手順を実施します。

関連情報

- What is the procedure to shutdown and bring up the entire ceph cluster?

13.6. CONTROLLER ノードのシャットダウン

Red Hat OpenStack Platform 環境のシャットダウンのサブタスクとして、それぞれの Controller ノードにログインしてシャットダウンします。

前提条件

- Pacemaker クラスターが停止している。
- Controller ノードのすべての Red Hat OpenStack Platform サービスが停止している。

手順

1. Controller ノードに **root** ユーザーとしてログインします。
2. ノードをシャットダウンします。

```
# shutdown -h now
```

3. すべての Controller ノードをシャットダウンするまで、それぞれの Controller ノードでこの手順を実施します。

13.7. アンダークラウドのシャットダウン

Red Hat OpenStack Platform 環境のシャットダウンのサブタスクとして、アンダークラウドノードにログインしてアンダーグラウンドをシャットダウンします。

前提条件

- 動作中のアンダークラウド

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シャットダウンします。

```
$ sudo shutdown -h now
```

13.8. システムメンテナンスの実施

アンダークラウドおよびオーバークラウドを完全にシャットダウンしたら、環境内のシステムに対するメンテナンスを実施し、続いてアンダークラウドおよびオーバークラウドを起動します。

13.9. アンダークラウドおよびオーバークラウドの起動順序

Red Hat OpenStack Platform 環境を起動するには、アンダークラウドおよびオーバークラウドを以下の順序で起動する必要があります。

1. アンダークラウドを起動します。
2. Controller ノードを起動します。
3. Ceph Storage ノードを起動します。
4. Compute ノードを起動します。
5. オーバークラウドの Compute ノードでインスタンスを起動します。

13.10. アンダークラウドの起動

Red Hat OpenStack Platform 環境の起動のサブタスクとして、アンダークラウドノードの電源をオンにし、アンダークラウドにログインし、アンダークラウドのサービスを確認します。

前提条件

- アンダークラウドの電源がオフになっている。

手順

- アンダークラウドの電源をオンにし、アンダークラウドがブートするまで待ちます。

検証

1. アンダークラウドホストに **stack** ユーザーとしてログインします。
2. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

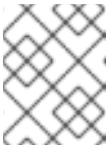
3. アンダークラウドのサービスを確認します。

```
$ systemctl list-units 'tripleo_*'
```

4. **tripleo-ansible-inventory.yaml** という名前の静的インベントリーファイルを検証します。

```
$ validation run --group pre-introspection -i <inventory_file>
```

- **<inventory_file>** ファイルを Ansible インベントリーファイルの名前および場所に置き換えます (例: ~/**tripleo-deploy/undercloud/tripleo-ansible-inventory.yaml**)。



注記

検証を実行すると、出力の **Reasons** 列は 79 文字に制限されます。検証結果を完全に表示するには、検証ログファイルを表示します。

5. すべてのサービスとコンテナーがアクティブおよび正常であることを確認します。

```
$ validation run --validation service-status --limit undercloud -i <inventory_file>
```

関連情報

- 検証フレームワークの使用

13.11. CONTROLLER ノードの起動

Red Hat OpenStack Platform 環境の起動のサブタスクとして、それぞれの Controller ノードの電源をオンにし、そのノードの Pacemaker 以外のサービスを確認します。

前提条件

- Controller ノードの電源がオフになっている。

手順

- それぞれの Controller ノードの電源をオンにします。

検証

1. **root** ユーザーとして各 Controller ノードにログインします。
2. Controller ノードのサービスを確認します。

```
$ systemctl -t service
```

Pacemaker ベース以外のサービスだけが動作中です。

3. Pacemaker サービスが起動するまで待ち、サービスが起動したことを確認します。

```
$ pcs status
```

注記

環境でインスタンス HA を使用している場合、Pacemaker リソースは、Compute ノードを起動するか、**pcs stonith confirm <compute_node>** コマンドを使用して手動でフェンスを解除する操作を実行するまで起動しません。このコマンドは、インスタンス HA を使用する各 Compute ノードで実行する必要があります。

13.12. CEPH STORAGE ノードの起動

Red Hat OpenStack Platform 環境の起動のサブタスクとして、Ceph MON ノードおよび Ceph Storage ノードの電源をオンにし、Ceph Storage サービスを有効にします。

前提条件

- 電源がオフの Ceph Storage クラスター。
- 電源がオフのスタンドアロンの Ceph MON ノードまたは電源がオンの Controller ノードで、Ceph MON サービスが有効になっている。

手順

1. 使用している環境にスタンドアロンの Ceph MON ノードがある場合、それぞれの Ceph MON ノードの電源をオンにします。
2. それぞれの Ceph Storage ノードの電源をオンにします。
3. Ceph MON サービスを実行するノード (例: Controller ノードまたはスタンドアロンの Ceph MON ノード) に **root** ユーザーとしてログインします。
4. クラスターノードのステータスを確認します。以下の例の **podman** コマンドにより、Controller ノード上の Ceph MON コンテナー内でステータス確認が実施されます。

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

それぞれのノードの電源がオンで、接続された状態であることを確認します。

5. クラスターの **noout**、**norecover**、**norebalance**、**nobackfill**、**nodown**、および **pause** フラグの設定を解除します。以下の例の **podman** コマンドにより、Controller ノード上の Ceph MON コンテナーを通じてこれらのフラグの設定が解除されます。

```
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset noout
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norecover
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset norebalance
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nobackfill
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset nodown
# sudo podman exec -it ceph-mon-controller-0 ceph osd unset pause
```

検証

1. クラスターが正常であることを確認します。以下の例の **podman** コマンドにより、Controller ノード上の Ceph MON コンテナー内でステータス確認が実施されます。

```
# sudo podman exec -it ceph-mon-controller-0 ceph status
```

ステータスが **HEALTH_OK** であることを確認します。

関連情報

- What is the procedure to shutdown and bring up the entire ceph cluster?

13.13. COMPUTE ノードの起動

Red Hat OpenStack Platform 環境の起動のサブタスクとして、それぞれの Compute ノードの電源をオンにし、そのノードのサービスを確認します。

前提条件

- 電源がオフの Compute ノード

手順

1. それぞれの Compute ノードの電源をオンにします。

検証

1. 各 Compute に **root** ユーザーとしてログインします。
2. Compute ノードのサービスを確認します。

```
$ systemctl -t service
```

13.14. オーバークラウド COMPUTE ノード上のインスタンスの起動

Red Hat OpenStack Platform 環境の起動のサブタスクとして、Compute ノード上のインスタンスを起動します。

前提条件

- アクティブなノードを持つアクティブなオーバークラウド

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. source コマンドでオーバークラウドの認証情報ファイルを読み込みます。

```
$ source ~/overcloudrc
```

3. オーバークラウドで実行中のインスタンスを表示します。

```
$ openstack server list --all-projects
```

4. オーバークラウド内のインスタンスを起動します。

```
$ openstack server start <INSTANCE>
```

第14章 DIRECTOR のエラーに関する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。このセクションでは、典型的な問題の診断を説明します。

14.1. ノードの登録に関する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、ノードの情報が含まれるテンプレートファイルを検証して、インポートされたノードの情報を修正します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. **--validate-only** オプションを指定して、ノードのインポートコマンドを実行します。このオプションを指定した場合には、インポートを実施せずにノードのテンプレートを検証します。

```
(undercloud) $ openstack overcloud node import --validate-only ~/nodes.json
Waiting for messages on queue 'tripleo' with no timeout.

Successfully validated environment file
```

3. インポートしたノードの誤った情報を修正するには、**openstack baremetal** コマンドを実行してノードの情報を更新します。ネットワーク設定の詳細を変更する方法を、以下の例に示します。

   a. インポートしたノードに割り当てられたポートの UUID を特定します。

```
$ source ~/stackrc
(undercloud) $ openstack baremetal port list --node [NODE UUID]
```

   b. MAC アドレスを更新します。

```
(undercloud) $ openstack baremetal port set --address=[NEW MAC] [PORT UUID]
```

   c. ノードの新しい IPMI アドレスを設定します。

```
(undercloud) $ openstack baremetal node set --driver-info ipmi_address=[NEW IPMI ADDRESS] [NODE UUID]
```

14.2. ハードウェアのイントロスペクションに関するトラブルシューティング

Bare Metal Provisioning インスペクタサービスの **ironic-inspector** は、インスペクション RAM ディスクが応答しない場合、デフォルトの 1 時間後にタイムアウトします。タイムアウトは検査 RAM ディスクのバグを示している可能性がありますが、通常は環境の設定ミスが原因でタイムアウトが発生します。

一般的な環境設定ミスの問題を診断して解決し、イントロスペクションプロセスが最後まで確実に実行されるようにすることができます。

手順

1. **stackrc** アンダークラウド認証情報ファイルを入手します。

```
$ source ~/stackrc
```

2. ノードを **manageable** の状態にします。イントロスペクションは、デプロイメント用を意味する **available** の状態にあるノードを検査しません。**available** の状態にあるノードを検査するには、イントロスペクションの前にノードのステータスを **manageable** の状態に変更します。

```
(undercloud)$ openstack baremetal node manage <node_uuid>
```

3. イントロスペクションのデバッグ中にイントロスペクション RAM ディスクへの一時的なアクセスを設定するには、**sshkey** パラメーターを使用して、公開 SSH キーを **/httpboot/inspector.ipxe** ファイルの **kernel** 設定に追加します。

```
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 selinux=0 sshkey="<public_ssh_key>"
```

4. ノード上でイントロスペクションを実行します。

```
(undercloud)$ openstack overcloud node introspect <node_uuid> --provide
```

イントロスペクションの完了後にノードの状態を **available** に変更するには、**--provide** オプションを使用します。

5. **dnsmasq** ログでノードの IP アドレスを特定します。

```
(undercloud)$ sudo tail -f /var/log/containers/ironic-inspector/dnsmasq.log
```

6. エラーが発生する場合には、root ユーザーおよび一時アクセス用の詳細を使用してノードにアクセスします。

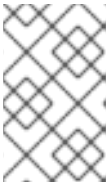
```
$ ssh root@192.168.24.105
```

イントロスペクション中にノードにアクセスして、診断コマンドを実行してイントロスペクションの失敗に関するトラブルシューティングを行います。

7. イントロスペクションのプロセスを停止するには、以下のコマンドを実行します。

```
(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。



注記

イントロスペクションの最初の中止後、Red Hat OpenStack Platform director は実行を 3 回試みます。イントロスペクションを完全に中止するには、それぞれの試行時に **openstack baremetal introspection abort** コマンドを実行します。

14.3. オーバークラウドの作成およびデプロイメントに関するトラブルシューティング

オーバークラウドの初回作成は、OpenStack Orchestration (heat) サービスにより実行されます。オーバークラウドのデプロイメントに失敗した場合には、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。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. エフェメラル Heat プロセスを開始します。

```
(undercloud)$ openstack tripleo launch heat --heat-dir /home/stack/overcloud-deploy/overcloud/heat-launcher --restore-db
(undercloud)$ export OS_CLOUD=heat
```

3. 失敗の詳細を表示します。

```
(undercloud)$ openstack stack failures list <overcloud> --long
```

- **<overcloud>** 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。

4. 失敗したスタックを特定します。

```
(undercloud)$ openstack stack list --nested --property status=FAILED
```

5. エフェメラル Heat プロセスをアンダークラウドから削除します。

```
(undercloud)$ openstack tripleo launch heat --kill
```

14.4. ノードのプロビジョニングに関するトラブルシューティング

OpenStack Orchestration (heat) サービスは、プロビジョニングプロセスを制御します。ノードのプロビジョニングに失敗した場合には、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、問題の診断を行います。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. Bare Metal サービスをチェックして、全登録ノードおよびそれらの現在の状態を表示します。

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

プロビジョニングに使用できるすべてのノードは、以下の状態でなければなりません。

- Maintenance: **False** に設定。
- Provision State: プロビジョニングの前に **available** に設定。

3. ノードの **Maintenance** が **False** に設定されていない場合、または **Provision State** が **available** に設定されていない場合は、次の表を使用して問題と解決策を特定します。

問題	原因	ソリューション
Maintenance が自動的に **True** に設定される。	director がノードの電源管理にアクセスできない。	ノードの電源管理の認証情報を確認します。
Provision State は **available** に設定されているが、ノードがプロビジョニングされない。	ベアメタルのデプロイが開始する前に問題が発生した。	ノードのハードウェア詳細が要件を満たしていることを確認します。
Provision State がノードの **wait call-back** に設定される。	このノードのプロビジョニングプロセスがまだ終了していない。	このステータスが変わるまで待ちます。あるいは、ノードの仮想コンソールに接続し、出力を確認します。
Provision State および Power State はそれぞれ **active** および **power on** だが、ノードが応答しない。	ノードのプロビジョニングは正常に終了したが、デプロイメント後の設定ステップで問題が生じている。	ノード設定のプロセスを診断します。ノードの仮想コンソールに接続し、出力を確認します。
Provision State が **error** または **deploy failed** である。	ノードのプロビジョニングに失敗している。	**openstack baremetal node show** コマンドを使用してベアメタルノードの詳細を表示し、エラーに関する説明が含まれる **last_error** フィールドを確認します。

関連情報

- ベアメタルノードのプロビジョニング状態

14.5. プロビジョニング時の IP アドレス競合に関するトラブルシューティング

対象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、イントロスペクションおよびデプロイメントのタスクは失敗します。このタスク失敗を防ぐには、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します。

手順

1. **nmap** をインストールします。

```
$ sudo dnf install nmap
```

2. **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
```

3. **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。

```
$ 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
```

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。

14.6. オーバークラウドの設定に関するトラブルシューティング

Red Hat OpenStack Platform director は、Ansible を使用してオーバークラウドを設定します。オーバークラウドでの Ansible Playbook のエラー (**config-download**) を診断するには、以下の手順を実施します。

手順

1. **stack** ユーザーが **undercloud** 上の ~/**config-download**/**overcloud** ディレクトリー内のファイルにアクセスできるようにします。

```
$ sudo setfacl -R -m u:stack:rwx ~/config-download/overcloud
```

2. **config-download** ファイルの作業ディレクトリーに移動します。

```
$ cd ~/config-download/overcloud
```

3. **ansible.log** ファイルを確認し、異常のあった場所を探します。

```
$ less ansible.log
```

異常のあったステップを書き留めます。

4. 作業ディレクトリー内の **config-download** Playbook で異常のあったステップを探し、実行していたアクションを特定します。

14.7. コンテナーの設定に関するトラブルシューティング

Red Hat OpenStack Platform director は、**podman** を使用してコンテナーを管理し、**puppet** を使用してコンテナー設定を作成します。以下の手順で、エラーが発生した場合にコンテナーを診断する方法を説明します。

ホストへのアクセス

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 障害の発生したコンテナーがあるノードの IP アドレスを取得します。

```
(undercloud) $ metalsmith list
```

3. ノードにログインします。

```
(undercloud) $ ssh tripleo-admin@192.168.24.60
```

障害が発生したコンテナーの識別

1. すべてのコンテナーを表示します。

```
$ sudo podman ps --all
```

障害の発生したコンテナーを特定します。障害の発生したコンテナーは、通常ゼロ以外のステータスで終了します。

コンテナーログの確認

1. 各コンテナーは、主要プロセスからの標準出力を保持します。この出力をログとして使用し、コンテナー実行時に実際に何が発生したのかを特定するのに役立てます。たとえば、**keystone** コンテナーのログを確認するには、以下のコマンドを実行します。

```
$ sudo podman logs keystone
```

多くの場合、このログにコンテナー障害の原因に関する情報が含まれます。

2. ホストには、失敗したサービスの **stdout** ログも保持されます。**stdout** ログは、**/var/log/containers/stdouts/** に保存されます。たとえば、障害の発生した **keystone** コンテナーのログを確認するには、以下のコマンドを使用します。

```
$ cat /var/log/containers/stdouts/keystone.log
```

コンテナーの検査

状況によっては、コンテナー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** コンテナーのデータを確認します。

```
$ sudo podman inspect keystone
```

このコマンドにより、ローレベルの設定データが含まれた JSON オブジェクトが返されます。その出力を **jq** コマンドにパイプして、特定のデータを解析することが可能です。たとえば、**keystone** コンテナー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 podman 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、コンテナー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** コンテナー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 podman inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range .Mounts}} -v
{{.Source}}:{{.Destination}}:{{ join .Options "," }}{{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 **podman 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ーを再度作成します。

```
$ OPTIONS=$( sudo podman inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range
.Mounts}} -v {{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}'
keystone )
$ sudo podman run --rm $OPTIONS /bin/bash
```

コンテナー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ー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 **podman** コマンドを使用して、稼働中のコンテナー内でコマンドを実行します。たとえば、**podman exec** コマンドを実行して、**keystone** コンテナー内でコマンドを実行します。

```
$ sudo podman 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>** を実行するコマンドに置き換えてください。たとえば、各コンテナーには、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 podman exec -ti keystone /openstack/healthcheck
```

コンテナーのシェルにアクセスするには、コンテナー内で実行するコマンドとして **/bin/bash** を使用して **podman exec** を実行します。

```
$ sudo podman exec -ti keystone /bin/bash
```

コンテナーファイルシステムの確認

1. 障害の発生したコンテナーのファイルシステムを確認するには、**podman mount** コマンドを実行します。たとえば、障害の発生した **keystone** コンテナーのファイルシステムを確認するには、以下のコマンドを実行します。

   ```
   $ sudo podman mount keystone
   ```

   これによりマウント位置が表示され、ファイルシステムの内容を確認することができます。

   ```
   /var/lib/containers/storage/overlay/78946a109085aeb8b3a350fc20bd8049a08918d74f573396d7358270e711c610/merged
   ```

   これは、コンテナー内の Puppet レポートを確認する際に役立ちます。これらのレポートは、コンテナーのマウント内の **var/lib/puppet/** ディレクトリーにあります。

コンテナーのエクスポート

コンテナー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、コンテナー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** コンテナー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 podman export keystone -o keystone.tar
```

このコマンドにより **keystone.tar** アーカイブが作成されます。これを抽出して、調べることができます。

14.8. COMPUTE ノードの異常に関するトラブルシューティング

Compute ノードは、Compute サービスを使用して、ハイパーバイザーベースの操作を実行します。これは、このサービスを中心に Compute 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。

手順

1. **stackrc** ファイルを取得します。

   ```
   $ source ~/stackrc
   ```

2. 障害が発生した Compute ノードの IP アドレスを取得します。

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

3. ノードにログインします。

```
(undercloud) $ ssh tripleo-admin@192.168.24.60
```

4. root ユーザーに変更します。

```
$ sudo -i
```

5. コンテナーのステータスを表示します。

```
$ sudo podman ps -f name=nova_compute
```

6. Compute ノードの主なログファイルは **/var/log/containers/nova/nova-compute.log** です。Compute ノードの通信で問題が発生した場合、このファイルを使用して診断を始めます。

7. Compute 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 Compute ノードに移行し、ノードを無効にします。

14.9. SOS レポートの作成

Red Hat に連絡して Red Hat 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** を生成しなければならない場合があります。**SOS レポート** 作成の詳細は、以下のドキュメントを参照してください。

- How to collect all required logs for Red Hat Support to investigate an OpenStack issue

14.10. ログの場所

問題のトラブルシューティングを行う場合は、以下のログを使用してアンダークラウドおよびオーバークラウドに関する情報を収集します。

表14.1 アンダークラウドおよびオーバークラウドノード両方に関するログ

情報	ログの場所
コンテナー化されたサービスに関するログ	**/var/log/containers/**
コンテナー化されたサービスからの標準出力	**/var/log/containers/stdouts**
Ansible の設定に関するログ	**~/ansible.log**

表14.2 アンダークラウドノードに関する追加のログ

情報	ログの場所
openstack overcloud deploy のコマンド履歴	**/home/stack/.tripleo/history**
アンダークラウドのインストールに関するログ	**/home/stack/install-undercloud.log**

表14.3 オーバークラウドノードに関する追加のログ

情報	ログの場所
cloud-init に関するログ	**/var/log/cloud-init.log**
高可用性に関するログ	**/var/log/pacemaker.log**

第15章 アンダークラウドおよびオーバークラウドサービスに関するヒント

このセクションでは、アンダークラウド上の特定の OpenStack サービスのチューニングと管理についてアドバイスを行います。

15.1. デプロイメントパフォーマンスのチューニング

Red Hat OpenStack Platform director は、OpenStack Orchestration (heat) を使用してメインのデプロイメントおよびプロビジョニング機能を実施します。Heat は一連のワーカーを使用してデプロイメントタスクを実行します。デフォルトのワーカー数を計算するには、director の heat 設定ではアンダークラウドの合計 CPU スレッド数を 2 で割ります。ここでは、スレッド数とは CPU コア数にハイパースレッディングの値を掛けたものを指します。たとえば、アンダークラウドの CPU スレッド数が 16 の場合には、デフォルトでは heat により 8 つのワーカーが提供されます。デフォルトでは、director の設定に最小および最大のワーカー数も適用されます。

サービス	最小値	最大値
OpenStack Orchestration (heat)	4	24

ただし、環境ファイルの **HeatWorkers** パラメーターを使用して、手動でワーカー数を設定することができます。

heat-workers.yaml

```
parameter_defaults:
  HeatWorkers: 16
```

undercloud.conf

```
custom_env_files: heat-workers.yaml
```

15.2. HAPROXY の SSL/TLS 暗号ルールの変更

アンダークラウドで SSL/TLS を有効にした場合は (「アンダークラウド設定パラメーター」 を参照)、HAProxy 設定で使用される SSL/TLS 暗号とルールを強化することが推奨されます。この強化は、POODLE TLS 脆弱性 などの SSL/TLS の脆弱性を回避するのに役立ちます。

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-sslv3 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-sslv3 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。

openstack undercloud install を実行する前に作成した hieradata オーバーライドファイルを使用するように、**undercloud.conf** ファイルの **hieradata_override** パラメーターを設定します。

```
[DEFAULT]
...
hieradata_override = haproxy-hiera-overrides.yaml
...
```

第16章 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。この付録には、director がサポートする電源管理機能のリストが記載されています。オーバークラウドのノードを登録する際に、これらの電源管理設定を使用します。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。

16.1. INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI)

Baseboard Management Controller (BMC) を使用する際の標準的な電源管理手法

pm_type
: このオプションを **ipmi** に設定します。

pm_user、pm_password
: IPMI のユーザー名およびパスワード

pm_addr
: IPMI Controller の IP アドレス

pm_port (オプション)
: IPMI Controller に接続するためのポート

16.2. REDFISH

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

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 Controller 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 Controller (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ーが低下するので、注意してください。

16.3. DELL REMOTE ACCESS CONTROLLER (DRAC)

DRAC は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェイスです。

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

16.4. INTEGRATED LIGHTS-OUT (ILO)

Hewlett-Packard の iLO は、電源管理やサーバー監視などの帯域外 (OOB) リモート管理機能を提供するインターフェイスです。

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にするには、**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。
- 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。

16.5. FUJITSU INTEGRATED REMOTE MANAGEMENT CONTROLLER (IRMC)

Fujitsu 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

重要

iRMC ユーザーには **ADMINISTRATOR** 権限が必要です。

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。

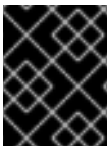
pm_sensor_method (オプション)

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にするには、**undercloud.conf** の **enabled_hardware_types** オプションに **irmc** を追加してから、**openstack undercloud install** コマンドを再実行します。

16.6. 手動管理ドライバー

電源管理を持たないベアメタルデバイスを制御するには、**manual-management** ドライバーを使用します。director は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およびデプロイメントの特定の時点で、手動で電源管理操作を実施する必要があります。



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 **manual-management** 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_hardware_types** オプションに **manual-management** を追加してから、**openstack undercloud install** コマンドを再実行します。
- **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 **manual-management** に設定します。

イントロスペクション

- 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手動でノードを起動します。ノードが PXE を介して起動することを確認します。
- ノードクリーニングを有効にしている場合は、**openstack baremetal node list** コマンドを実行した際に **Introspection completed** メッセージが表示され、各ノードの状態が **clean wait** になった後、手動でノードを再起動するようにしてください。ノードが PXE を介して起動することを確認します。
- イントロスペクションとクリーニングのプロセスが完了したら、ノードをシャットダウンします。

Deployment

- オーバークラウドデプロイメントを実行する場合は、**openstack baremetal node list** コマンド

を使用してノードのステータスを確認してください。ノードのステータスが **deploying** から **wait call-back** に変わるまで待ってから、ノードを手動で開始します。ノードが PXE を介して起動することを確認します。

- オーバークラウドプロビジョニングプロセスが完了すると、ノードはシャットダウンします。設定プロセスを開始するには、ディスクからノードを起動する必要があります。プロビジョニングの完了を確認するには、**openstack baremetal node list** コマンドを使用してノードのステータスを確認し、ノードのステータスが各ノードで **active** に変わるまで待ちます。ノードステータスが **active** な場合、プロビジョニングされたオーバークラウドノードを手動で起動します。