Red Hat OpenStack Platform 16.1

13 から 16.1 へのアップグレードフレームワーク

Red Hat OpenStack Platform 13 から 16.1 へのインプレースアップグレード

Last Updated: 2020-12-03

Red Hat OpenStack Platform 16.1 13 から 16.1 へのアップグレードフレームワーク

Red Hat OpenStack Platform 13 から 16.1 へのインプレースアップグレード

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドでは、ロングライフバージョン間のインプレースアップグレードのフレームワークについて説明します。このフレームワークには、OpenStack Platform 環境をあるロングライフバージョンから次のロングライフバージョンにアップグレードするためのツールが含まれます。今回、本書では Red Hat OpenStack Platform 13 (Queens) から 16.1 (Train) へのアップグレードに重点を置いています。

目次

第1章 RED HAT OPENSTACK PLATFORM のアップグレードフレームワークについて

Red Hat OpenStack Platform のアップグレードフレームワークは、Red Hat OpenStack Platform 環境をあるロングライフバージョンから次のロングライフバージョンにアップグレードするためのワークフローです。このワークフローはインプレースのソリューションで、アップグレードは既存の環境内で実行されます。

1.1. ロングライフバージョンのアップグレードフレームワーク

Red Hat OpenStack Platform の **アップグレードフレームワーク** を使用して、複数のオーバークラウドバージョンを経由するインプレースアップグレードパスを実施することができます。この機能は、**ロングライフバージョン** とされている特定の OpenStack のバージョンの使用を継続し、次のロングライフバージョンが提供された際にアップグレードする機会を提供することを目的としています。

本ガイドは、以下のバージョン間のアップグレードフレームワークを提供します。

現在のバージョン	目的のバージョン
Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 16.1

1.2. ロングライフバージョンのライフサイクルサポート

Red Hat OpenStack Platform のライフサイクルサポートに関する正確な日付けおよび詳細な情報は、「Red Hat OpenStack Platform のライフサイクル」 を参照してください。

1.3. ロングライフリリースのアップグレードパス

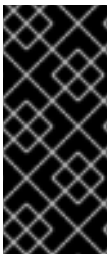
Red Hat では、お使いの環境を次のロングライフリリースにアップグレードするためのオプションを 2 つ提供しています。

インプレースアップグレード

既存の環境でサービスのアップグレードを実施します。本ガイドでは、主にこのオプションを中心に説明します。

並列移行

新しい Red Hat OpenStack Platform 16.1 環境を作成し、ワークロードを現在の環境から新しい環境に移行します。Red Hat OpenStack Platform の並列移行についての詳しい情報は、Red Hat Global Professional Services にお問い合わせください。



重要

以下の表に示す時間は内部テストに基づく最短の推定値であり、すべての実稼働環境には適用されない可能性があります。たとえば、ハードウェアのスペックが低い場合やブート時間が長い場合は、これらの時間に余裕を持たせてください。各タスクのアップグレード時間を正確に測定するには、実稼働環境と類似したハードウェアを持つテスト環境でこれらの手順を実施してください。

表1.1 アップグレードパスの影響と時間

	インプレースアップグレード	並列移行
アンダークラウドのアップグレード時間	それぞれの主要な操作の推定時間は以下のとおりです。 ● Leapp アップグレードコマンド: 30 分間 ● Leapp リブート: 30 分間 ● director のアップグレード: 40 分間	なし。既存のアンダークラウドに加えて、新しいアンダークラウドを作成します。
オーバークラウドコントロールプレーンのアップグレード時間	コントローラーノードごとの推定時間は以下のとおりです。 ● Leapp アップグレードおよびリブート: 60 分間 ● サービスのアップグレード: 60 分間	なし。既存のコントロールプレーンに加えて、新しいコントロールプレーンを作成します。
コントロールプレーンの機能停止時間	ブートストラップコントローラーノードのサービスアップグレード時間: 約 60 分間	なし。ワークロードの移行中、両方のオーバークラウドは稼動状態にあります。
コントロールプレーンの機能停止による影響	機能停止時間中 OpenStack の操作を行うことはできません。	機能停止時間はありません。
オーバークラウドデータプレーンのアップグレード時間	コンピュートノードおよび Ceph Storage ノードごとの推定時間は以下のとおりです。 ● Leapp アップグレードおよびリブート: 60 分間 ● サービスのアップグレード: 30 分間	なし。既存のデータプレーンに加えて、新しいデータプレーンを作成します。
データプレーンの機能停止時間	ノード間のワークロードの移行により、機能停止時間は最小限に抑えられます。	オーバークラウド間のワークロードの移行により、機能停止時間は最小限に抑えられます。
追加ハードウェアに関する要件	追加のハードウェアは必要ありません。	新しいアンダークラウドおよびオーバークラウドを作成するために、追加のハードウェアが必要です。

第2章 インプレースアップグレードの計画および準備

OpenStack Platform 環境のインプレースアップグレードを実施する前に、アップグレードのプランを作成し、正常なアップグレードを妨げる可能性のある障害に対処してください。

2.1. RED HAT OPENSTACK PLATFORM 16.1 の理解

アップグレードを実施する前に Red Hat OpenStack Platform 16.1 をよく理解しておくことで、結果として生じる環境や、アップグレードに影響を与える可能性のあるバージョン間の変更点を理解することができます。Red Hat OpenStack Platform 16.1 の理解を深めるには、以下の推奨事項に従ってください。

- アップグレードパスにわたるすべてのバージョンのリリースノートを確認し、計画が必要になる可能性のある要素を識別します。
    - 新しい機能が含まれるコンポーネント
    - 既知の問題

    以下のリンクから、各バージョンのリリースノートを確認してください。

    - Red Hat OpenStack Platform 13 (現在のバージョン)
    - Red Hat OpenStack Platform 14
    - Red Hat OpenStack Platform 15
    - Red Hat OpenStack Platform 16.0
    - Red Hat OpenStack Platform 16.1 (目的のバージョン)
- バージョン 16.1 の『director のインストールと使用方法』を参照し、新たな要件および本ガイドのプロセスについて十分に理解してください。
- 概念実証用の Red Hat OpenStack Platform 16.1 アンダークラウドおよびオーバークラウドをインストールします。対象のバージョンの OpenStack Platform を実際に操作して経験を積み、対象のバージョンと現在のバージョンの違いを調査します。

2.2. RED HAT OPENSTACK PLATFORM 16.1 における変更点の概要

Red Hat OpenStack Platform 16.1 へのアップグレード時に、以下に概要を示す変更が行われます。

- OpenStack Platform director 16.1 では、**config-download** と呼ばれる Ansible による手法を使用してオーバークラウドを設定します。これは、標準の heat ベースの設定手法に代わるものです。director は引き続き heat を使用してプロビジョニング操作のオーケストレーションを行います。
- director のインストールには、オーバークラウドのデプロイメントと同じ手法が使用されます。したがって、アンダークラウドでの各サービスのインストールおよび設定にも、ブループリントとして **openstack-tripleo-heat-templates** が使用されます。
- アンダークラウドでは、OpenStack サービスはコンテナー内で実行されます。
- アンダークラウドは、新たな方法でコンテナーイメージをプルして保管します。オーバークラウドのデプロイ前にコンテナーイメージをプルする代わりに、アンダークラウドはデプロイメントプロセス中に関連するすべてのコンテナーイメージをプルします。

- オーバークラウドのデプロイメントプロセスには、ノードを登録するための Advanced Subscription Management 手法が含まれます。この手法には、OpenStack Platform ノードを登録するための Ansible ロールが組み込まれています。さらに、新しい手法では、必要に応じて異なるノードロールに異なるサブスクリプションを適用します。

- オーバークラウドは、デフォルトの ML2 メカニズムドライバーとして Open Virtual Network (OVN) を使用するようになりました。Open vSwitch (OVS) サービスを OVN に移行することができます。この移行は、アップグレードが正常に完了した後に実施します。

- アンダークラウドおよびオーバークラウドは、共に Red Hat Enterprise Linux 8 上で動作します。

- **openstack-tripleo-heat-templates** には、**deployment** ディレクトリーに統一されたコンポーザブルサービステンプレートコレクションが含まれています。このディレクトリーに含まれるテンプレートのコンテンツは、コンテナー化されたサービスと Puppet ベースのコンポーザブルサービスの両テンプレートからのコンテンツをマージしたものです。

- OpenStack Data Processing サービス (sahara) はサポートされなくなりました。

重要

お使いの Red Hat OpenStack Platform 13 環境で sahara を有効にしている場合には、このアップグレードを続行せず、Red Hat Global Support Services にお問い合わせください。

- Service Telemetry Framework (STF) を優先し、OpenStack Telemetry のコンポーネントは非推奨になりました。

2.3. RED HAT ENTERPRISE LINUX 8 の変更点

アンダークラウドおよびオーバークラウドは、共に Red Hat Enterprise Linux 8 上で動作します。これには、アンダークラウドおよびオーバークラウドに関連する新しいツールおよび機能が含まれます。

- アンダークラウドおよびオーバークラウドは Red Hat Container Toolkit を使用します。コンテナーライフサイクルをビルドおよび制御する **docker** に代わって、Red Hat Enterprise Linux 8 には、新しいコンテナーイメージをビルドするための **buildah** およびコンテナー管理用の **podman** が含まれます。

- Red Hat Enterprise Linux 8 には **docker-distribution** パッケージが含まれていません。アンダークラウドには、オーバークラウドノードにコンテナーイメージを提供するためのプライベート HTTP レジストリーが追加されました。

- Red Hat Enterprise Linux 7 から 8 へのアップグレードプロセスには、**leapp** ツールが使用されます。

- Red Hat Enterprise Linux 8 は、**ntp** サービスを使用しません。その代わりに、Red Hat Enterprise Linux 8 では **chronyd** が使用されます。

- Red Hat Enterprise Linux 8 には、新しいバージョンの高可用性ツールが含まれています。

Red Hat OpenStack Platform 16.1 は、ベースオペレーティングシステムとして Red Hat Enterprise Linux 8.2 を使用します。アップグレードプロセスの一環として、ノードのベースオペレーティングシステムを Red Hat Enterprise Linux 8.2 にアップグレードします。

Red Hat Enterprise Linux 7 と 8 の主な相違点は、『RHEL 8 の導入における検討事項』を参照してください。Red Hat Enterprise Linux 8 に関する一般的な情報は、「Product Documentation for Red Hat Enterprise Linux 8」を参照してください。

2.4. RED HAT OPENSTACK PLATFORM での LEAPP アップグレードの使用

ロングライフバージョンの Red Hat OpenStack Platform のアップグレードには、Red Hat Enterprise Linux 7 から Red Hat Enterprise Linux 8 へのアップグレードも必要です。Red Hat Enterprise Linux 7 には **leapp** と呼ばれるツールが含まれており、このツールにより Red Hat Enterprise Linux 8 へのアップグレードを実施します。オペレーティングシステムのアップグレードを実施する際には、アンダークラウドとオーバークラウドではそれぞれ別個のプロセスが使用されます。

アンダークラウドのプロセス

openstack undercloud upgrade コマンドを実行する前に、手動で **leapp** によるアップグレードを行います。アンダークラウドのアップグレードには、**leapp** によるアップグレードを実施する手順が含まれます。

オーバークラウドのプロセス

オーバークラウドのアップグレードフレームワークでは、**leapp** によるアップグレードが自動的に実施されます。

制限

アップグレードに影響を及ぼす可能性のある制限の詳細は、**『RHEL 7 から RHEL 8 へのアップグレード』**の以下のセクションを参照してください。

- アップグレードの計画
- 既知の問題

特に、ディスク全体またはパーティションで暗号化が使用される (LUKS 暗号化、ファイルシステムの暗号化など) ノードで Leapp によるアップグレードを行うことはできません。この制限は、**dmcrypt: true** パラメーターで設定した Ceph OSD ノードに影響します。

既知の制限がお使いの環境に影響を及ぼす場合は、Red Hat Technical Support Team にアドバイスを求めてください。

2.5. サポート対象のアップグレードシナリオ

アップグレードを進める前に、ご自分のオーバークラウドがサポート対象であることを確認してください。

注記

以下の一覧に記載されていない特定のシナリオがサポート対象かどうか不明な場合は、Red Hat Technical Support Team にアドバイスを求めてください。

サポート対象のシナリオ

以下のインプレースアップグレードシナリオは、テスト済みでサポートの対象です。

- デフォルトのロール種別を使用する標準環境: Controller、Compute、および Ceph Storage OSD

- 分割 Controller コンポーザブルロール
- Ceph Storage コンポーザブルロール
- ハイパーコンバージドインフラストラクチャー: 同一ノード上の Compute サービスおよび Ceph Storage OSD サービス
- ネットワーク機能仮想化 (NFV) 技術を使用する環境: Single-root Input/Output Virtualization (SR-IOV) および Data Plane Development Kit (DPDK)

テクノロジープレビューのシナリオ

アップグレードフレームワークを以下の機能と併用した場合は、テクノロジープレビューとみなされます。したがって、Red Hat では全面的にはサポートしていません。以下のシナリオは概念実証の環境でのみテストし、実稼働環境へのアップグレードは行わないでください。テクノロジープレビュー機能についての詳しい情報は、「対象範囲の詳細」を参照してください。

- エッジおよび分散コンピュートノード (DCN) のシナリオ

テストされていないサポート対象外のシナリオ

以下に示すインプレースアップグレードのシナリオは、テストされていないか、サポート対象外かのいずれかです。ご自分のオーバークラウド設定が以下のシナリオ一覧のいずれかと一致する場合は、アップグレードを延期して Red Hat Technical Support Team にアドバイスを求めてください。

- インスタンス HA が有効な環境

2.6. 外部の CEPH デプロイメントと組み合わせたアップグレードに関する考慮事項

別途 Red Hat Ceph Storage システムをデプロイしていて、director を使用して OpenStack をデプロイおよび設定している場合は、Red Hat OpenStack Platform のアップグレードフレームワークを使用して、外部の Ceph デプロイメントと共にインプレースアップグレードを行うことができます。このシナリオは、director を使用してデプロイされた Ceph クラスターのアップグレードとは異なります。

外部の Ceph デプロイメントと組み合わせたインプレースアップグレードのプランニングおよび準備を行う際に考慮すべき違いは以下のとおりです。

1. Red Hat OpenStack Platform デプロイメントをバージョン 13 からバージョン 16.1 にアップグレードする前に、Red Hat Ceph Storage クラスターをバージョン 3 からバージョン 4 にアップグレードする必要があります。詳細は、**Red Hat Ceph Storage 4『インストールガイド』**の「Red Hat Ceph Storage クラスターのアップグレード」 を参照してください。

2. Red Hat Ceph Storage クラスターをバージョン 3 からバージョン 4 にアップグレードした後も、Red Hat OpenStack Platform 13 では引き続き RHCSv3 クライアントコンポーネントが実行されている可能性がありますが、これらは RHCSv4 クラスターに対して互換性があります。

3. 『13 から 16.1 へのアップグレードフレームワーク』に記載のアップグレードパスに従うことができます。該当する場合は、この特定のシナリオをサポートする条件ステップを実行する必要があります。条件ステップは、「外部の Ceph デプロイメントと共にアップグレードする場合は」で始まる箇所です。

4. 外部の Ceph デプロイメントと共にアップグレードする場合は、オーバークラウドのアップグレードプロセスの一部として RHCSv4 **ceph-ansible** をインストールします。director を使用してデプロイされた Ceph クラスターをアップグレードする場合は、オーバークラウドのアップグレードプロセスの完了後に RHCSv4 **ceph-ansible** をインストールします。

2.7. アップグレードを妨げる可能性のある既知の問題

アップグレードの正常な完了に影響を及ぼす可能性のある、以下の既知の問題を確認してください。

BZ#1852360: swift_rsync and swift_rsync_healthcheck moved in failed state after UC upgrade

Red Hat OpenStack Platform 16.1 アンダークラウド上のコンテナー化された **tripleo_swift_rsync** サービスが failed の状態を報告します。これは、**xinetd** を介して OpenStack Object Storage (swift) **rsync** デーモンを実行する Red Hat OpenStack Platform 16.1 バージョンとの競合によるものです。本ガイドでは、アンダークラウドのアップグレード前に削除する必要のあるパッケージの一覧に **xinetd** が含まれています。Red Hat は、16.1 の次回マイナーリリースの更新でこの問題を解決することを目指しています。

BZ#1866479: container-tools module stream not enabled correctly on update from 16.0.2 to 16.1

Red Hat Enterprise Linux 8 AppStream リポジトリーは、デフォルトの **dnf** モジュールとして **container-tools:rhel8** を使用します。Red Hat OpenStack Platform 16.1 がサポートされる正しいバージョンの Podman を使用するには、オーバークラウドおよびアンダークラウドで **container-tools:2.0** モジュールへの変更が必要です。アンダークラウドについてはこのモジュールを手動で変更しますが、オーバークラウド用には director がアップグレードプロセス中にこのモジュールを自動的に変更する必要があります。本ガイドには、**UpgradeInitCommand** パラメーターを使用してオーバークラウドでこのモジュール変更を行う回避策が含まれています。

BZ#1871834: ovn controllers cannot connect ovn dbs during 13-16.1 FFU upgrade

Open Virtual Network (OVN) の仮想 IP アドレス変更により、OVN コントローラーはアップグレード中に OVN データベースに接続できなくなります。この問題を回避するには、以下の手順を実行します。

1. コントローラーノードをアップグレードしたら、新たな OVN 仮想 IP アドレスを取得します。

```
[root@controller-0 ~]# ovs-vsctl get open . external_ids:ovn-remote | sed -e 's/\"//g'
```

このコマンドにより、以下の形式で出力が生成されます。

```
tcp:<NewOvnVIP>:6642
```

2. コンピュートのアップグレードプロセスを開始する前に、それぞれのコンピュートノードにログインして OVN 仮想 IP アドレスを設定します。

```
[root@compute-0 ~]# sudo docker exec -uroot ovn_controller ovs-vsctl set Open_Vswitch
. external_ids:ovn-remote="tcp:<NewOvnVIP>:6642"
```

Red Hat は、16.1 の次回マイナーリリースの更新でこの問題を解決することを目指しています。

BZ#1885212: ovn-controllers can't connect to SB DB in hybrid state

+

OVN を使用する OpenStack Platform 13 環境では、アップグレード中に OVN Southbound データベース接続に問題が生じます。コンピュートノードで実行されている **ovn-controller** は OVN 2.11 を使用しますが、コントローラーノード上の Southbound データベースは OVN 2.13 を使用します。これにより、アップグレード中に負荷の移行などコンピュートベースの操作を実施する際に問題が発生します。Red Hat は、16.1 の次回マイナーリリースの更新でこの問題を解決することを目指しています。

BZ#1882400: During FFU new vms with SRIOV ports cannot be created

+ --- ポートバインディングの変更により、コンピュートノードがハイブリッドの状態である間 SR-IOV ポートが設定された新しい仮想マシンを作成することはできません。アップグレードを開始する前、またはアップグレードが正常に完了した後に、SR-IOV ポートが設定された仮想マシンを作成します。詳しくは、「During Framework Upgrade from OpenStack 13 to 16.1 new instances with SR-IOV ports cannot be created」を参照してください。 ---

Red Hat Ceph Storage の問題

BZ#1855813: Ceph tools repo should be switched from RHCS3 to RHCS4 only after converge, before running external-upgrade

アンダークラウド上の **ceph-ansible** Playbook コレクションにより、オーバークラウド上に Red Hat Ceph Storage コンテナーがデプロイされます。環境をアップグレードするには、アップグレードプロセス中 Ceph Storage 3 コンテナーを維持するために、Red Hat Ceph Storage 3 バージョンの **ceph-ansible** が必要です。本ガイドには、Ceph Storage 4 へのアップグレード準備が整うまで、アップグレードプロセス中 **ceph-ansible** バージョン 3 を維持する手順が含まれています。13 から 16.1 へのアップグレードを実施する前に、Red Hat OpenStack Platform 13 環境のマイナーバージョン更新を実施し、**ceph-ansible** のバージョンが 3.2.46 以降になるようにしてください。

BZ#1870617: noout ceph flag not unset during the FFU workflow

本ガイドには、Ceph Storage OSD クラスターのアップグレード前に **noout** および **norebalance** フラグを設定し、Ceph Storage OSD クラスターのアップグレードが完了したら設定を解除するためのコマンドが含まれています。Red Hat は、16.1 の次回マイナーリリースの更新でこの手順を自動化することを目指しています。

Red Hat Enterprise Linux の問題

BZ#1861977: RHSA-2020:3216 grub2 security update renders system unbootable

RHSA-2020:3216 セキュリティーアップデートを適用すると、grub メニューが読み込まれなくなります。この問題は、**shim-x64-15-14.el8_2** パッケージの使用時に発生します。**shim-x64-15-15.el8_2** およびそれ以降のバージョンのパッケージでは、この問題が修正されています。ノードのリブート時に grub メニューが表示されない場合は、「System hangs after POST and the grub menu never loads after applying the RHSA-2020:3216 or RHSA-2020:3217」を参照してください。

2.8. バックアップおよびリストア

Red Hat OpenStack Platform 13 環境をアップグレードする前に、アンダークラウドおよびオーバークラウドのコントロールプレーンをバックアップします。Relax-and-recover (ReaR) ユーティリティーを使用したノードのバックアップに関する詳細は、**『アンダークラウドとコントロールプレーンのバックアップおよびリストア』**を参照してください。

- アップグレードを実施する前にノードをバックアップします。アップグレード前のノードバックアップの詳細については、『Red Hat OpenStack Platform 13 アンダークラウドとコントロールプレーンのバックアップおよびリストア』を参照してください。
- アップグレードした後に各ノードをバックアップすることができます。アップグレードしたノードのバックアップに関する詳しい情報は、『Red Hat OpenStack Platform 16.1 アンダークラウドとコントロールプレーンのバックアップおよびリストア』を参照してください。

2.9. マイナーバージョンの更新

Red Hat OpenStack Platform 環境をアップグレードする前に、環境を現在のリリースの最新マイナーバージョンに更新します。たとえば、Red Hat OpenStack Platform 16.1 へのアップグレードを実施する前に、お使いの Red Hat OpenStack Platform 13 環境を最新の 13 に更新します。

Red Hat OpenStack Platform 13 のマイナーバージョンの更新を実施する手順は、『Red Hat OpenStack Platform の最新状態の維持』を参照してください。

2.10. プロキシー設定

Red Hat OpenStack Platform 13 環境でプロキシーを使用している場合、**/etc/environment** ファイルのプロキシー設定は、オペレーティングシステムのアップグレードおよび Red Hat OpenStack Platform 16.1 へのアップグレード後も維持されます。

- Red Hat OpenStack Platform 13 のプロキシー設定についての詳しい情報は、「アンダークラウドプロキシーの設定」を参照してください。
- Red Hat OpenStack Platform 16.1 のプロキシー設定についての詳しい情報は、「アンダークラウドプロキシーの設定」を参照してください。

2.11. アップグレード前の RED HAT OPENSTACK PLATFORM 13 の検証

Red Hat OpenStack Platform 16.1 にアップグレードする前に、**tripleo-validations** Playbook を使用してアンダークラウドおよびオーバークラウドを検証します。Red Hat OpenStack Platform 13 において、これらの Playbook を OpenStack Workflow サービス (mistral) を使用して実行します。

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

```
$ source ~/stackrc
```

3. 以下の内容で、**pre-upgrade-validations.sh** という名前の bash スクリプトを作成します。

```
#!/bin/bash
for VALIDATION in $(openstack action execution run tripleo.validations.list_validations
'{"groups": ["pre-upgrade"]}' | jq ".result[] | .id")
do
  echo "=== Running validation: $VALIDATION ==="
  STACK_NAME=$(openstack stack list -f value -c 'Stack Name')
  ID=$(openstack workflow execution create -f value -c ID tripleo.validations.v1.run_validation
"{\"validation_name\": $VALIDATION, \"plan\": \"$STACK_NAME\"}")
  while [ $(openstack workflow execution show $ID -f value -c State) == "RUNNING" ]
  do
    sleep 1
  done
  echo ""
  openstack workflow execution output show $ID | jq -r ".stdout"
  echo ""
done
```

4. スクリプトを実行する権限を追加します。

```
$ chmod +x pre-upgrade-validations.sh
```

5. スクリプトを実行します。

```
$ ./pre-upgrade-validations.sh
```

スクリプトの出力を確認し、成功した検証と失敗した検証を確認します。

```
=== Running validation: "check-ftype" ===

Success! The validation passed for all hosts:
* undercloud
```

第3章 リポジトリー

本項では、アンダークラウドおよびオーバークラウドのリポジトリーについて説明します。特定の状況において、リポジトリーを有効にする必要がある場合は、本項を参照してください。

- Red Hat カスタマーポータルに登録する際にリポジトリーを有効にする。
- リポジトリーを有効にして Red Hat Satellite サーバーに同期させる。
- Red Hat Satellite サーバーに登録する際にリポジトリーを有効化する。

3.1. アンダークラウドのリポジトリー

Red Hat OpenStack Platform 16.1 は、Red Hat Enterprise Linux 8.2 上で動作します。したがって、これらのリポジトリーからのコンテンツを該当する Red Hat Enterprise Linux バージョンにロックする必要があります。

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 8.2 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ではなく **rhel-8-for-x86_64-baseos-eus-rpms** のままです。

警告

ここで指定する以外のリポジトリーは、サポートされません。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 8 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-8-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 8 for x86_64 - AppStream (RPMs)	**rhel-8-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 8 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-8-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux の高可用性ツール。コントローラーノードの高可用性に使用します。

名前	リポジトリー	要件の説明
Red Hat Ansible Engine 2.9 for RHEL 8 x86_64 (RPMs)	**ansible-2.9-for-rhel-8-x86_64-rpms**	Red Hat Enterprise Linux 用 Ansible エンジン。最新バージョンの Ansible を提供するために使用されます。
Advanced Virtualization for RHEL 8 x86_64 (RPMs)	**advanced-virt-for-rhel-8-x86_64-rpms**	OpenStack Platform 用仮想化パッケージを提供します。
Red Hat Satellite Tools for RHEL 8 Server RPMs x86_64	**satellite-tools-6.5-for-rhel-8-x86_64-rpms**	Red Hat Satellite 6 でのホスト管理ツール
Red Hat OpenStack Platform 16.1 for RHEL 8 (RPMs)	**openstack-16.1-for-rhel-8-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 8 (RPMS)	**fast-datapath-for-rhel-8-x86_64-rpms**	OpenStack Platform 用 Open vSwitch (OVS) パッケージを提供します。

Ceph リポジトリー

以下の表には、アンダークラウド用の 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 Ceph Storage Tools 4 for RHEL 8 x86_64 (RPMs)	**rhceph-4-tools-for-rhel-8-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。オーバークラウドで Ceph Storage を使用する場合、または既存の Ceph Storage クラスターと統合する場合、アンダークラウドにはこのリポジトリーからの **ceph-ansible** パッケージが必要です。

IBM POWER 用リポジトリー

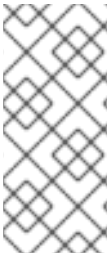
以下の表には、POWER PC アーキテクチャー上で Red Hat Openstack Platform を構築するためのリポジトリーを一覧にしてまとめています。コアリポジトリーの該当リポジトリーの代わりに、これらのリポジトリーを使用してください。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux for IBM Power, little endian - BaseOS (RPMs)	**rhel-8-for-ppc64le-baseos-rpms**	ppc64le システム用ベースオペレーティングシステムのリポジトリー

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 8 for IBM Power, little endian - AppStream (RPMs)	**rhel-8-for-ppc64le-appstream-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 8 for IBM Power, little endian - High Availability (RPMs)	**rhel-8-for-ppc64le-highavailability-rpms**	Red Hat Enterprise Linux の高可用性ツール。コントローラーノードの高可用性に使用します。
Red Hat Ansible Engine 2.8 for RHEL 8 IBM Power, little endian (RPMs)	**ansible-2.8-for-rhel-8-ppc64le-rpms**	Red Hat Enterprise Linux 用 Ansible エンジン。最新バージョンの Ansible を提供します。
Red Hat OpenStack Platform 16.1 for RHEL 8 (RPMs)	**openstack-16.1-for-rhel-8-ppc64le-rpms**	ppc64le システム用 Red Hat OpenStack Platform のコアリポジトリー

3.2. オーバークラウドのリポジトリー

Red Hat OpenStack Platform 16.1 は、Red Hat Enterprise Linux 8.2 上で動作します。したがって、これらのリポジトリーからのコンテンツを該当する Red Hat Enterprise Linux バージョンにロックする必要があります。



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 8.2 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ではなく **rhel-8-for-x86_64-baseos-eus-rpms** のままです。

警告

ここで指定する以外のリポジトリーは、サポートされません。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、オーバ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 8 for x86_64 - BaseOS (RPMs) Extended Update Support (EUS)	**rhel-8-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 8 for x86_64 - AppStream (RPMs)	**rhel-8-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 8 for x86_64 - High Availability (RPMs) Extended Update Support (EUS)	**rhel-8-for-x86_64-highavailability-eus-rpms**	Red Hat Enterprise Linux の高可用性ツール。
Red Hat Ansible Engine 2.9 for RHEL 8 x86_64 (RPMs)	**ansible-2.9-for-rhel-8-x86_64-rpms**	Red Hat Enterprise Linux 用 Ansible エンジン。最新バージョンの Ansible を提供するために使用されます。
Advanced Virtualization for RHEL 8 x86_64 (RPMs)	**advanced-virt-for-rhel-8-x86_64-rpms**	OpenStack Platform 用仮想化パッケージを提供します。
Red Hat Satellite Tools for RHEL 8 Server RPMs x86_64	**satellite-tools-6.5-for-rhel-8-x86_64-rpms**	Red Hat Satellite 6 でのホスト管理ツール
Red Hat OpenStack Platform 16.1 for RHEL 8 (RPMs)	**openstack-16.1-for-rhel-8-x86_64-rpms**	Red Hat OpenStack Platform のコアリポジトリー
Red Hat Fast Datapath for RHEL 8 (RPMS)	**fast-datapath-for-rhel-8-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 4 for RHEL 8 x86_64 (RPMs)	**rhceph-4-tools-for-rhel-8-x86_64-rpms**	Red Hat Enterprise Linux 8 での Red Hat Ceph Storage 4 用ツール

Ceph 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 8 for x86_64 - BaseOS (RPMs)	**rhel-8-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 8 for x86_64 - AppStream (RPMs)	**rhel-8-for-x86_64-appstream-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 8 for x86_64 - High Availability (RPMs)	**rhel-8-for-x86_64-highavailability-rpms**	Red Hat Enterprise Linux の高可用性ツール。
Red Hat Ansible Engine 2.9 for RHEL 8 x86_64 (RPMs)	**ansible-2.9-for-rhel-8-x86_64-rpms**	Red Hat Enterprise Linux 用 Ansible エンジン。最新バージョンの Ansible を提供するために使用されます。
Red Hat OpenStack Platform 16.1 Director Deployment Tools for RHEL 8 x86_64 (RPMs)	**openstack-16.1-deployment-tools-for-rhel-8-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
Red Hat Ceph Storage OSD 4 for RHEL 8 x86_64 (RPMs)	**rhceph-4-osd-for-rhel-8-x86_64-rpms**	(Ceph Storage ノード向け) Ceph Storage Object Storage デーモンのリポジトリー。Ceph Storage ノードにインストールします。
Red Hat Ceph Storage MON 4 for RHEL 8 x86_64 (RPMs)	**rhceph-4-mon-for-rhel-8-x86_64-rpms**	(Ceph Storage ノード向け) Ceph Storage Monitor デーモンのリポジトリー。Ceph Storage ノードを使用して OpenStack 環境にあるコントローラーノードにインストールします。
Red Hat Ceph Storage Tools 4 for RHEL 8 x86_64 (RPMs)	**rhceph-4-tools-for-rhel-8-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。Ceph Storage クラスターと共にオーバークラウドをデプロイする場合や、オーバークラウドを既存の Ceph Storage クラスターと統合する場合に、すべてのノードでこのリポジトリーを有効にします。

リアルタイムリポジトリー

以下の表には、リアルタイムコンピュート (RTC) 機能用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

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 8 for x86_64 - Real Time (RPMs)	**rhel-8-for-x86_64-rt-rpms**	リアルタイム KVM (RT-KVM) のリポジトリー。リアルタイムカーネルを有効化するためのパッケージが含まれています。RT-KVM 対象の全コンピュートノードで、このリポジトリーを有効にします。注記: このリポジトリーにアクセスするには、別途 **Red Hat OpenStack Platform for Real Time** SKU のサブスクリプションが必要です。
Red Hat Enterprise Linux 8 for x86_64 - Real Time for NFV (RPMs)	**rhel-8-for-x86_64-nfv-rpms**	NFV 向けのリアルタイム KVM (RT-KVM) のリポジトリー。リアルタイムカーネルを有効化するためのパッケージが含まれています。RT-KVM 対象の全 NFV コンピュートノードで、このリポジトリーを有効にします。注記: このリポジトリーにアクセスするには、別途 **Red Hat OpenStack Platform for Real Time** SKU のサブスクリプションが必要です。

IBM POWER 用リポジトリー

以下の表には、POWER PC アーキテクチャー上で OpenStack Platform を構築するためのリポジトリーをまとめています。コアリポジトリーの該当リポジトリーの代わりに、これらのリポジトリーを使用してください。

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux for IBM Power, little endian - BaseOS (RPMs)	**rhel-8-for-ppc64le-baseos-rpms**	ppc64le システム用ベースオペレーティングシステムのリポジトリー
Red Hat Enterprise Linux 8 for IBM Power, little endian - AppStream (RPMs)	**rhel-8-for-ppc64le-appstream-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 8 for IBM Power, little endian - High Availability (RPMs)	**rhel-8-for-ppc64le-highavailability-rpms**	Red Hat Enterprise Linux の高可用性ツール。コントローラーノードの高可用性に使用します。
Red Hat Ansible Engine 2.8 for RHEL 8 IBM Power, little endian (RPMs)	**ansible-2.8-for-rhel-8-ppc64le-rpms**	Red Hat Enterprise Linux 用 Ansible エンジン。最新バージョンの Ansible を提供するために使用されます。

名前	リポジトリー	要件の説明
Red Hat OpenStack Platform 16.1 for RHEL 8 (RPMs)	**openstack-16.1-for-rhel-8-ppc64le-rpms**	ppc64le システム用 Red Hat OpenStack Platform のコアリポジトリー

3.3. RED HAT SATELLITE 6 に関する考慮事項

Red Hat Satellite 6 を使用して Red Hat OpenStack Platform 環境用の RPM およびコンテナーイメージをホストする場合、Red Hat OpenStack Platform 16.1 のアップグレード中に Satellite 6 を使用してコンテンツを提供する際には、特定の考慮事項があります。

現在の環境についての仮定

- Satellite サーバーがすでに Red Hat OpenStack Platform 13 の RPM およびコンテナーイメージをホストしている。
- Red Hat OpenStack Platform 13 環境内の全ノードをすでに Satellite サーバーに登録している。たとえば、以前に Red Hat OpenStack Platform 13 のコンテンツビューにリンクされたアクティベーションキーを使用して、ノードを OpenStack Platform 13 のコンテンツに登録した。

Red Hat OpenStack Platform のアップグレードに関する推奨事項

- Red Hat OpenStack Platform 13 のアンダークラウドおよびオーバークラウドの両方に必要な RPM リポジトリーを有効にして同期します。これには、必要な Red Hat Enterprise Linux 8.2 リポジトリーが含まれます。
- Satellite サーバーにカスタム製品を作成し、以下の Red Hat OpenStak Platform バージョン用のコンテナーイメージをホストします。
  - Red Hat OpenStack Platform 16.1
  - Red Hat OpenStack Platform 15
- Red Hat OpenStack Platform 16.1 アップグレード用のコンテンツビューを作成してプロモートし、以下のコンテンツをコンテンツビューに含めます。
  - Red Hat Enterprise Linux 8.2 リポジトリーを含む、アンダークラウドおよびオーバークラウドの全 RPM リポジトリー
  - Red Hat OpenStack Platform 16.1 コンテナーイメージ
  - Red Hat OpenStack Platform 15 コンテナーイメージ
- Red Hat OpenStack Platform 16.1 へのアップグレード用のアクティベーションキーを作成し、そのアクティベーションキーを Red Hat OpenStack Platform 16.1 のコンテンツビューに関連付けます。このアクティベーションキーを Red Hat Enterprise Linux 8.2 に固定します。
- どのノードにも **katello-host-tools-fact-plugin** パッケージがインストールされていないことを確認します。Leapp によるアップグレードではこのパッケージがアップグレードされず、パッケージが Red Hat Enterprise Linux 8.2 システムに残されるので、**subscription-manager** がエラーを報告します。

パート I. アンダークラウドのアップグレード

第4章 アンダークラウドアップグレードの準備

アンダークラウドのアップグレードを実施する前に、アンダークラウドのアップグレードが正常に実行されるように準備の手順を完了する必要があります。

4.1. 外部の CEPH と組み合わせたアップグレードの前提条件

外部の Ceph デプロイメントと共にアップグレードする場合、Red Hat OpenStack Platform デプロイメントをアップグレードする前に、Red Hat Ceph Storage クラスターをバージョン 3 からバージョン 4 にアップグレードする必要があります。詳細は、**Red Hat Ceph Storage 4『インストールガイド』**の「Red Hat Ceph Storage クラスターのアップグレード」 を参照してください。

4.2. 新たなメモリー要件

Red Hat OpenStack Platform 16.1 では、アンダークラウドに新たなメモリー要件が適用されます。

Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 16.1
16 GB RAM	24 GB RAM

アップグレードを続行する前に、アンダークラウドがこれらの新たな要件を満たすことを確認してください。

4.3. 予測可能なアンダークラウドノード NIC 名の使用

アンダークラウドノードで Leapp によるアップグレードを実施する前に、カーネルベースの NIC 名が使用されているかどうかを確認する必要があります。この NIC 名には、通常 **eth** の接頭辞が含まれます。NIC の割り当てに関して、通常これらの NIC 名は予測可能ではありません。ノード上のいずれかの NIC で **eth** の接頭辞が使用されていると、Leapp によるアップグレードに失敗します。

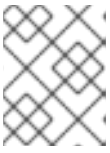
手順

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. **playbook-nics.yaml** という名前の Ansible Playbook を作成し、以下のコンテンツを Playbook にコピーします。

```
---
- name: Rename eth devices
  hosts: all
  become: yes
  vars:
    prefix: "em"
  tasks:
    - set_fact:
        eth_interfaces: "{{ ansible_interfaces | select('match','eth.*') | list }}"
    - debug:
        msg: "{{ eth_interfaces }}"
    - name: Update udev rules
      lineinfile:
        line: "SUBSYSTEM==\"net\", ACTION==\"add\", DRIVERS==\"?*\", ATTR{address}==\"
{{ ansible_facts[item]['perm_macaddress'] | default(ansible_facts[item]['macaddress']) }}\",
```

```
NAME=\"{{ item|replace('eth',prefix) }}\""
        path: /etc/udev/rules.d/70-rhosp-persistent-net.rules
        create: True
      with_items: "{{ eth_interfaces }}"
    - name: Rename eth files
      block:
        - name: Check that eth files exists
          stat:
            path: /etc/sysconfig/network-scripts/ifcfg-{{ item }}
          register: nic_result
          with_items: "{{ eth_interfaces }}"
        - name: Copy nic files using the new prefix
          copy:
            remote_src: True
            src: "{{ item.stat.path }}"
            dest: "{{ item.stat.path|replace('eth',prefix) }}"
          with_items: "{{ nic_result.results }}"
          when: item.stat.exists
        - name: Edit NAME in new network-script files
          lineinfile:
            regexp: "^NAME=.*"
            line: "NAME={{ item.item|replace('eth',prefix) }}"
            path: "{{ item.stat.path|replace('eth',prefix) }}"
          with_items: "{{ nic_result.results }}"
          when: item.stat.exists
        - name: Edit DEVICE in new network-script files
          lineinfile:
            regexp: "^DEVICE=.*"
            line: "DEVICE={{ item.item|replace('eth',prefix) }}"
            path: "{{ item.stat.path|replace('eth',prefix) }}"
          with_items: "{{ nic_result.results }}"
          when: item.stat.exists
        - name: Backup old eth network-script files
          copy:
            remote_src: True
            src: "{{ item.stat.path }}"
            dest: "{{ item.stat.path }}.bak"
          with_items: "{{ nic_result.results }}"
          when: item.stat.exists
        - name: Remove old eth network-script files
          file:
            path: "{{ item.stat.path }}"
            state: absent
          with_items: "{{ nic_result.results }}"
          when: item.stat.exists
    - name: Rename route files
      block:
        - name: Check that route files exists
          stat:
            path: /etc/sysconfig/network-scripts/route-{{ item }}
          register: route_result
          with_items: "{{ eth_interfaces }}"
        - name: Copy route files using the new prefix
          copy:
            remote_src: True
            src: "{{ item.stat.path }}"
```

```
      dest: "{{ item.stat.path|replace('eth',prefix) }}"
    with_items: "{{ route_result.results }}"
    when: item.stat.exists
  - name: Update prefix in route files that use IP command arguments format
    replace:
      regexp: "eth"
      replace: "{{ prefix }}"
      path: "{{ item.stat.path|replace('eth',prefix) }}"
    with_items: "{{ route_result.results }}"
    when: item.stat.exists
  - name: Backup old route files
    copy:
      remote_src: True
      src: "{{ item.stat.path }}"
      dest: "{{ item.stat.path }}.bak"
    with_items: "{{ route_result.results }}"
    when: item.stat.exists
  - name: Remove old route files
    file:
      path: "{{ item.stat.path }}"
      state: absent
    with_items: "{{ route_result.results }}"
    when: item.stat.exists
- name: Perform a final regex for any remaining eth prefixes in ifcfg files
  block:
    - name: Get a list of all ifcfg files
      find:
        paths: /etc/sysconfig/network-scripts/
        patterns: 'ifcfg-*'
      register: ifcfg_files
    - name: Perform final regex on ifcfg files
      replace:
        path: "{{ item[0].path }}"
        regexp: "{{ item[1] }}"
        replace: "{{ item[1]|replace('eth',prefix) }}"
      with_nested:
        - "{{ ifcfg_files.files }}"
        - "{{ eth_interfaces }}"
```



注記

この Playbook を使用して、アップグレードプロセスの後半でオーバークラウドの NIC の名前を変更します。

3. アンダークラウドで **playbook-nics.yaml** Playbook を実行します。

```
$ ansible-playbook -c local -i localhost, playbook-nics.yaml
```

この Playbook により、新しい NIC の接頭辞が **em** に設定されます。別の NIC 接頭辞を設定するには、Playbook の実行時に **prefix** 変数を設定します。

```
$ ansible-playbook -c local -i localhost, -e prefix="mynic" ~/playbook-nics.yaml
```

4. 標準のリブート手順を使用して、アンダークラウドノードをリブートします。詳しくは、「ノードのリブート」を参照してください。

リソース

- 「RHEL 7 でカーネルの NIC 名を使用している場合に RHEL 8 へのインプレースアップグレードを実行する方法」
- 『ネットワークガイド』の「命名スキームの序列」
- 『ネットワークガイド』の「予想可能なネットワークインターフェースデバイスの命名について」

4.4. アンダークラウドでの SSH ROOT 権限パラメーターの設定

Leapp によるアップグレードでは、**PermitRootLogin** パラメーターが **/etc/ssh/sshd_config** ファイルに存在するかどうかを確認します。このパラメーターを、明示的に **yes** または **no** のいずれかに設定する必要があります。

セキュリティー上の理由から、アンダークラウドで root ユーザーへの SSH アクセスを無効にするには、このパラメーターを **no** に設定します。

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. **/etc/ssh/sshd_config** ファイルに **PermitRootLogin** パラメーターがあるかどうかを確認します。

```
$ sudo grep PermitRootLogin /etc/ssh/sshd_config
```

3. **/etc/ssh/sshd_config** ファイルにパラメーターがない場合は、ファイルを編集して **PermitRootLogin** パラメーターを設定します。

```
PermitRootLogin no
```

4. ファイルを保存します。

4.5. 次世代電源管理ドライバーへの移行

Red Hat OpenStack Platform では **ハードウェアタイプ** とも呼ばれる次世代ドライバーが使用され、従来のドライバーがこれに置き換えられています。

従来のドライバーとそれと等価な次世代ハードウェアタイプの対比を、以下の表に示します。

従来のドライバー	新しいハードウェアタイプ
pxe_ipmitool	**ipmi**
pxe_drac	**idrac**
pxe_ilo	**ilo**

従来のドライバー	新しいハードウェアタイプ
pxe_irmc	**irmc**
VBMC (**pxe_ipmitool**)	**ipmi**
fake_pxe	**fake-hardware**

OpenStack Platform 15 では、これらの従来ドライバーは削除され、使用できなくなっています。OpenStack Platform 15 に **アップグレードする前に** ハードウェアタイプに変更する必要があります。

手順

1. 有効なハードウェアタイプの最新の一覧を確認します。

```
$ source ~/stackrc
$ openstack baremetal driver list --type dynamic
```

2. 有効ではないハードウェアタイプのドライバーを使用する場合には、**undercloud.conf** ファイルの **enabled_hardware_types** パラメーターを使用してそのドライバーを有効にします。

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

3. ファイルを保存し、アンダークラウドをリフレッシュします。

```
$ openstack undercloud install
```

4. 以下のコマンドを実行します。**OLDDRIVER** および **NEWDRIVER** 変数を、実際の電源管理タイプに置き換えてください。

```
$ source ~/stackrc
$ OLDDRIVER="pxe_ipmitool"
$ NEWDRIVER="ipmi"
$ for NODE in $(openstack baremetal node list --driver $OLDDRIVER -c UUID -f value) ; do
openstack baremetal node set $NODE --driver $NEWDRIVER; done
```

第5章 アンダークラウドのオペレーティングシステムのアップグレード

director をアップグレードする前に、アンダークラウドのオペレーティングシステムを Red Hat Enterprise Linux 7 から Red Hat Enterprise Linux 8 にアップグレードする必要があります。このオペレーティングシステムのアップグレードの一環として、Red Hat OpenStack Platform 13 のパッケージを削除し、続いて Leapp ユーティリティーを実行してシステムパッケージをアップグレードする必要があります。このパッケージの削除およびオペレーティングシステムのアップグレードは、アンダークラウドのデータベースには影響を及ぼしません。オペレーティングシステムのアップグレードが完了したら、Red Hat OpenStack Platform 16.1 バージョンの director パッケージを再インストールします。

5.1. RED HAT OPENSTACK PLATFORM DIRECTOR パッケージの削除

Leapp ユーティリティーを実行する前に、Red Hat Enterprise Linux 7 に関連付けられた Red Hat OpenStack Platform 13 パッケージを削除します。これらのパッケージ名には、リリースに関する接尾辞 **el7ost** が使用されます。一部の **el7ost** は、**subscription-manager** および Leapp ユーティリティーの依存関係としてシステム上に残ります。

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. アンダークラウド上の主要 OpenStack サービスを無効にします。

```
$ sudo systemctl stop openstack-* httpd haproxy mariadb rabbitmq* docker xinetd
```

3. OpenvSwitch およびアップグレードに必要な特定の Python 2 パッケージは除き、アンダークラウドから主要 OpenStack サービスを削除します。

```
$ sudo yum -y remove *el7ost* galera* haproxy* \
    httpd mysql* pacemaker* xinetd python-jsonpointer \
    qemu-kvm-common-rhev qemu-img-rhev rabbit* \
    redis* \
    -- \
    -*openvswitch* -python-docker -python-PyMySQL \
    -python-pysocks -python2-asn1crypto -python2-babel \
    -python2-cffi -python2-cryptography -python2-dateutil \
    -python2-idna -python2-ipaddress -python2-jinja2 \
    -python2-jsonpatch -python2-markupsafe -python2-pyOpenSSL \
    -python2-requests -python2-six -python2-urllib3 \
    -python-httplib2 -python-passlib -python2-netaddr -ceph-ansible
```

4. **/etc/httpd** および **/var/lib/docker** ディレクトリーからコンテンツを削除します。

```
$ sudo rm -rf /etc/httpd /var/lib/docker
```

5.2. アンダークラウドでの LEAPP アップグレードの実施

Leapp ユーティリティーをインストールして実行し、オペレーティングシステムを Red Hat Enterprise Linux 8 にアップグレードします。

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. Leapp ユーティリティーをインストールします。

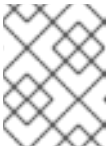
```
$ sudo yum install leapp
```

3. ナレッジベースのアーティクル「RHEL 7 から RHEL 8 へのインプレースアップグレード時に Leapp ユーティリティーで必要なデータ」に添付されている追加の必須データファイル (RPM パッケージの変更および RPM リポジトリーマッピング) をダウンロードし、それらのファイルを **/etc/leapp/files/** ディレクトリーに置きます。
4. Red Hat サブスクリプションを更新します。
   - アンダークラウドの登録に Red Hat カスタマーポータルを使用している場合、現在のサブスクリプションをリフレッシュし、Red Hat Enterprise Linux 8.2 コンテンツへのアクセス権限を取得します。

```
$ sudo subscription-manager refresh
```

   - アンダークラウドの登録に Red Hat Satellite サーバーを使用している場合は、アンダークラウドを Red Hat OpenStack Platform 16.1 のアクティベーションキーに関連付けられたコンテンツビューに再登録します。

```
$ sudo subscription-manager register --force --org ORG --activationkey
ACTIVATION_KEY --release 8.2
```



注記

Red Hat OpenStack Platform 16.1 用に作成するコンテンツビューには、Red Hat Enterprise Linux 8.2 のコンテンツが含まれている必要があります。

5. Red Hat OpenStack Platform 16.1 では、新しいバージョンの Open vSwitch が使用されます。**to_remove** および **to_install** トランザクションファイルにより、Open vSwitch のバージョンを置き換えます。

```
$ echo 'openvswitch2.11' | sudo tee -a /etc/leapp/transaction/to_remove
$ echo 'openvswitch2.13' | sudo tee -a /etc/leapp/transaction/to_install
```

6. **to_keep** トランザクションファイルを使用して、アップグレードプロセス中 Red Hat Ceph Storage 3 バージョンの **ceph-ansible** を維持します。

```
$ echo 'ceph-ansible' | sudo tee -a /etc/leapp/transaction/to_keep
```

7. Leapp によるアップグレードプロセスを開始します。

```
$ sudo leapp upgrade --debug --enablerepo rhel-8-for-x86_64-baseos-eus-rpms --
enablerepo rhel-8-for-x86_64-appstream-eus-rpms --enablerepo fast-datapath-for-rhel-8-
x86_64-rpms --enablerepo ansible-2.9-for-rhel-8-x86_64-rpms
```

--enablerepo オプションを使用して、Leapp によるアップグレードプロセス中に有効にするリポジトリーを設定します。特に新しいバージョンの Open vSwitch を使用する場合は、Red Hat OpenStack Platform 16.1 への移行を円滑に行うために、これらのリポジトリーを追加する必要があります。

8. **leapp upgrade** コマンドが正常に完了するのを待ちます。
9. ルートディレクトリーに空の **.autorelabel** ファイルを作成します。

```
$ sudo touch /.autorelabel
```

リブート後、SELinux はこのファイルを検出し、自動的にファイルシステムのラベルを変更します。

10. アンダークラウドをリブートします。

```
$ sudo reboot
```

リソース

- 「RHEL 7 から RHEL 8 へのインプレースアップグレード時に Leapp ユーティリティーで必要なデータ」
- 「RHEL 7 でカーネルの NIC 名を使用している場合に RHEL 8 へのインプレースアップグレードを実行する方法」
- 『RHEL 7 から RHEL 8 へのアップグレード』

第6章 DIRECTOR のアップグレード

アンダークラウドのオペレーティングシステムのアップグレードが完了したら、director をアップグレードします。以前の Red Hat OpenStack Platform 13 のアンダークラウドのデータベースは、オペレーティングシステムのアップグレード後にホスト上に残ります。**openstack undercloud upgrade** コマンドを実行する前に、新しい Red Hat OpenStack Platform 16.1 パッケージをインストールし、Red Hat OpenStack Platform 16.1 コンテナーイメージの新しいソースを設定します。

6.1. 環境の RED HAT ENTERPRISE LINUX リリースへのロック

Red Hat OpenStack Platform 16.1 は Red Hat Enterprise Linux 8.2 でサポートされています。更新を実施する前に、オペレーティングシステムをより新しいマイナーリリースにアップグレードしないように、アンダークラウドのリポジトリーを Red Hat Enterprise Linux 8.2 リリースにロック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. **subscription-manager release** コマンドを使用して、アンダークラウドを特定のバージョンにロックします。

```
$ sudo subscription-manager release --set=8.2
```

6.2. 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 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-8-for-x86_64-baseos-eus-rpms --enable=rhel-8-for-x86_64-appstream-eus-rpms --enable=rhel-8-for-x86_64-highavailability-eus-rpms --enable=ansible-2.9-for-rhel-8-x86_64-rpms --enable=openstack-16.1-for-rhel-8-x86_64-rpms --enable=fast-datapath-for-rhel-8-x86_64-rpms --enable=advanced-virt-for-rhel-8-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. **container-tools** リポジトリーモジュールをバージョン **2.0** に設定し ます。

```
[stack@director ~]$ sudo dnf module disable -y container-tools:rhel8
[stack@director ~]$ sudo dnf module enable -y container-tools:2.0
```

4. **virt** リポジトリーモジュールをバージョン **8.2** に設定します。

```
[stack@director ~]$ sudo dnf module disable -y virt:rhel
[stack@director ~]$ sudo dnf module enable -y virt:8.2
```

5. オペレーティングシステムを同期し、システムパッケージがオペレーティングシステムのバージョンと一致するようにします。

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

6.3. DIRECTOR パッケージのインストール

Red Hat OpenStack Platform director に関連するパッケージをインストールします。

手順

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

```
[stack@director ~]$ sudo dnf install -y python3-tripleoclient
```

6.4. 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。この環境ファイルを生成してカスタマイズし、コンテナーイメージを準備するのに使用します。

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. 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。このオプションにより、director は必要なイメージを Red Hat Container Catalog からプルし、そのイメージ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 はアンダークラウド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。コンテナーイメージを 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**: 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る環境ファイルを指定します。この例では、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。

6.5. コンテナーイメージ準備のパラメーター

コンテナー準備用のデフォルトファイル (**containers-prepare-parameter.yaml**) には、**ContainerImagePrepare** heat パラメーターが含まれます。このパラメーターで、イメージのセットを準備するためのさまざまな設定を定義します。

```
parameter_defaults:
  ContainerImagePrepare:
  - (strategy one)
  - (strategy two)
  - (strategy three)
  ...
```

それぞれの設定では、サブパラメーターのセットにより使用するイメージやイメージの使用方法を定義することができます。以下の表には、**ContainerImagePrepare** の各設定で使用することのできるサブパラメーターの情報をまとめています。

パラメーター	説明
excludes	設定からイメージ名を除外する正規表現の一覧
includes	設定に含める正規表現の一覧。少なくとも 1 つのイメージ名が既存のイメージと一致している必要があります。**includes** パラメーターを指定すると、**excludes** の設定はすべて無視されます。
modify_append_tag	対象となるイメージのタグに追加する文字列。たとえば、**14.0-89** のタグが付けられたイメージをプルし、**modify_append_tag** を **-hotfix** に設定すると、director は最終イメージを **14.0-89-hotfix** とタグ付けします。
modify_only_with_labels	変更するイメージを絞り込むイメージラベルのディクショナリー。イメージが定義したラベルと一致する場合には、director はそのイメージを変更プロセスに含めます。
modify_role	イメージのアップロード中 (ただし目的のレジストリーにプッシュする前) に実行する Ansible ロール名の文字列
modify_vars	**modify_role** に渡す変数のディクショナリー

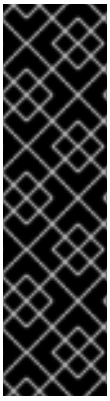
パラメーター	説明
push_destination	アップロードプロセス中にイメージをプッシュするレジストリーの名前空間を定義します。 ● **true** に設定すると、**push_destination** はホスト名を使用してアンダークラウドレジストリーの名前空間に設定されます。これが推奨される方法です。 ● **false** に設定するとローカルレジストリーへのプッシュは実行されず、ノードはソースから直接イメージをプルします。 ● カスタムの値に設定すると、director はイメージを外部のローカルレジストリーにプッシュします。 コンテナーイメージを Red Hat Container Catalog から直接プルする場合、実稼働環境ではこのパラメーターを **false** に設定しないでください。設定すると、すべてのオーバークラウドノードが同時に外部接続を通じて Red Hat Container Catalog からイメージをプルするため、帯域幅の問題が発生する可能性があります。**push_destination** パラメーターが **false** に設定されている (または定義されていない) 時にリモートレジストリーで認証が必要な場合は、**ContainerImageRegistryLogin** パラメーターを **true** に設定し、**ContainerImageRegistryCredentials** パラメーターで認証情報を含めます。
pull_source	元のコンテナーイメージをプルするソースレジストリー
set	初期イメージの取得場所を定義する、**key: value** 定義のディクショナリー
tag_from_label	指定したコンテナーイメージラベルの値を使用して、全イメージのバージョン付きタグを検出してプルします。director は **tag** に設定した値がタグ付けされた各コンテナーイメージを検査し、続いてコンテナーイメージラベルを使用して新しいタグを構築し、レジストリーからプルします。たとえば、**tag_from_label: {version}-{release}** を設定した場合、director は **version** および **release** ラベルを使用して新しいタグを構築します。あるコンテナーについて **version** を **13.0** に、**release** を **34** に設定した場合、タグは **13.0-34** となります。

重要

イメージをアンダークラウドにプッシュする場合は、**push_destination: UNDERCLOUD_IP:PORT** ではなく **push_destination: true** を使用します。**push_destination: true** を使用することで、IPv4 アドレスおよび IPv6 アドレスの両方で一貫性が確保されます。

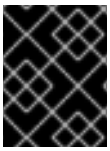
set パラメーターには、複数の **key: value** 定義を設定することができます。

キー	説明
ceph_image	Ceph Storage コンテナーイメージの名前
ceph_namespace	Ceph Storage コンテナーイメージの名前空間
ceph_tag	Ceph Storage コンテナーイメージのタグ
name_prefix	各 OpenStack サービスイメージの接頭辞
name_suffix	各 OpenStack サービスイメージの接尾辞
namespace	各 OpenStack サービスイメージの名前空間
neutron_driver	使用する OpenStack Networking (neutron) コンテナーを定義するのに使用するドライバー。標準の **neutron-server** コンテナーに設定するには、null 値を使用します。OVN ベースのコンテナーを使用するには、**ovn** に設定します。
tag	ソースからの全イメージに特定のタグを設定します。**tag_from_label** の値を指定せずこのオプションを使用した場合、director はこのタグを使用するすべてのコンテナーイメージをプルします。ただし、このオプションを **tag_from_label** の値と共に使用する場合は、director はソースイメージとして **tag** を使用して、ラベルに基づいて特定のバージョンタグを識別します。このキーは Red Hat OpenStack Platform のバージョン番号であるデフォルト値に設定したままにします。



重要

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



重要

コンテナーイメージでは、Red Hat OpenStack Platform のバージョンに基づいたマルチストリームタグが使用されます。したがって、今後 **latest** タグは使用されません。

ContainerImageRegistryCredentials パラメーターは、コンテナーレジストリーをそのレジストリーに対して認証を行うためのユーザー名とパスワードにマッピングします。

コンテナーレジストリーでユーザー名およびパスワードが必要な場合には、**ContainerImageRegistryCredentials** を使用して以下の構文で認証情報を設定することができます。

```
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。詳しくは、「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 パラメーターは、デプロイ中のシステムのレジストリーへのログインを制御するために使用されます。**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、これを **true** に設定する必要があります。

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

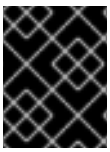
push_destination を設定している場合は、**ContainerImageRegistryLogin** を **true** に設定しないでください。このオプションを **true** に設定した場合、オーバークラウドノードに **ContainerImageRegistryCredentials** で定義されたレジストリーホストへのネットワーク接続がないと、ログインを試みる際にデプロイメントが失敗する可能性があります。

6.6. 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

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。

```
parameter_defaults:
  ContainerImagePrepare:
  - (strategy one)
  - (strategy two)
```

```
  - (strategy three)
  ContainerImageRegistryCredentials:
    registry.example.com:
      username: "p@55w0rd!"
```

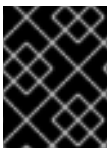


重要

プライベートレジストリーでは、**ContainerImagePrepare** の該当する設定について、**push_destination** を **true** に設定する必要があります。

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:
  ...
  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。

ContainerImageRegistryLogin パラメーターを使用して、コンテナーを取得するために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。

```
parameter_defaults:
  ...
  ContainerImageRegistryLogin: true
```

重要

特定の設定について、**push_destination** が設定されていない場合は、**ContainerImageRegistryLogin** を **true** に設定する必要があります。**ContainerImagePrepare** 設定で **push_destination** が設定され、**ContainerImageRegistryCredentials** パラメーターが設定されている場合、システムはログインしてコンテナーを取得し、それをリモートシステムにプッシュします。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、それぞれ設定します。

6.7. アップグレード用の移行コンテナーの取得

アップグレードには、以前のバージョンの Red Hat OpenStack Platform および Red Hat Ceph Storage のコンテナーが必要です。これらのコンテナーは、Red Hat OpenStack Platform 16.1 への移行に役立ちます。

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. **containers-prepare-parameter.yaml** ファイルを編集します。
3. 遷移コンテナーのパラメーターを **ContainerImagePrepare** パラメーターの **set** に追加します。設定するパラメーターは、director でデプロイされた Ceph Storage クラスターを使用するか、外部の Ceph Storage クラスターを使用するかによって異なります。
   - director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、以下のパラメーターを追加します。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ...
      name_prefix_stein: openstack-
      name_suffix_stein: ''
      namespace_stein: registry.redhat.io/rhosp15-rhel8
      tag_stein: 15.0
      ceph3_namespace: registry.redhat.io/rhceph
      ceph3_tag: latest
      ceph3_image: rhceph-3-rhel7
      ...
```

     - ***_stein** パラメーターで定義する Red Hat OpenStack Platform 15 用のコンテナーイメージが、アップグレードプロセスでデータベースの移行に使用されます。
     - **ceph3_*** パラメーターは、オーバークラウドが使用する現在の Red Hat Ceph Storage コンテナーイメージを定義します。Red Hat Ceph Storage 3 から 4 への移行には、オーバークラウドに **ceph3_*** と **ceph_*** 両方のパラメーターが必要です。
     - コンテナーイメージのストレージに Red Hat Satellite を使用している場合は、名前空間を Red Hat Satellite サーバー上のイメージの場所に設定します。
   - 外部の Ceph デプロイメントと共にアップグレードする場合は、以下のパラメーターを追加します。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ...
      name_prefix_stein: openstack-
      name_suffix_stein: ''
      namespace_stein: registry.redhat.io/rhosp15-rhel8
      tag_stein: 15.0
      ceph_namespace: registry.redhat.io/rhceph
```

```
    ceph_tag: latest
    ceph_image: rhceph-4-rhel8
    ...
```

- ***_stein** パラメーターで定義する Red Hat OpenStack Platform 15 用のコンテナーイメージが、アップグレードプロセスでデータベースの移行に使用されます。
- **ceph_*** パラメーターは、オーバークラウドが使用する現在の Red Hat Ceph Storage 4 コンテナーイメージを定義します。
- コンテナーイメージのストレージに Red Hat Satellite を使用している場合は、名前空間を Red Hat Satellite サーバー上のイメージの場所に設定します。

4. **neutron_driver** パラメーターを **openvswitch** に変更します。

```
parameter_defaults:
  ContainerImagePrepare:
  - push_destination: true
    set:
      ...
      neutron_driver: openvswitch
      ...
```

アップグレードプロセスの全期間中、Open vSwitch との互換性が維持されます。Red Hat OpenStack Platform 16.1 へのアップグレードが完了したら、オーバークラウドを Open vSwitch から Open Virtual Network (OVN) に移行します。

5. **containers-prepare-parameter.yaml** ファイルを保存します。

6.8. UNDERCLOUD.CONF ファイルの更新

引き続き Red Hat OpenStack Platform 13 環境からの元の **undercloud.conf** ファイルを使用できますが、Red Hat OpenStack Platform 16.1 との互換性を維持するために、ファイルを変更する必要があります。

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. **undercloud.conf** ファイルを編集します。
3. ファイル内の **DEFAULT** セクションに、以下のパラメーターを追加します。

```
container_images_file = /home/stack/containers-prepare-parameter.yaml
```

director が正しい場所からアンダークラウドのコンテナーイメージをプルできるように、このパラメーターで **containers-prepare-parameter.yaml** 環境ファイルの場所を定義します。

4. **generate_service_certificate** パラメーターを確認します。このパラメーターのデフォルト設定は、**false** から **true** に変更されます。これにより、アップグレード中にアンダークラウドで SSL/TLS が有効になります。
5. 予測可能な NIC 命名規則に移行している場合は、**local_interface** パラメーターを確認してください。

6. Red Hat OpenStack Platform 13 で **masquerade_network** パラメーターを設定している場合は、このパラメーターを削除して各サブネットに **masquerade = true** を設定します。

7. ファイル内の他のすべてのパラメータが変更されていないか確認します。

8. ファイルを保存します。

6.9. DIRECTOR の設定パラメーター

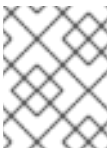
以下の一覧で、**undercloud.conf** ファイルを設定するパラメーターについて説明します。エラーを避けるために、パラメーターは決して該当するセクションから削除しないでください。

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一覧。現在、オーバークラウドは **ppc64le** アーキテクチャーをサポートしています。



注記

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

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

一時ファイルをクリーンナップします。デプロイメントコマンドの実行後もデプロイメント時に使用した一時ファイルをそのまま残すには、このパラメーターを **False** に設定します。ファイルを残すと、生成されたファイルのデバッグを行う場合やエラーが発生した場合に役に立ちます。

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 8.2 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一覧。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** の一覧にドライバーを含める必要があります。

enable_ironic、enable_ironic_inspector、enable_mistral、enable_nova、enable_tempest、enable_validations、enable_zaqar

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_pxe** ドライバーをデフォルトとして使用しますが、**discovery_default_driver** を設定して上書きすることもできます。また、イントロスペクションルールを使用して、新しく登録したノードにドライバーの情報を指定することもできます。

enable_novajoin

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

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。

enable_swift_encryption

保存データの Swift 暗号化を有効にするかどうかを定義します。

enable_telemetry

アンダークラウドに OpenStack Telemetry サービス (gnocchi、aodh、panko) をインストールするかどうかを定義します。Telemetry サービスを自動的にインストール/設定するには、**enable_telemetry** パラメーターを **true** に設定します。デフォルト値は **false** で、アンダークラウド上の telemetry が無効になります。このパラメーターは、メトリックデータを消費する Red Hat CloudForms などの他の製品を使用する場合に必要です。

enabled_hardware_types

アンダークラウドで有効にするハードウェアタイプの一覧

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。この機能の使用についての詳細は、「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。

ipa_otp

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

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一覧を以下に示します。

- 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。

local_interface

director のプロビジョニング NIC 用に選択するインターフェース。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: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen
1000
    link/ether 52:54:00:75:24:09 brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.178/24 brd 192.168.122.255 scope global dynamic eth0
       valid_lft 3462sec preferred_lft 3462sec
    inet6 fe80::5054:ff:fe75:2409/64 scope link
```

```
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noop state DOWN
    link/ether 42:0b:c2:a5:c1:26 brd ff:ff:ff:ff:ff:ff
```

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

local_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。デフォルトの 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。

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

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 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** で設定し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ースにオプションを追加する場合に、このパラメーターを使用します。**/usr/share/instack-undercloud/templates/net-config.json.template** の例を参照してください。

networks_file

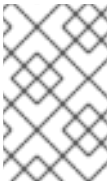
heat をオーバーライドするネットワークファイル

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一覧。デフォルト値に含まれるのは、**ctlplane-subnet** サブネットのみです。詳しくは、「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_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。設定すると、アンダークラウドのインストールで全システムのホスト名が設定されます。未設定のままにすると、アンダークラウドは現在のホスト名を使用しますが、システムのホスト名設定をすべて適切に定義する必要があります。

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 ネームサーバーの一覧

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 サーバーの一覧

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_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
```

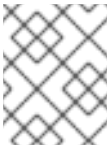
プロビジョニングネットワークは、環境に応じて、必要なだけ指定することができます。

cidr

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

masquerade

外部ネットワークへのアクセスのために、**cidr** で定義したネットワークをマスカレードするかどうかを定義します。このパラメーターにより、director 経由で外部ネットワークにアクセスすることができるように、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) の一部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。

dhcp_exclude

DHCP 割り当て範囲で除外する IP アドレス

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。

6.10. DIRECTOR のアップグレードの実施

director をアップグレードするには、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、アンダークラウド上の director をアップグレードします。

```
$ openstack undercloud upgrade
```

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 によりパッケージがアップグレードされ、**undercloud.conf** の設定に合わせてサービスが設定されます。このスクリプトは、完了までに数分かかります。

注記

director の設定スクリプトでは、次の設定に進む前に確認が求められます。この確認を省略するには、**-y** オプションを使用します。

```
$ openstack undercloud upgrade -y
```

2. このスクリプトにより、アンダークラウド上の OpenStack Platform サービスのコンテナーも、すべて自動的に起動されます。**systemd** リソースでそれぞれのサービスを管理します。**systemd** リソースを確認します。

```
$ sudo systemctl list-units "tripleo_*"
```

各 **systemd** サービスでコンテナーを制御します。以下のコマンドを使用して、有効化されたコンテナーを確認してください。

```
$ sudo podman ps
```

3. スクリプトにより **stack** ユーザーが **docker** グループに追加され、**stack** ユーザーがコンテナー管理コマンドを使用できるようになります。以下のコマンドで **stack** ユーザーのアクセス権限をリフレッシュします。

```
$ exec su -l stack
```

このコマンドを実行すると、再度ログインが求められます。stack ユーザーのパスワードを入力します。

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

```
$ 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) $
```

director のアップグレードが完了しました。

パート II. オーバークラウドアップグレードの準備

第7章 オーバークラウド準備の初期手順

オーバークラウドのアップグレードに向けた準備を行うには、いくつかの初期手順を完了する必要があります。

7.1. オーバークラウドサービスのダウンタイムの準備

オーバークラウドのアップグレードプロセスにより、重要なポイントで主要のサービスは無効化されます。このため、アップグレード中は、オーバークラウドサービスを使用して新規リソースを作成することはできません。アップグレード中は、オーバークラウドで実行中のワークロードはアクティブな状態のままなので、インスタンスはアップグレード中にも稼働し続けることになります。

アップグレード中にはユーザーがオーバークラウドのサービスにアクセスできないように、メンテナンスの時間帯を計画することが重要となります。

オーバークラウドのアップグレードによる影響を受ける項目

- OpenStack Platform サービス

オーバークラウドのアップグレードによる影響を受けない項目

- アップグレード中に実行するインスタンス
- Ceph Storage OSD (インスタンス向けのバックエンドストレージ)
- Linux ネットワーク
- Open vSwitch ネットワーク
- アンダークラウド

7.2. アップグレードテスト用のコンピュートノードの選択

オーバークラウドのアップグレードプロセスでは、次のいずれかを行うことができます。

- 1 つのロールのノードをすべてアップグレードする
- 個別のノードを別々にアップグレードする

オーバークラウドのアップグレードプロセスを円滑にするには、全コンピュートノードをアップグレードする前に、環境内にある個々のコンピュートノードのいくつかでアップグレードをテストすると役立ちます。これにより、アップグレード中に大きな問題が発生しなくなり、ワークロードのダウンタイムを最小限に抑えることができます。

アップグレードをテストするノードを選択するにあたっては、以下の推奨事項を参考にしてください。

- アップグレードのテストには、2 台または 3 台のコンピュートノードを選択します。
- クリティカルなインスタンスが実行されていないノードを選択します。
- 必要な場合には、選択したテスト用のコンピュートノードから他のコンピュートノードにクリティカルなインスタンスを移行します。

7.3. アップグレード前の要件の検証

pre-upgrade 検証グループを実行して、アップグレード前の要件を確認します。

手順

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

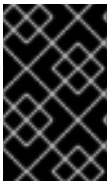
```
$ source ~/stackrc
```

2. **--group pre-upgrade** オプションを指定して **openstack tripleo validator run** コマンドを実行します。

```
$ openstack tripleo validator run --group pre-upgrade
```

3. 検証レポートの結果を確認します。特定の検証からの詳細出力を表示するには、レポートからの特定検証の UUID を指定して **openstack tripleo validator show run** コマンドを実行します。

```
$ openstack tripleo validator show run <UUID>
```



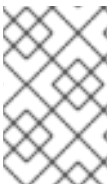
重要

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。

7.4. オーバークラウドでのフェンシングの無効化

オーバークラウドをアップグレードする場合、高可用性機能を維持するために、各コントローラーノードを個別にアップグレードします。このアップグレード期間中、オーバークラウドは特定ノードが無効であることを検出し、フェンシング操作を試みる場合があります。これにより、意図しない動作が生じる可能性があります。

オーバークラウドでフェンシングを有効にしている場合には、意図しない動作を防ぐために、アップグレード期間中フェンシングを一時的に無効にする必要があります。



注記

環境ファイルを使用してオーバークラウドのフェンシングを有効にしている場合には、director はコントローラーノードクラスターの再作成時に再度フェンシングを有効にします。

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

```
$ source ~/stackrc
```

3. コントローラーノードにログインし、Pacemaker コマンドを実行してフェンシングを無効にします。

```
$ ssh heat-admin@CONTROLLER_IP "sudo pcs property set stonith-enabled=false"
```

-

関連資料

- 「STONITH を使用したコントローラノードのフェンシング」

7.5. オーバークラウドインベントリーファイルの作成

tripleo-ansible-inventory コマンドを使用して、環境内の全ノードの Ansible インベントリー ファイルを生成します。

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

```
$ source ~/stackrc
```

3. 全ノードの静的なインベントリーファイルを作成します。

```
$ tripleo-ansible-inventory --static-yaml-inventory ~/inventory.yaml --stack STACK_NAME
```

デフォルトのスタック名 **overcloud** を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

4. お使いの環境で Ansible のプレイブックを実行するには、**ansible-playbook** コマンドを実行し、**-i** オプションを使用して動的インベントリーツールの完全パスを追加します。以下に例を示します。

```
(undercloud) $ ansible-playbook -i ~/inventory.yaml PLAYBOOK
```

第8章 LEAPP アップグレードのためのオーバークラウド設定

オーバークラウドのアップグレードに向けた準備を行うには、いくつかの Leapp 設定を完了する必要があります。

8.1. アップグレード環境ファイルの作成

アップグレードプロセスでは、環境ファイルを使用して、アップグレードプロセスを有効にして特定のアップグレードパラメーターを設定します。

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. **templates** ディレクトリーに **upgrades-environment.yaml** という名前の環境ファイルを作成します。

```
$ touch templates/upgrades-environment.yaml
```

3. ファイルを編集し、以下の必須コンテンツを追加します。

```
parameter_defaults:
  UpgradeInitCommand: |
    sudo dnf module disable -y container-tools:rhel8
    sudo dnf module enable -y container-tools:2.0
    sudo dnf module disable -y virt:rhel
    sudo dnf module enable -y virt:8.2
  UpgradeLeappCommandOptions: "--enablerepo rhel-8-for-x86_64-baseos-eus-rpms --enablerepo rhel-8-for-x86_64-appstream-eus-rpms --enablerepo fast-datapath-for-rhel-8-x86_64-rpms"
```

- **UpgradeInitCommand** は、ノードでアップグレードを開始する前に director が実行するバッシュコマンドのセットを設定します。この場合、デフォルトの **container-tools:rhel8** モジュールから **container-tools:2.0** に切り替える **dnf** コマンドを使用するように、パラメーターを設定します。
- **UpgradeLeappCommandOptions** は、Leapp ツールに渡す追加のオプションを設定します。この場合、Leapp による移行に役立つ Red Hat OpenStack Platform リポジトリーを有効にするように、パラメーターを設定します。

4. **upgrades-environment.yaml** ファイルを保存します。

8.2. アップグレードのパラメーター

パラメーター	説明
UpgradeInitCommand	アップグレードプロセスを初期化するためにすべてのオーバークラウドノード上で実行するコマンドまたはスクリプトのスニペット。たとえば、リポジトリーの切り替えなど。

パラメーター	説明
UpgradeInitCommonCommand	アップグレードプロセスに必要な共通のコマンド。操作者は通常このパラメーターを変更する必要はなく、major-upgrade-composable-steps.yaml および major-upgrade-converge.yaml 環境ファイルで設定および設定解除されます。
UpgradeLeappCommandOptions	Leapp コマンドに追加するその他のコマンドラインオプション
UpgradeLeappDebug	Leapp の実行中にデバッグのアウトプットを出力します。デフォルト値は **True** です。
UpgradeLeappDevelSkip	開発/テスト環境で Leapp を実行する場合は、環境変数を設定して Leapp の確認を省略します。たとえば、LEAPP_DEVEL_SKIP_RHSM=1 と設定します。
UpgradeLeappEnabled	オペレーティングシステムのアップグレードに Leapp を使用します。デフォルト値は **False** です。
UpgradeLeappRebootTimeout	Leapp による OS アップグレードフェーズのタイムアウト時間 (秒単位)。デフォルト値は **3600** です。
UpgradeLeappToInstall	Leapp によるアップグレード後にインストールするパッケージの一覧
UpgradeLeappToRemove	Leapp によるアップグレード時に削除するパッケージの一覧

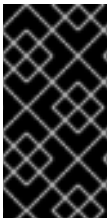
8.3. オーバークラウドノードへの LEAPP データのコピー

それぞれのオーバークラウドノードには、Leapp データファイルが必要です。director は、アンダークラウドの **/etc/leapp/files** ディレクトリーにあるデータファイルを、各オーバークラウドの同じ場所に自動的にコピーします。**pes-events.json** および **repomap.csv** ファイルがアンダークラウドの **/etc/leapp/files** ディレクトリーにまだ保存されていることを確認します。

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. アンダークラウドの **/etc/leapp/files** ディレクトリーを確認します。

```
$ sudo ls /etc/leapp/files/
```



重要

これらのファイルが削除されている場合は、ナレッジベースのアーティクル「RHEL 7 から RHEL 8 へのインプレースアップグレード時に Leapp ユーティリティーで必要なデータ」に添付されているファイルをダウンロードし、それらのファイルをアンダークラウドの **/etc/leapp/files/** ディレクトリーに保存します。

8.4. オーバークラウドノードでの予測可能な NIC 名の使用

オーバークラウドノードで Leapp によるアップグレードを実施する前に、カーネルベースの NIC 名が使用されているかどうかを確認する必要があります。この NIC 名には、通常 **eth** の接頭辞が含まれます。NIC の割り当てに関して、通常これらの NIC 名は予測可能ではありません。ノード上のいずれかの NIC で **eth** の接頭辞が使用されていると、Leapp によるアップグレードに失敗します。

前提条件

- アンダークラウドの準備プロセス中に作成された **playbook-nics.yaml** Playbook。**playbook-nics.yaml** Playbook は、イーサネットデバイス、ブリッジ、および Linux ボンディングを使用するほとんどのオーバークラウドネットワーク設定のシナリオに対応します。お使いの環境でこれらのデバイス種別以外の追加設定が必要な場合は、手順を進める前に以下の推奨事項に従ってください。
  - 実際のオーバークラウドノードと類似したネットワーク設定の別システムで Playbook をテストする。
  - Playbook を変更し、他のデバイス種別の設定内の **eth** プレフィックスの名称変更に対応する。
  - 以下の手順を完了したら、オーバークラウドノードのネットワーク設定を確認する。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. すべてのオーバークラウドノードで **playbook-nics.yaml** Playbook を実行します。

```
$ ansible-playbook -i ~/inventory.yaml playbook-nics.yaml
```

この Playbook により、新しい NIC の接頭辞が **em** に設定されます。別の NIC 接頭辞を設定するには、Playbook の実行時に **prefix** 変数を設定します。

```
$ ansible-playbook -i ~/inventory.yaml -e prefix="mynic" playbook-nics.yaml
```

3. 標準のリブート手順を使用して、オーバークラウドノードをリブートします。詳しくは、「ノードのリブート」を参照してください。

リソース

- 「RHEL 7 でカーネルの NIC 名を使用している場合に RHEL 8 へのインプレースアップグレードを実行する方法」
- 『ネットワークガイド』の「命名スキームの序列」
- 『ネットワークガイド』の「予想可能なネットワークインターフェースデバイスの命名について」

8.5. オーバークラウドでの SSH ROOT 権限パラメーターの設定

Leapp によるアップグレードでは、**PermitRootLogin** パラメーターが各ノードの **/etc/ssh/sshd_config** ファイルに存在するかどうかを確認します。このパラメーターを、明示的に **yes** または **no** のいずれかに設定する必要があります。

セキュリティー上の理由から、オーバークラウドノードで root ユーザーへの SSH アクセスを無効にするには、このパラメーターを **no** に設定します。

前提条件

- オーバークラウドノードの Ansible インベントリーファイル

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. **playbook-ssh.yaml** という名前のファイルを作成し、以下のコンテンツをファイルに貼り付けます。

```
---
- name: Configure SSH PermitRootLogin parameter
  hosts: overcloud
  become: yes
  tasks:
    - name: Set the PermitRootLogin parameter to no
      lineinfile:
        path: /etc/ssh/sshd_config
        state: present
        line: "PermitRootLogin no"
```

3. Playbook を実行します。

```
$ ansible-playbook -i ~/inventory.yaml playbook-ssh.yaml
```

第9章 コンポーザブルサービスおよびパラメーターの更新

オーバークラウドのアップグレードに向けた準備を行うには、いくつかのコンポーザブルサービス設定を完了する必要があります。

9.1. カスタム ROLES_DATA ファイルのコンポーザブルサービスの更新

本項では、新しいコンポーザブルサービスおよび非推奨のコンポーザブルサービスについて説明します。

- デフォルトの **roles_data** ファイルを使用する場合は、これらのサービスが自動的に含まれます。
- カスタムの **roles_data** ファイルを使用する場合は、該当するロールごとに新しいサービスを追加し非推奨のサービスを削除します。

コントローラーノード

コントローラーノードでは、以下のサービスが非推奨になっています。Controller ロールからこれらのサービスを削除します。

サービス	理由
OS::TripleO::Services::AodhApi **OS::TripleO::Services::AodhEvaluator** **OS::TripleO::Services::AodhListener** **OS::TripleO::Services::AodhNotifier** **OS::TripleO::Services::CeilometerApi** **OS::TripleO::Services::CeilometerCollector** **OS::TripleO::Services::CeilometerExpirer** **OS::TripleO::Services::MongoDb** **OS::TripleO::Services::PankoApi**	OpenStack Platform には OpenStack Telemetry サービスが含まれなくなり、メトリクスおよびモニタリングに Service Telemetry Framework (STF) が使用されるようになったため。
OS::TripleO::Services::Congress	Congress はサポートされなくなったため。
OS::TripleO::Services::GlanceRegistry	OpenStack Platform Image サービス (glance) API v2 により、このサービスはサポートされなくなったため。
OS::TripleO::Services::NeutronCorePluginPlumgrid **OS::TripleO::Services::NeutronCorePluginMidonet**	これらの OpenStack Networking (neutron) 用プラグインは非推奨になったため。

サービス	理由
OS::TripleO::Services::NeutronLbaasv2Agent **OS::TripleO::Services::NeutronLbaasv2Api**	Octavia を優先し、OpenStack Networking (neutron) Load Balancing as a Service は非推奨になったため。
OS::TripleO::Services::NovaConsoleauth	このサービスは非推奨になったため。
OS::TripleO::Services::NovaPlacement	**OS::TripleO::Services::PlacementApi** を優先し、非推奨になったため。
OS::TripleO::Services::OpenDaylightApi **OS::TripleO::Services::OpenDaylightOvs**	OpenDaylight はサポートされなくなったため。
OS::TripleO::Services::RabbitMQ	このサービスは、2 つの新規サービスに置き換えられています。 **OS::TripleO::Services::OsloMessagingRpc** **OS::TripleO::Services::OsloMessagingNotify**
OS::TripleO::Services::SkydiveAgent **OS::TripleO::Services::SkydiveAnalyzer**	Skydive はサポートされなくなったため。
OS::TripleO::Services::Tacker	Tacker はサポートされなくなったため。

コントローラーノードの新規サービスは以下のとおりです。Controller ロールにこれらのサービスを追加します。

サービス	理由
OS::TripleO::Services::CephGrafana	Ceph Dashboard サービスを有効にするタスク
OS::TripleO::Services::CinderBackendDellEMCPowermax **OS::TripleO::Services::CinderBackendDellEMCSc** **OS::TripleO::Services::CinderBackendNVMeOF**	Block Storage (cinder) の新規バックエンド
OS::TripleO::Services::ContainerImagePrepare	コマンドを実行して、オーバークラウドのサービスに関連するコンテナーイメージを自動的にプルして準備します。

サービス	理由
OS::TripleO::Services::DesignateApi **OS::TripleO::Services::DesignateCentral** **OS::TripleO::Services::DesignateProducer** **OS::TripleO::Services::DesignateWorker** **OS::TripleO::Services::DesignateMDNS** **OS::TripleO::Services::DesignateSink**	DNS-as-a-Service 用サービス (designate)
OS::TripleO::Services::IronicInspector	オーバークラウドのベアメタルイントロスペクション用サービス
OS::TripleO::Services::IronicNeutronAgent	OpenStack Bare Metal (ironic) 用ネットワークエージェント
OS::TripleO::Services::NeutronAgentsIBConfig	Mellanox InfiniBand OpenStack Networking (neutron) エージェント用サービス
OS::TripleO::Services::OpenStackClients	Red Hat OpenStack Platform コマンドラインツールをインストールするためのサービス
OS::TripleO::Services::OsloMessagingRpc **OS::TripleO::Services::OsloMessagingNotify**	**OS::TripleO::Services::RabbitMQ** の代替サービス
OS::TripleO::Services::PlacementApi	Placement API 用サービス

コンピュートノード

コンピュートノードでは、以下のサービスが非推奨になっています。Compute ロールからこれらのサービスを削除します。

サービス	理由
OS::TripleO::Services::OpenDaylightOvs	OpenDaylight はサポートされなくなったため。
OS::TripleO::Services::SkydiveAgent	Skydive はサポートされなくなったため。

コンピュートノードの新規サービスは以下のとおりです。Compute ロールにこれらのサービスを追加します。

サービス	理由

サービス	理由
OS::TripleO::Services::NovaAZConfig	OpenStack Compute (nova) でホストアグリゲートおよびアベイラビリティーゾーンを設定するためのサービス

全ノード

すべてのノードで、以下のサービスが非推奨になっています。すべてのロールからこれらのサービスを削除します。

サービス	理由
OS::TripleO::Services::Docker	Podman に置き換わったため。
OS::TripleO::Services::Fluentd	**OS::TripleO::Services::Rsyslog** を優先し、非推奨になったため。
OS::TripleO::Services::Ntp	**OS::TripleO::Services::Timesync** を優先し、非推奨になったため。
OS::TripleO::Services::SensuClient	このサービスは非推奨になったため。
OS::TripleO::Services::Ptp	**OS::TripleO::Services::Timesync** を優先し、非推奨になったため。

全ノードでの新規サービスは以下のとおりです。すべてのロールにこれらのサービスを追加します。

サービス	理由
OS::TripleO::Services::BootParams	カーネル引数、Tuned プロファイル、および CPU 分離を設定するためのサービス
OS::TripleO::Services::Collectd	Collectd を設定するためのサービス
OS::TripleO::Services::Multipathd	デフォルトで無効化されている Multipathd サービスを提供します。
OS::TripleO::Services::Podman	Podman をインストールし有効にするためのサービス
OS::TripleO::Services::Rear	Relax-and-Recover (ReaR) バックアップおよびリストアツールをインストールおよび有効にするためのサービス
OS::TripleO::Services::Rsyslog	集中ログコレクションを設定するためのサービス

サービス	理由
OS::TripleO::Services::Timesync	時刻同期方法 (デフォルトでは Chronyd) を有効にするためのサービス

9.2. カスタム環境ファイルのコンポーザブルサービスの更新

resource_registry セクションのあるカスタム環境ファイルを使用している場合は、**resource_registry** セクションでコンポーザブルサービステンプレートのマッピングに変更がないか確認してください。Red Hat OpenStack Platform 16.1 では、コンポーザブルサービスのファイルは **/usr/share/openstack-tripleo-heat-templates/** 内の新しい場所にあります。

Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 16.1
docker/services/	**deployment**

deployment ディレクトリーには、コンポーザブルサービスをグループ化するためのサブディレクトリーセットが含まれます。たとえば、**keystone** サブディレクトリーには、OpenStack Identity (keystone) のコンポーザブルサービステンプレートが含まれます。

カスタム環境ファイルのコンポーザブルサービスを再マッピングするには、現在のサービスマッピングのテンプレートの場所を確認し、マッピングを新しい場所に編集します。以下の手順では、例として **ceph-mgr.yaml** を使用しています。

重要

以下の手順は、コンポーザブルサービスを再マッピングする際のガイドとしてのみ提供されます。いずれかのマッピングが不明であれば、Red Hat に連絡して正しいマッピングについてアドバイスを求めてください。

手順

1. コンポーザブルサービスを使用するカスタム環境ファイルを検索します。通常、カスタム環境ファイルは **/home/stack/templates** ディレクトリーに保存されます。

```
$ cd ~/templates/
$ grep "OS::TripleO::Services" *
```

このシナリオでは、ファイルの 1 つに古くなったマッピングがあるとします。

```
  OS::TripleO::Services::CephMgr: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-mgr.yaml
```

2. **/usr/share/openstack-tripleo-heat-templates/** で新しい **ceph-mgr.yaml** の場所を特定します。このファイルは、**deployment/ceph-ansible** ディレクトリーに置かれています。

```
$ find /usr/share/openstack-tripleo-heat-templates/ -name ceph-mgr.yaml
/usr/share/openstack-tripleo-heat-templates/deployment/ceph-ansible/ceph-mgr.yaml
```

3. カスタム環境ファイルでサービスを編集します。

```
resource_registry:
  OS::TripleO::Services::CephMgr: /usr/share/openstack-tripleo-heat-
templates/deployment/ceph-ansible/ceph-mgr.yaml
```

ファイルを保存します。

9.3. アンダークラウドレジストリーへのアクセスの設定

アンダークラウドのコントロールプレーンのホスト名を、**DockerInsecureRegistryAddress** パラメーターに追加します。このパラメーターを **containers-prepare-parameter.yaml** ファイルに置き、パラメーターがこれ以降のオーバークラウドデプロイメントに含まれるようにします。

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. アンダークラウドのコントロールプレーンのホスト名を取得します。

```
$ sudo hiera container_image_prepare_node_names
["undercloud.ctlplane.localdomain"]
```

3. **containers-prepare-parameter.yaml** ファイルを編集し、**DockerInsecureRegistryAddress** パラメーターを追加してアンダークラウドのコントロールプレーンのホスト名が含まれる YAML の一覧を指定します。

```
parameter_defaults:
  DockerInsecureRegistryAddress:
  - undercloud.ctlplane.localdomain:8787
  ...
```

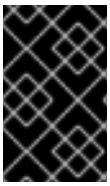
ホスト名の値に、オーバークラウドレジストリーのポート番号も追加する必要があります。ポート番号は **8787** です。

9.4. 非推奨化および廃止された NOVASCHEDULERDEFAULTFILTERS パラメーターのフィルター

お使いの環境でカスタムの **NovaSchedulerDefaultFilters** パラメーターが使用されている場合は、パラメーターを編集して以下に示す非推奨化および廃止されたフィルターを削除します。

フィルター	ステータス
AggregateCoreFilter	非推奨
AggregateRamFilter	非推奨
AggregateDiskFilter	非推奨
ExactCoreFilter	廃止
ExactRamFilter	廃止

フィルター	ステータス
ExactDiskFilter	廃止
CoreFilter	廃止
RamFilter	廃止
DiskFilter	廃止
RetryFilter	非推奨



重要

非推奨 と識別されたフィルターを使用しないでください。Red Hat OpenStack Platform 16.1 には引き続き非推奨のフィルターが含まれていますが、Red Hat OpenStack Platform の今後のバージョンにはこれらのフィルターは含まれません。

9.5. COMPUTE 名の形式の設定

Red Hat OpenStack Platform 13 では、コンピュートノードのデフォルト命名形式として **%stackname%-compute-%index%** が使用されます。Red Hat OpenStack Platform 16.1 では、コンピュートノードのデフォルト命名形式として **%stackname%-novacompute-%index%** が使用されます。デフォルトの命名形式を変更して、元の Red Hat OpenStack Platform 13 の命名形式を維持します。元の命名形式を使用しない場合には、director は新しい命名形式の新たな OpenStack Compute (nova) エージェントを設定し、古い命名形式の既存の OpenStack Compute (nova) エージェントを孤立サービスとして維持します。

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. Compute の命名形式を設定します。
   - カスタムの **roles_data** ファイルを使用する場合は、カスタム **roles_data** ファイルを編集して **Compute** ロールの **HostnameFormatDefault** パラメーターを設定します。

     ```
     - name: Compute
       …
       HostnameFormatDefault: '%stackname%-compute-%index%'
       …
     ```

     カスタムの **roles_data** ファイルを保存します。
   - **openstack-tripleo-heat-templates** のデフォルト **roles_data** ファイルを使用する場合は、環境ファイルで命名形式を設定します。実際のノード数およびフレーバーに合わせて環境ファイルを編集します。このファイルは、通常 **node-info.yaml** という名前です。**parameter_defaults** セクションに **ComputeHostnameFormat** パラメーターを追加します。

     ```
     parameter_defaults:
     ```

```
…
ComputeHostnameFormat: '%stackname%-compute-%index%'
…
```

node-info.yaml ファイルを保存します。

9.6. SSL/TLS 設定の更新

resource_registry から **NodeTLSData** リソースを削除して、SSL/TLS 設定を更新します。

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

```
$ source ~/stackrc
```

3. カスタムのオーバークラウド SSL/TLS パブリックエンドポイントファイルを編集します。このファイルは、通常 **~/templates/enable-tls.yaml** という名前です。
4. resource_registry から **NodeTLSData** リソースを削除します。

```
resource_registry:
  OS::TripleO::NodeTLSData: /usr/share/openstack-tripleo-heat-
templates/puppet/extraconfig/tls/tls-cert-inject.yaml
  …
```

オーバークラウドデプロイメントは、HAProxy の新しいサービスを使用して SSL/TLS が有効かどうかを判断します。

注記

enable-tls.yaml ファイルの **resource_registry** セクションのリソースがこれだけであれば、**resource_registry** セクション全体を削除します。

5. SSL/TLS パブリックエンドポイントファイルを保存します。

9.7. 設定後テンプレートの更新

登録に **OS::TripleO::NodeExtraConfigPost** リソースを使用し、設定後テンプレートを実行する場合は、テンプレートに **EndpointMap** パラメーターを追加する必要があります。

手順

1. 設定後テンプレートを編集します。
2. **parameters** セクションに **EndpointMap** パラメーターおよびそのサブパラメーターを追加します。

```
parameters:
  servers:
    type: json
```

```
    nameserver_ip:
      type: string
    DeployIdentifier:
      type: string
    EndpointMap:
      default: {}
      type: json
```

3. テンプレートを保存します。

第10章 RED HAT カスタマーポータルへのオーバークラウド登録の更新

Red Hat OpenStack Platform 16.1 では、Ansible ベースの手法を使用してオーバークラウドノードを Red Hat カスタマーポータルに登録します。

10.1. RED HAT SUBSCRIPTION MANAGER (RHSM) コンポーザブルサービス

rhsm コンポーザブルサービスは、Ansible を介してオーバークラウドノードを登録する方法を提供します。デフォルトの **roles_data** ファイルの各ロールには、**OS::TripleO::Services::Rhsm** リソースが含まれており、これはデフォルトで無効になっています。サービスを有効にするには、このリソースを **rhsm** コンポーザブルサービスのファイルに登録します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
```

rhsm コンポーザブルサービスは **RhsmVars** パラメーターを受け入れます。これにより、登録に必要な複数のサブパラメーターを定義することができます。以下に例を示します。

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - rhel-8-for-x86_64-baseos-eus-rpms
      - rhel-8-for-x86_64-appstream-eus-rpms
      - rhel-8-for-x86_64-highavailability-eus-rpms
      …
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_release: 8.2
```

RhsmVars パラメーターをロール固有のパラメーター (例: **ControllerParameters**) と共に使用することにより、異なるノードタイプ用の特定のリポジトリーを有効化する場合に柔軟性を提供することもできます。

10.2. RHSMVARS サブパラメーター

すべての Ansible パラメーターを把握するには、ロールに関するドキュメント を参照してください。

rhsm	説明
rhsm_method	登録の方法を選択します。**portal**、**satellite**、または **disable** のいずれかです。
rhsm_org_id	登録に使用する組織。この ID を特定するには、アンダークラウドノードから **sudo subscription-manager orgs** を実行します。プロンプトが表示されたら、Red Hat の認証情報を入力して、出力される **Key** 値を使用します。組織の ID についての詳細は、「Red Hat Subscription Management における組織 ID について理解する」を参照してください。

rhsm	説明
rhsm_pool_ids	使用するサブスクリプションプール ID。サブスクリプションを自動でアタッチしない場合には、このパラメーターを使用してください。この ID を特定するには、アンダークラウドノードから **sudo subscription-manager list --available --all --matches="OpenStack"** を実行し、出力される **Pool ID** の値を使用します。
rhsm_activation_key	登録に使用するアクティベーションキー。**rhsm_repos** が設定されている場合には機能しません。
rhsm_autosubscribe	互換性のあるサブスクリプションをこのシステムに自動的にアタッチします。有効にするには、**true** に設定します。
rhsm_baseurl	コンテンツを取得するためのベース URL。デフォルトは Red Hat コンテンツ配信ネットワークの URL です。Satellite サーバーを使用している場合は、この値を Satellite サーバーコンテンツリポジトリーのベース URL に変更します。
rhsm_server_hostname	登録用のサブスクリプション管理サービスのホスト名。デフォルトは Red Hat Subscription Management のホスト名です。Satellite サーバーを使用している場合は、この値を Satellite サーバーのホスト名に変更します。
rhsm_repos	有効にするリポジトリーの一覧。**rhsm_activation_key** が設定されている場合には機能しません。
rhsm_username	登録用のユーザー名。可能な場合には、登録にアクティベーションキーを使用します。
rhsm_password	登録用のパスワード。可能な場合には、登録にアクティベーションキーを使用します。
rhsm_release	リポジトリー固定用の Red Hat Enterprise Linux リリース。Red Hat OpenStack Platform 16.1 の場合、これは **8.2** に設定されます。
rhsm_rhsm_proxy_hostname	HTTP プロキシーのホスト名。たとえば、**proxy.example.com**。
rhsm_rhsm_proxy_port	HTTP プロキシー通信用のポート。たとえば、**8080**。
rhsm_rhsm_proxy_user	HTTP プロキシーにアクセスするためのユーザー名
rhsm_rhsm_proxy_password	HTTP プロキシーにアクセスするためのパスワード

10.3. RHSM コンポーザブルサービスへの切り替え

従来の **rhel-registration** メソッドは、bash スクリプトを実行してオーバークラウドの登録を処理します。この方法のスクリプトと環境ファイルは、**/usr/share/openstack-tripleo-heat-**

templates/extraconfig/pre_deploy/rhel-registration/ のコア Heat テンプレートコレクションにあります。

rhel-registration メソッドを **rhsm** コンポーザブルサービスに切り替えるには、以下の手順を実施します。

手順

1. **rhel-registration** 環境ファイルは、今後のデプロイメント操作から除外します。通常は、以下のファイルを除外します。
   - **rhel-registration/environment-rhel-registration.yaml**
   - **rhel-registration/rhel-registration-resource-registry.yaml**
2. カスタムの **roles_data** ファイルを使用する場合には、**roles_data** ファイルの各ロールに必ず **OS::TripleO::Services::Rhsm** コンポーザブルサービスを含めてください。以下に例を示します。

```
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  ...
  ServicesDefault:
    ...
    - OS::TripleO::Services::Rhsm
    ...
```

3. **rhsm** コンポーザブルサービスのパラメーター用の環境ファイルを今後のデプロイメント操作に追加します。

このメソッドは、**rhel-registration** パラメーターを **rhsm** サービスのパラメーターに置き換えて、サービスを有効化する Heat リソースを変更します。

```
resource_registry:
  OS::TripleO::NodeExtraConfig: rhel-registration.yaml
```

上記の行を以下のように変更します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
```

デプロイメントに **/usr/share/openstack-tripleo-heat-templates/environments/rhsm.yaml** 環境ファイルを追加して、サービスを有効にすることもできます。

10.4. RHEL-REGISTRATION から RHSM へのマッピング

rhel-registration	rhsm / RhsmVars
rhel_reg_method	**rhsm_method**

rhel-registration	rhsm / RhsmVars
rhel_reg_org	**rhsm_org_id**
rhel_reg_pool_id	**rhsm_pool_ids**
rhel_reg_activation_key	**rhsm_activation_key**
rhel_reg_auto_attach	**rhsm_autosubscribe**
rhel_reg_sat_url	**rhsm_satellite_url**
rhel_reg_repos	**rhsm_repos**
rhel_reg_user	**rhsm_username**
rhel_reg_password	**rhsm_password**
rhel_reg_release	**rhsm_release**
rhel_reg_http_proxy_host	**rhsm_rhsm_proxy_hostname**
rhel_reg_http_proxy_port	**rhsm_rhsm_proxy_port**
rhel_reg_http_proxy_username	**rhsm_rhsm_proxy_user**
rhel_reg_http_proxy_password	**rhsm_rhsm_proxy_password**

10.5. RHSM コンポーザブルサービスを使用したオーバークラウドの登録

以下の手順に従って、**rhsm** コンポーザブルサービスを有効化して設定する環境ファイルを作成します。director はこの環境ファイルを使用して、ノードを登録し、サブスクライブします。

手順

1. 設定を保存するための環境ファイル (**templates/rhsm.yaml**) を作成します。
2. 環境ファイルに設定を追加します。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-
templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - rhel-8-for-x86_64-baseos-eus-rpms
      - rhel-8-for-x86_64-appstream-eus-rpms
      - rhel-8-for-x86_64-highavailability-eus-rpms
      …
```

```
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_pool_ids: "1a85f9223e3d5e43013e3d6e8ff506fd"
    rhsm_method: "portal"
    rhsm_release: 8.2
```

- **resource_registry** セクションは、各ロールで利用可能な **OS::TripleO::Services::Rhsm** リソースに **rhsm** コンポーザブルサービスを関連付けます。
- **RhsmVars** の変数は、Red Hat の登録を設定するためにパラメーターを Ansible に渡します。

3. 環境ファイルを保存します。

10.6. 異なるロールに対する RHSM コンポーザブルサービスの適用

rhsm コンポーザブルサービスをロールごとに適用することができます。たとえば、コントローラーノード、コンピュートノード、および Ceph Storage ノードに、異なる設定セットを適用することができます。

手順

1. 設定を保存するための環境ファイル (**templates/rhsm.yaml**) を作成します。
2. 環境ファイルに設定を追加します。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-
templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
parameter_defaults:
  ControllerParameters:
    RhsmVars:
      rhsm_repos:
        - rhel-8-for-x86_64-baseos-eus-rpms
        - rhel-8-for-x86_64-appstream-eus-rpms
        - rhel-8-for-x86_64-highavailability-eus-rpms
        - ansible-2.9-for-rhel-8-x86_64-rpms
        - advanced-virt-for-rhel-8-x86_64-rpms
        - openstack-16.1-for-rhel-8-x86_64-rpms
        - rhceph-4-mon-for-rhel-8-x86_64-rpms
        - rhceph-4-tools-for-rhel-8-x86_64-rpms
        - fast-datapath-for-rhel-8-x86_64-rpms
      rhsm_username: "myusername"
      rhsm_password: "p@55w0rd!"
      rhsm_org_id: "1234567"
      rhsm_pool_ids: "55d251f1490556f3e75aa37e89e10ce5"
      rhsm_method: "portal"
      rhsm_release: 8.2
  ComputeParameters:
    RhsmVars:
      rhsm_repos:
        - rhel-8-for-x86_64-baseos-eus-rpms
        - rhel-8-for-x86_64-appstream-eus-rpms
        - rhel-8-for-x86_64-highavailability-eus-rpms
```

```
      - ansible-2.9-for-rhel-8-x86_64-rpms
      - advanced-virt-for-rhel-8-x86_64-rpms
      - openstack-16.1-for-rhel-8-x86_64-rpms
      - rhceph-4-tools-for-rhel-8-x86_64-rpms
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_pool_ids: "55d251f1490556f3e75aa37e89e10ce5"
    rhsm_method: "portal"
    rhsm_release: 8.2
CephStorageParameters:
  RhsmVars:
    rhsm_repos:
      - rhel-8-for-x86_64-baseos-rpms
      - rhel-8-for-x86_64-appstream-rpms
      - rhel-8-for-x86_64-highavailability-rpms
      - ansible-2.9-for-rhel-8-x86_64-rpms
      - openstack-16.1-deployment-tools-for-rhel-8-x86_64-rpms
      - rhceph-4-osd-for-rhel-8-x86_64-rpms
      - rhceph-4-tools-for-rhel-8-x86_64-rpms
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_pool_ids: "68790a7aa2dc9dc50a9bc39fabc55e0d"
    rhsm_method: "portal"
    rhsm_release: 8.2
```

resource_registry は、各ロールで利用可能な **OS::TripleO::Services::Rhsm** リソースに **rhsm** コンポーザブルサービスを関連付けます。

ControllerParameters、**ComputeParameters**、および **CephStorageParameters** は、独自の **RhsmVars** パラメーターを使用してそれぞれのロールにサブスクリプション情報を渡します。

注記

Red Hat Ceph Storage のサブスクリプションおよび Ceph Storage 固有のリポジトリーを使用するように、**CephStorageParameters** パラメーター内の **RhsmVars** パラメーターを設定します。**rhsm_repos** パラメーターに、コントローラーノードおよびコンピュートノードに必要な Extended Update Support (EUS) リポジトリーではなく、標準の Red Hat Enterprise Linux リポジトリーが含まれるようにします。

3. 環境ファイルを保存します。

第11章 RED HAT SATELLITE へのオーバークラウド登録の更新

Red Hat OpenStack Platform 16.1 では、Ansible ベースの手法を使用してオーバークラウドノードを Red Hat Satellite 6 サーバーに登録します。

11.1. RED HAT SUBSCRIPTION MANAGER (RHSM) コンポーザブルサービス

rhsm コンポーザブルサービスは、Ansible を介してオーバークラウドノードを登録する方法を提供します。デフォルトの **roles_data** ファイルの各ロールには、**OS::TripleO::Services::Rhsm** リソースが含まれており、これはデフォルトで無効になっています。サービスを有効にするには、このリソースを **rhsm** コンポーザブルサービスのファイルに登録します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
```

rhsm コンポーザブルサービスは **RhsmVars** パラメーターを受け入れます。これにより、登録に必要な複数のサブパラメーターを定義することができます。以下に例を示します。

```
parameter_defaults:
  RhsmVars:
    rhsm_repos:
      - rhel-8-for-x86_64-baseos-eus-rpms
      - rhel-8-for-x86_64-appstream-eus-rpms
      - rhel-8-for-x86_64-highavailability-eus-rpms
      …
    rhsm_username: "myusername"
    rhsm_password: "p@55w0rd!"
    rhsm_org_id: "1234567"
    rhsm_release: 8.2
```

RhsmVars パラメーターをロール固有のパラメーター (例: **ControllerParameters**) と共に使用することにより、異なるノードタイプ用の特定のリポジトリーを有効化する場合に柔軟性を提供することもできます。

11.2. RHSMVARS サブパラメーター

すべての Ansible パラメーターを把握するには、ロールに関するドキュメント を参照してください。

rhsm	説明
rhsm_method	登録の方法を選択します。**portal**、**satellite**、または **disable** のいずれかです。
rhsm_org_id	登録に使用する組織。この ID を特定するには、アンダークラウドノードから **sudo subscription-manager orgs** を実行します。プロンプトが表示されたら、Red Hat の認証情報を入力して、出力される **Key** 値を使用します。組織の ID についての詳細は、「Red Hat Subscription Management における組織 ID について理解する」を参照してください。

rhsm	説明
rhsm_pool_ids	使用するサブスクリプションプール ID。サブスクリプションを自動でアタッチしない場合には、このパラメーターを使用してください。この ID を特定するには、アンダークラウドノードから **sudo subscription-manager list --available --all --matches="OpenStack"** を実行し、出力される **Pool ID** の値を使用します。
rhsm_activation_key	登録に使用するアクティベーションキー。**rhsm_repos** が設定されている場合には機能しません。
rhsm_autosubscribe	互換性のあるサブスクリプションをこのシステムに自動的にアタッチします。有効にするには、**true** に設定します。
rhsm_baseurl	コンテンツを取得するためのベース URL。デフォルトは Red Hat コンテンツ配信ネットワークの URL です。Satellite サーバーを使用している場合は、この値を Satellite サーバーコンテンツリポジトリーのベース URL に変更します。
rhsm_server_hostname	登録用のサブスクリプション管理サービスのホスト名。デフォルトは Red Hat Subscription Management のホスト名です。Satellite サーバーを使用している場合は、この値を Satellite サーバーのホスト名に変更します。
rhsm_repos	有効にするリポジトリーの一覧。**rhsm_activation_key** が設定されている場合には機能しません。
rhsm_username	登録用のユーザー名。可能な場合には、登録にアクティベーションキーを使用します。
rhsm_password	登録用のパスワード。可能な場合には、登録にアクティベーションキーを使用します。
rhsm_release	リポジトリー固定用の Red Hat Enterprise Linux リリース。Red Hat OpenStack Platform 16.1 の場合、これは **8.2** に設定されます。
rhsm_rhsm_proxy_hostname	HTTP プロキシーのホスト名。たとえば、**proxy.example.com**。
rhsm_rhsm_proxy_port	HTTP プロキシー通信用のポート。たとえば、**8080**。
rhsm_rhsm_proxy_user	HTTP プロキシーにアクセスするためのユーザー名
rhsm_rhsm_proxy_password	HTTP プロキシーにアクセスするためのパスワード

11.3. RHSM コンポーザブルサービスへの切り替え

従来の **rhel-registration** メソッドは、bash スクリプトを実行してオーバークラウドの登録を処理しま

す。この方法のスクリプトと環境ファイルは、**/usr/share/openstack-tripleo-heat-templates/extraconfig/pre_deploy/rhel-registration/** のコア Heat テンプレートコレクションにあります。

rhel-registration メソッドを **rhsm** コンポーザブルサービスに切り替えるには、以下の手順を実施します。

手順

1. **rhel-registration** 環境ファイルは、今後のデプロイメント操作から除外します。通常は、以下のファイルを除外します。
   - **rhel-registration/environment-rhel-registration.yaml**
   - **rhel-registration/rhel-registration-resource-registry.yaml**
2. カスタムの **roles_data** ファイルを使用する場合には、**roles_data** ファイルの各ロールに必ず **OS::TripleO::Services::Rhsm** コンポーザブルサービスを含めてください。以下に例を示します。

```
- name: Controller
  description: |
    Controller role that has all the controller services loaded and handles
    Database, Messaging and Network functions.
  CountDefault: 1
  ...
  ServicesDefault:
    ...
    - OS::TripleO::Services::Rhsm
    ...
```

3. **rhsm** コンポーザブルサービスのパラメーター用の環境ファイルを今後のデプロイメント操作に追加します。

このメソッドは、**rhel-registration** パラメーターを **rhsm** サービスのパラメーターに置き換えて、サービスを有効化する Heat リソースを変更します。

```
resource_registry:
  OS::TripleO::NodeExtraConfig: rhel-registration.yaml
```

上記の行を以下のように変更します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
```

デプロイメントに **/usr/share/openstack-tripleo-heat-templates/environments/rhsm.yaml** 環境ファイルを追加して、サービスを有効にすることもできます。

11.4. RHEL-REGISTRATION から RHSM へのマッピング

rhel-registration	rhsm / RhsmVars
rhel_reg_method	**rhsm_method**
rhel_reg_org	**rhsm_org_id**
rhel_reg_pool_id	**rhsm_pool_ids**
rhel_reg_activation_key	**rhsm_activation_key**
rhel_reg_auto_attach	**rhsm_autosubscribe**
rhel_reg_sat_url	**rhsm_satellite_url**
rhel_reg_repos	**rhsm_repos**
rhel_reg_user	**rhsm_username**
rhel_reg_password	**rhsm_password**
rhel_reg_release	**rhsm_release**
rhel_reg_http_proxy_host	**rhsm_rhsm_proxy_hostname**
rhel_reg_http_proxy_port	**rhsm_rhsm_proxy_port**
rhel_reg_http_proxy_username	**rhsm_rhsm_proxy_user**
rhel_reg_http_proxy_password	**rhsm_rhsm_proxy_password**

11.5. RED HAT SATELLITE へのオーバークラウドの登録

ノードを Red Hat カスタマーポータルではなく Red Hat Satellite に登録するには、**rhsm** コンポーザブルサービスを有効にして設定する環境ファイルを作成します。その手順を以下に示します。

手順

1. 設定を保存するための環境ファイル (**templates/rhsm.yaml**) を作成します。
2. 環境ファイルに設定を追加します。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::Rhsm: /usr/share/openstack-tripleo-heat-
templates/deployment/rhsm/rhsm-baremetal-ansible.yaml
parameter_defaults:
  RhsmVars:
    rhsm_activation_key: "myactivationkey"
    rhsm_method: "satellite"
    rhsm_org_id: "ACME"
```

```
    rhsm_server_hostname: satellite.example.com"
    rhsm_baseurl: "https://satellite.example.com/pulp/repos"
    rhsm_release: 8.2
```

resource_registry は、各ロールで利用可能な **OS::TripleO::Services::Rhsm** リソースに **rhsm** コンポーザブルサービスを関連付けます。

RhsmVars の変数は、Red Hat の登録を設定するためにパラメーターを Ansible に渡します。

3. 環境ファイルを保存します。

11.6. SATELLITE サーバーを使用するための LEAPP の準備

Satellite 6 を使用して RPM コンテンツをホストする場合は、以下の準備手順を実施して、Satellite を使用して Leapp によるアップグレードが正常に完了するようにします。

前提条件

- Red Hat OpenStack Platform 16.1 および Red Hat Enterprise Linux 8.2 のリポジトリーにリンクされた Satellite サーバーのアクティベーションキー
- オーバークラウドノードの Ansible インベントリーファイル

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. **playbook-satellite.yaml** という名前のファイルを作成し、以下のコンテンツをファイルに貼り付けます。

```
- name: Pre-install leapp
  hosts: overcloud
  become: yes
  tasks:
    - name: Pre-install leapp
      yum:
        name:
          - leapp
          - leapp-repository
        state: installed
    - name: Remove katello-host-tools-fact-plugin
      yum:
        name:
          - katello-host-tools-fact-plugin
        state: removed
    - name: Register system
      import_role:
        name: redhat-subscription
      vars:
        rhsm_activation_key: "osp16-key"
        rhsm_method: "satellite"
        rhsm_org_id: "ACME"
        rhsm_server_hostname: "satellite.example.com"
```

```
    rhsm_baseurl: "https://satellite.example.com/pulp/repos"
    rhsm_release: "8.2"
    rhsm_force_register: True
```

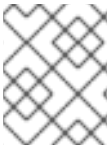
rhsm_* 変数をお使いの Satellite サーバーに合わせて変更します。

3. Playbook を実行します。

```
$ ansible-playbook -i ~/inventory.yaml playbook-satellite.yaml
```

第12章 DIRECTOR でデプロイされた CEPH STORAGE のアップグレードの準備

director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、本項に記載する手順を完了する必要があります。



注記

外部の Ceph デプロイメントと共にアップグレードする場合は、本項に記載する手順を省略し、次のセクションに進む必要があります。

Red Hat OpenStack Platform 16.1 へのアップグレード期間中、プロセスでは Red Hat Ceph Storage 3 のコンテナー化されたサービスを継続して使用します。Red Hat OpenStack Platform 16.1 へのアップグレードが完了したら、Ceph Storage サービスを Red Hat Ceph Storage 4 にアップグレードします。

12.1. CEPH STORAGE ノードのアップグレードプロセスの概要

オーバークラウドのアップグレードプロセス中、director でデプロイされた Ceph Storage ノードは Red Hat Ceph Storage 3 コンテナーを継続して使用します。アップグレードプロセス中に Ceph Storage ノードおよびサービスがどのように影響を受けるかについて理解するには、Ceph Storage アップグレードプロセスの各要素ごとに以下の概要を参照してください。

ceph-ansible

ceph-ansible は、ロールおよび Playbook のコレクションです。director はこれを使用して Ceph Storage サービスのインストール、維持、およびアップグレードを行います。アンダークラウドをアップグレードする際に実行した特定のコマンドにより、Red Hat Enterprise Linux 8.2 への移行後に **ceph-ansible** は最新のバージョン 3 コレクションの状態を維持します。バージョン 3 の **ceph-ansible** は、オーバークラウドのアップグレード期間中、バージョン 3 のコンテナー化された Ceph Storage サービスを維持します。アップグレードが完了したら、**Red Hat Ceph Storage Tools 4 for RHEL 8**リポジトリーを有効にし、**ceph-ansible** をバージョン 4 に更新します。

Podman への移行

オーバークラウドのアップグレード時に **openstack overcloud external-upgrade run --tags ceph_systemd** コマンドを実行し、Ceph Storage のコンテナー化されたサービスを制御する **systemd** サービスが Docker ではなく Podman を使用するように変更する必要があります。Ceph Storage のコンテナー化されたサービスが含まれるいずれかのノードでオペレーティングシステムをアップグレードする前に、このコマンドを実行します。

systemd サービスがノード上で Podman を使用するように変更したら、オペレーティングシステムのアップグレードおよび OpenStack Platform サービスのアップグレードを実施します。そのノード上の Ceph Storage コンテナーは、OpenStack Platform サービスのアップグレード後に再度実行されます。

Ceph Storage のオペレーティングシステムのアップグレード

概略としては、Ceph Storage ノードで、オーバークラウドノードで実施するのと同じワークフローに従います。Ceph Storage ノードに対して **openstack overcloud upgrade run --tags system_upgrade** コマンドを実行すると、director は Ceph Storage ノードで Leapp を実行し、オペレーティングシステムを Red Hat Enterprise Linux 8.2 にアップグレードします。続いて、Ceph Storage ノードに対してタグ付けされていない **openstack overcloud upgrade run** コマンドを実行します。これにより、以下のコンテナーが実行されます。

- Red Hat Ceph Storage 3 のコンテナー化されたサービス

- Red Hat OpenStack Platform 16.1 のコンテナー化されたサービス

Red Hat Ceph Storage 4 へのアップグレード

Leapp によるアップグレードおよび Red Hat OpenStack Platform のアップグレードを完了した後も、Ceph Storage のコンテナー化されたサービスは引き続きバージョン 3 のコンテナーを使用します。この時点で **ceph-ansible** をバージョン 4 にアップグレードする必要があります。続いて **openstack overcloud external-upgrade run --tags ceph** コマンドを実行し、すべてのノード上の Red Hat Ceph Storage サービスをすべてバージョン 4 にアップグレードします。

Ceph Storage ワークフローの概要

Red Hat Ceph Storage をアップグレードするワークフローの概要を以下に示します。このワークフローは Red Hat OpenStack Platform の全体的なワークフローに統合されていて、アンダークラウド上でアップグレードフレームワークのコマンドを実行すると、このワークフローの操作が実施されます。

1. アンダークラウドをアップグレードします。ただし、バージョン 3 の **ceph-ansible** を維持します。
2. オーバークラウドのアップグレードを開始します。
3. Ceph Storage のコンテナー化されたサービスをホストするノードごとに、以下のタスクを実行します。
    a. Ceph Storage のコンテナー化されたサービスを Podman に移行する。
    b. オペレーティングシステムをアップグレードする。
    c. OpenStack Platform サービスをアップグレードする。これにより、Ceph Storage バージョン 3 のコンテナー化されたサービスが再起動されます。
4. オーバークラウドのアップグレードを完了します。
5. アンダークラウドで **ceph-ansible** をバージョン 4 にアップグレードします。
6. オーバークラウドで Red Hat Ceph Storage 4 にアップグレードします。

注記

このリストは Red Hat OpenStack Platform 16.1 アップグレードプロセス全体の全ステップを網羅している訳ではありません。アップグレードプロセス中 Ceph Storage サービスに何が起こるかを説明するために、Red Hat Ceph Storage に該当する要素にのみ焦点を当てています。

12.2. CEPH-ANSIBLE バージョンの確認

アンダークラウドのアップグレード中、Ceph Storage 3 バージョンの **ceph-ansible** パッケージを維持します。これは、Ceph Storage ノード上の Ceph Storage 3 コンテナーの互換性を維持するのに役立ちます。このパッケージがアンダークラウドに残っていることを確認します。

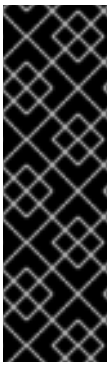
手順

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. **dnf** コマンドを実行して、**ceph-ansible** パッケージのバージョンを確認します。

```
$ sudo dnf info ceph-ansible
```

コマンド出力には、**ceph-ansible** パッケージがバージョン 3 であることが示されます。

```
Installed Packages
Name        : ceph-ansible
Version     : 3.xx.xx.xx
...
```



重要

ceph-ansible パッケージが見つからない、またはバージョン 3 のパッケージではない場合は、Red Hat Package Browser から最新のバージョン 3 パッケージをダウンロードし、アンダークラウドにパッケージを手動でインストールします。**ceph-ansible** バージョン 3 パッケージは Red Hat Enterprise Linux 7 リポジトリーからのみ利用可能で、Red Hat Enterprise Linux 8 リポジトリーでは利用できない点に注意してください。**ceph-ansible** バージョン 3 は、Red Hat OpenStack Platform のアップグレードフレームワークでの使用を除き、Red Hat Enterprise Linux 8 ではサポートされていません。

12.3. CEPH-ANSIBLE リポジトリーの設定

director がオーバークラウドを Red Hat Ceph Storage 4 にアップグレードする前に、Red Hat OpenStack Platform 16.1 の検証フレームワークで **ceph-ansible** が適切にインストールされていることを確認します。フレームワークは、**CephAnsibleRepo** パラメーターを使用して、**ceph-ansible** が正しいリポジトリーからインストールされていることを確認します。**openstack overcloud upgrade prepare** コマンドの実行後に director はこのテストを無効にし、テストは Red Hat OpenStack Platform 16.1 オーバークラウドのアップグレード期間中は無効なままとなります。**openstack overcloud upgrade converge** コマンドの実行後に、director はこのテストを再度有効にします。ただし、この検証を準備するために、**CephAnsibleRepo** パラメーターを **Red Hat Ceph Storage Tools 4 for RHEL 8** リポジトリーに設定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. オーバークラウドの Ceph Storage 設定が含まれる環境ファイルを編集します。通常、このファイルは **ceph-config.yaml** という名前で **templates** ディレクトリーにあります。

```
$ vi /home/stack/templates/ceph-config.yaml
```

3. **parameter_defaults** セクションに **CephAnsibleRepo** パラメーターを追加します。

```
parameter_defaults:
  ...
  CephAnsibleRepo: rhceph-4-tools-for-rhel-8-x86_64-rpms
  ...
```

CephAnsibleRepo は、**ceph-ansible** が含まれるリポジトリーを設定します。検証フレームワークでは、このパラメーターを使用して、アンダークラウドに **ceph-ansible** がインストールされていることを確認します。

4. **ceph-config.yaml** ファイルを保存します。

12.4. アップグレード前の CEPH クラスターステータスの確認

オーバークラウドのアップグレードを進める前に、Ceph クラスターが想定どおりに機能していることを確認する必要があります。

手順

1. **ceph-mon** サービスを実行しているノードにログインします。このノードは、通常コントローラーノードまたはスタンドアロンの Ceph Monitor ノードです。
2. 以下のコマンドを入力して、Ceph クラスターのステータスを表示します。

```
$ docker exec ceph-mon-$HOSTNAME ceph -s
```

3. クラスターの健全性ステータスが **HEALTH_OK** であること、およびすべての OSD が **up** の状態にあることを確認します。

第13章 外部の CEPH デプロイメントと組み合わせたアップグレードの準備

外部の Ceph デプロイメントと共にアップグレードする場合は、本項に記載する手順を完了する必要があります。

注記

デプロイメントで外部の Ceph Storage クラスターが使用されない場合は、本項に記載する手順を省略し、次のセクションに進む必要があります。

13.1. CEPH-ANSIBLE のインストール

外部の Ceph デプロイメントと共にアップグレードする場合は、以下の手順を完了する必要があります。

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

手順

1. Ceph Tools リポジトリーを有効にします。

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
```

2. **ceph-ansible** パッケージをインストールします。

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

13.2. CEPH-ANSIBLE リポジトリーの設定

director がオーバークラウドを Red Hat Ceph Storage 4 にアップグレードする前に、Red Hat OpenStack Platform 16.1 の検証フレームワークで **ceph-ansible** が適切にインストールされていることを確認します。フレームワークは、**CephAnsibleRepo** パラメーターを使用して、**ceph-ansible** が正しいリポジトリーからインストールされていることを確認します。**openstack overcloud upgrade prepare** コマンドの実行後に director はこのテストを無効にし、テストは Red Hat OpenStack Platform 16.1 オーバークラウドのアップグレード期間中は無効なままとなります。**openstack overcloud upgrade converge** コマンドの実行後に、director はこのテストを再度有効にします。ただし、この検証を準備するために、**CephAnsibleRepo** パラメーターを **Red Hat Ceph Storage Tools 4 for RHEL 8** リポジトリーに設定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. オーバークラウドの Ceph Storage 設定が含まれる環境ファイルを編集します。通常、このファイルは **ceph-config.yaml** という名前で **templates** ディレクトリーにあります。

```
$ vi /home/stack/templates/ceph-config.yaml
```

3. **parameter_defaults** セクションに **CephAnsibleRepo** パラメーターを追加します。

```
parameter_defaults:
  ...
  CephAnsibleRepo: rhceph-4-tools-for-rhel-8-x86_64-rpms
  ...
```

CephAnsibleRepo は、**ceph-ansible** が含まれるリポジトリーを設定します。検証フレームワークでは、このパラメーターを使用して、アンダークラウドに **ceph-ansible** がインストールされていることを確認します。

4. **ceph-config.yaml** ファイルを保存します。

第14章 ネットワーク設定の更新

オーバークラウドのアップグレードに向けた準備を行うには、いくつかのネットワーク設定を完了する必要があります。

14.1. ネットワークインターフェーステンプレートの更新

Red Hat OpenStack Platform には、不足しているパラメーターを NIC テンプレートファイルに自動的に追加するスクリプトが用意されています。

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

```
$ source ~/stackrc
```

3. アンダークラウドにおいて、以下の内容で **update-nic-templates.sh** という名前のファイルを作成します。

```
#!/bin/bash
STACK_NAME="overcloud"
ROLES_DATA="/usr/share/openstack-tripleo-heat-templates/roles_data.yaml"
NETWORK_DATA="/usr/share/openstack-tripleo-heat-templates/network_data.yaml"
NIC_CONFIG_LINES=$(openstack stack environment show $STACK_NAME | grep
"::Net::SoftwareConfig" | sed -E 's/ *OS::TripleO:// ; s/::Net::SoftwareConfig:// ; s/ http.*user-
files/ /')
echo "$NIC_CONFIG_LINES" | while read LINE; do
    ROLE=$(echo "$LINE" | awk '{print $1;}')
    NIC_CONFIG=$(echo "$LINE" | awk '{print $2;}')

    if [ -f "$NIC_CONFIG" ]; then
        echo "Updating template for $ROLE role."
        python3 /usr/share/openstack-tripleo-heat-templates/tools/merge-new-params-nic-
config-script.py \
            --tht-dir /usr/share/openstack-tripleo-heat-templates \
            --roles-data $ROLES_DATA \
            --network-data $NETWORK_DATA \
            --role-name "$ROLE" \
            --discard-comments yes \
            --template "$NIC_CONFIG"
    else
        echo "No NIC template detected for $ROLE role. Skipping $ROLE role."
    fi
done
```

- カスタムのオーバークラウド名を使用している場合には、**STACK_NAME** 変数をそのオーバークラウドの名前に設定します。オーバークラウドスタックのデフォルト名は **overcloud** です。
- カスタムの **roles_data** ファイルを使用する場合は、**ROLES_DATA** 変数をカスタムファイルの場所に設定します。デフォルトの **roles_data** ファイルを使用する場合には、変数を **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** のままにします。

- カスタムの **network_data** ファイルを使用する場合は、**NETWORK_DATA** 変数をカスタムファイルの場所に設定します。デフォルトの **network_data** ファイルを使用する場合には、変数を **/usr/share/openstack-tripleo-heat-templates/network_data.yaml** のままにします。

- **/usr/share/openstack-tripleo-heat-templates/tools/merge-new-params-nic-config-script.py -h** を実行して、スクリプトに追加するオプションの一覧を確認します。

4. スクリプトに実行権限を追加します。

```
$ chmod +x update-nic-templates.sh
```

5. スクリプトを実行します。

```
$ ./update-nic-templates.sh
```

スクリプトにより各カスタム NIC テンプレートのコピーが保存され、不足しているパラメーターで各テンプレートが更新されます。このスクリプトは、カスタムテンプレートを持たないロールもスキップします。

```
No NIC template detected for BlockStorage role. Skipping BlockStorage role.
Updating template for CephStorage role.
The original template was saved as: /home/stack/templates/custom-nics/ceph-storage.yaml.20200903144835
The update template was saved as: /home/stack/templates/custom-nics/ceph-storage.yaml
Updating template for Compute role.
The original template was saved as: /home/stack/templates/custom-nics/compute.yaml.20200903144838
The update template was saved as: /home/stack/templates/custom-nics/compute.yaml
Updating template for Controller role.
The original template was saved as: /home/stack/templates/custom-nics/controller.yaml.20200903144841
The update template was saved as: /home/stack/templates/custom-nics/controller.yaml
No NIC template detected for ObjectStorage role. Skipping ObjectStorage role.
```

14.2. アップグレード中の OPEN VSWITCH との互換性の維持

Red Hat OpenStack Platform 13 では、OpenStack Networking (neutron) のデフォルト ML2 バックエンドとして Open vSwitch (OVS) が使用されます。新しいバージョンの Red Hat OpenStack Platform では、OVS 機能を拡張した Open Virtual Network (OVN) が使用されます。ただし、安定したアップグレードを確保するには、アップグレードの期間中 OVS 機能を維持し、アップグレードの完了後に OVN に移行する必要があります。

アップグレード中に OVS との互換性を維持するには、環境ファイルコレクションの一部として以下の環境ファイルを追加します。

- **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml**

OVN への移行が完了するまで、このファイルをデプロイメントの一部として扱います。すべてのオーバークラウドアップグレードコマンドおよびデプロイメントコマンドに、このファイルを追加します。

- **openstack overcloud upgrade prepare**

- **openstack overcloud upgrade converge**

- **openstack overcloud deploy**
- **openstack overcloud update prepare**
- **openstack overcloud update converge**
- 環境ファイルを使用するその他すべてのコマンド

14.3. アップグレード中のコンポーザブルネットワーク互換性の維持

Red Hat OpenStack Platform 16.1 の **network_data** ファイル形式には、ネットワークの追加サブネットおよびルートを定義するのに使用できる新しいセクションが含まれています。ただし、カスタムの **network_data** ファイルを使用している場合は、引き続き Red Hat OpenStack Platform 13 からの **network_data** ファイル形式を使用することができます。

- Red Hat OpenStack Platform 13 から 16.1 にアップグレードする場合は、アップグレード中およびアップグレード後に Red Hat OpenStack Platform 13 の **network_data** ファイル形式を使用します。Red Hat OpenStack Platform 13 コンポーザブルネットワークの構文についての詳しい情報は、「カスタムコンポーザブルネットワーク」を参照してください。
- Red Hat OpenStack Platform 16.1 に新しいオーバークラウドを作成する場合には、Red Hat OpenStack Platform 16.1 の **network_data** ファイル形式を使用します。Red Hat OpenStack Platform 16.1 コンポーザブルネットワークの構文についての詳しい情報は、「カスタムコンポーザブルネットワーク」を参照してください。

第15章 ネットワーク機能仮想化 (NFV) の準備

ネットワーク機能仮想化 (NFV) を使用する場合は、オーバークラウドのアップグレードに向けて準備タスクを完了する必要があります。

15.1. ネットワーク機能仮想化 (NFV) 用環境ファイル

典型的な NFV ベースの環境では、以下のようなサービスを有効にすることができます。

- Single-root input/output virtualization (SR-IOV)
- Data Plane Development Kit (DPDK)

Red Hat OpenStack Platform 16.1 へのアップグレードに対応するために、これらのサービスに対して特定の再設定を行う必要はありません。ただし、NFV 機能を有効にする環境ファイルは、以下の要件を満たすようにしてください。

- NFV 機能を有効にするデフォルトの環境ファイルは、Red Hat OpenStack Platform 16.1 **openstack-tripleo-heat-templates** コレクションの **environments/services** ディレクトリーにあります。Red Hat OpenStack Platform 13 デプロイメントに **openstack-tripleo-heat-templates** からのデフォルト NFV 環境ファイルを追加している場合は、Red Hat OpenStack Platform 16.1 での該当機能の正しい環境ファイルの場所を確認してください。
  - Open vSwitch (OVS) ネットワークおよび SR-IOV: **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-sriov.yaml**
  - Open vSwitch (OVS) ネットワークおよび DPDK: **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml**
- Red Hat OpenStack Platform 13 から Red Hat OpenStack Platform 16.1 へのアップグレード中に OVS との互換性を維持するには、**/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml** 環境ファイルを追加する必要があります。環境ファイルを指定してデプロイメントおよびアップグレードのコマンドを実行する際には、**neutron-ovs.yaml** ファイルの後に NFV 関連の環境ファイルをすべて追加する必要があります。たとえば、OVS および NFV 用環境ファイルを指定して **openstack overcloud upgrade prepare** を実行する場合は、以下の順序でファイルを追加します。
- OVS 用環境ファイル
- SR-IOV 用環境ファイル
- DPDK 用環境ファイル

```
$ openstack overcloud upgrade prepare \
    ...
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-sriov.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml \
    ...
```

第16章 アップグレード前の最終確認

アップグレードを開始する前に、すべての準備手順の最終確認を行います。

16.1. デプロイメントに追加する新たな環境ファイル

通常のオーバークラウドの環境ファイルに加えて、Red Hat OpenStack Platform 16.1 へのアップグレードを円滑に行うために、新しい環境ファイルを追加する必要があります。

ファイル	備考
/home/stack/templates/upgrades-environment.yaml	このファイルには、アップグレードに固有のパラメーターが含まれます。このファイルは、アップグレード期間中にのみ必要です。**openstack overcloud upgrade converge** を実行した後は、このファイルを破棄します。
/home/stack/templates/rhsm.yaml	このファイルには、オーバークラウドの登録およびサブスクリプション情報が含まれます。このファイルにより、お使いのシステムを Red Hat カスタマーポータルまたは Red Hat Satellite サーバーのいずれかに登録します。
/home/stack/containers-prepare-parameter.yaml	ソースおよび準備の手順が含まれるファイルです。これは、アンダークラウドのアップグレードに使用するファイルと同じです。
/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml	OpenStack Platform 16.1 では、デフォルトのネットワークバックエンドとして Open Virtual Network (OVN) が使用されます。しかし、OpenStack Platform 13 では Open vSwitch (OVS) が使用されていました。アップグレード中に OVS との互換性を維持するために、このファイルを追加します。OpenStack Platform 16.1 のドキュメントには、アップグレード後に OVS から OVN に移行するための手順が記載されています。

以下のコマンドを実行する際に、環境ファイル一覧の最後にこれらのファイルを追加します。

- **openstack overcloud upgrade prepare**
- **openstack overcloud upgrade converge**
- **openstack overcloud deploy**

16.2. デプロイメントから削除する環境ファイル

OpenStack Platform Red Hat OpenStack Platform 13 に固有の環境ファイルをすべて削除します。

- Red Hat OpenStack Platform 13 のコンテナーイメージー覧

- Red Hat OpenStack Platform 13 のカスタマーポータルまたは Satellite 用 **rhel-registration** スクリプト

以下のコマンドを実行する際に指定する環境ファイルの一覧から、これらのファイルを削除します。

- **openstack overcloud upgrade prepare**
- **openstack overcloud upgrade converge**
- **openstack overcloud deploy**

16.3. アップグレードのチェックリスト

以下のチェックリストを使用して、オーバークラウドをアップグレードする準備ができているかどうかを判断します。

確認項目	実施状況
動作中のオーバークラウドを検証した。	はい / いいえ
オーバークラウドコントロールプレーンの Relax-and-Recover (ReaR) バックアップを実施した。	はい / いいえ
以下の項目を含め、Leapp に関するすべての準備を実施した。 - Leapp 環境ファイル - オーバークラウドノードに Leapp データをコピーした。 - すべてのオーバークラウドノードで予測可能な NIC 名が使用されている。 - すべてのオーバークラウドノードに **PermitRootLogin** が設定されている。	はい / いいえ
登録情報を Red Hat OpenStack Platform 16.1 リポジトリーに更新し、Ansible ベースの手法を使用するように環境ファイルを変更した。	はい / いいえ
ネットワーク設定テンプレートを更新した。	はい / いいえ
環境ファイルの一覧を Red Hat OpenStack Platform 16.1 用の新しい環境ファイルで更新した。	はい / いいえ
古い Red Hat 登録やコンテナーイメージの場所に関するファイルなど、Red Hat OpenStack Platform 13 にしか該当しない古い環境ファイルを削除した。	はい / いいえ

パート III. オーバークラウドのアップグレード

第17章 アップグレードコマンドの概要

アップグレードプロセスでは、プロセスの特定の段階で異なるコマンドを実行します。

重要

本セクションでは、それぞれのコマンドについてのみ説明します。これらのコマンドは特定の順序で実行し、オーバークラウドに固有のオプションを指定する必要があります。適切なステップでこれらのコマンドを実行するよう指示されるまで待ちます。

17.1. OPENSTACK OVERCLOUD UPGRADE PREPARE

このコマンドにより、オーバークラウドのアップグレードの初期準備のステップが実行されます。これには、アンダークラウド上の現在のオーバークラウドプランを新しい OpenStack Platform 16.1 オーバークラウドプランおよび更新された環境ファイルに置き換えることが含まれます。このコマンドは、**openstack overcloud deploy** コマンドと同じように機能し、多くの同一オプションが使用されます。

17.2. OPENSTACK OVERCLOUD UPGRADE RUN

このコマンドにより、アップグレードプロセスが実施されます。director は、新しい OpenStack Platform 16.1 オーバークラウドプランに基づいて Ansible Playbook のセットを作成し、オーバークラウド全体で Fast Forward タスクを実行します。これには、OpenStack Platform の 13 から 16.1 までの各バージョンでアップグレードプロセスを実行することが含まれます。

標準のアップグレードプロセスに加えて、このコマンドによりオーバークラウドノード上のオペレーティングシステムの Leapp アップグレードを実施することができます。**--tags** オプションを使用して、これらのタスクを実行します。

Leapp のアップグレードタスクタグ

system_upgrade

system_upgrade_prepare、**system_upgrade_run**、および **system_upgrade_reboot** のタスクを組み合わせるタスク

system_upgrade_prepare

Leapp を使用したオペレーティングシステムのアップグレードに向けた準備を行うタスク

system_upgrade_run

Leapp を実行し、オペレーティングシステムをアップグレードするタスク

system_upgrade_reboot

システムをリブートし、オペレーティングシステムのアップグレードを完了するタスク

ワークロード移行のアップグレードタスクタグ

nova_hybrid_state

アップグレード中のワークロードの移行を円滑に行うために、コンピュートノード上に一時的な OpenStack Platform 16.1 コンテナーをセットアップするタスク

17.3. OPENSTACK OVERCLOUD EXTERNAL-UPGRADE RUN

このコマンドにより、標準のアップグレードプロセス以外のアップグレードタスクが実行されます。director は新しい OpenStack Platform 16.1 オーバークラウドプランに基づいて Ansible Playbook のセットを作成するので、**--tags** オプションを使用して特定のタスクを実行します。

コンテナー管理の外部タスクタグ

container_image_prepare

アンダークラウドレジストリーにコンテナーイメージをプルし、オーバークラウドが使用するようにイメージを準備するタスク

Ceph Storage アップグレードの外部タスクタグ

- director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、以下のタグを使用することができます。

  ceph

  ceph-ansible Playbook を使用して Red Hat Ceph Storage をインストールするタスク

  ceph_systemd

  podman 管理を使用するために、Red Hat Ceph Storage の systemd ユニットファイルを変換するタスク

- 外部の Ceph デプロイメントと共にアップグレードする場合は、**ceph** および **ceph_systemd** タグを使用するタスクを省略することができます。

データベース移行の外部タスクタグ

system_upgrade_cleanup

system_upgrade_transfer_data タスクに関連するストレージディレクトリーを消去するタスク

system_upgrade_stop_services

すべてのサービスをシャットダウンするタスク

system_upgrade_transfer_data

すべてのサービスをシャットダウンし、ブートストラップノードへのデータベース移行を実施するタスク

17.4. OPENSTACK OVERCLOUD UPGRADE CONVERGE

このコマンドにより、オーバークラウドのアップグレードの最終ステップが実施されます。この最終ステップでは、オーバークラウドの heat スタックを OpenStack Platform 16.1 のオーバークラウドプランおよび更新された環境ファイルと同期します。このプロセスにより、作成されるオーバークラウドが新しい OpenStack Platform 16.1 オーバークラウドの設定と一致するようになります。このコマンドは **openstack overcloud deploy** コマンドと類似していて、多くの同一オプションが使用されます。

17.5. オーバークラウドノードのアップグレードワークフロー

各オーバークラウドノードでアップグレードを実施する場合、以下の要素を考慮して、アップグレードの各段階で実行する正しいコマンドを判断する必要があります。

コントローラーサービス

- **ノードに Pacemaker サービスが含まれているか?** まず、ブートストラップノードをアップグレードする必要があります。これには、Red Hat OpenStack 13 から 16.1 へのアップグレード時

の移行を円滑に行うためのデータベース移行および一時コンテナーの起動が含まれます。ブートストラップノードをアップグレードしたら、Pacemaker サービスが含まれるその他のノードをアップグレードし、各ノードがブートストラップノードで開始した新しい Pacemaker クラスターに参加するようにします。Pacamaker が含まれない split-service コントローラーノードをアップグレードするプロセスでは、これらの追加手順は必要ありません。

Compute サービス

- **ノードがコンピュートノードか?** ノードに Compute サービスが含まれる場合、そのノードから仮想マシンを移行して最大限の可用性を確保する必要があります。ここで言うコンピュートノードには、仮想マシンをホストするためのあらゆるノードが含まれます。この定義には、以下のコンピュートノード種別が含まれます。
  - 通常のコンピュートノード
  - ハイパーコンバージドインフラストラクチャー (HCI) を持つコンピュートノード
  - Data Plane Development Kit (DPDK) または Single Root Input/Output Virtualization (SR-IOV) 等のネットワーク機能仮想化技術を使用するコンピュートノード
  - リアルタイムコンピュートノード

Ceph Storage サービス

- **ノードに Ceph Storage サービスが含まれているか?docker** ではなく **podman** を使用するために、ノード上のコンテナー化された Ceph Storage サービスの **systemd** ユニットファイルを変換する必要があります。以下のノード種別がこれに該当します。
  - Ceph Storage OSD ノード
  - Ceph MON サービスが含まれるコントローラーノード
  - Split-Controller Ceph MON ノード
  - ハイパーコンバージドインフラストラクチャー (HCI) を持つコンピュートノード

ワークフロー

以下のワークフロー図を使用して、特定ノードの正しい更新パスを特定します。

第18章 標準的なオーバークラウドのアップグレード

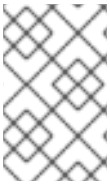
このシナリオでは、標準的なオーバークラウド環境のアップグレードプロセスの例を説明します。この環境には、以下のノード種別が含まれます。

- コントローラーノード 3 台
- Ceph Storage ノード 3 台
- 複数のコンピュートノード

18.1. オーバークラウドアップグレード準備タスクの実行

アップグレードを行うには、**openstack overcloud upgrade prepare** コマンドを実行する必要があります。このコマンドにより、以下のタスクが実行されます。

- オーバークラウドのプランを OpenStack Platform 16.1 に更新する。
- アップグレードに向けてノードを準備する。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. upgrade prepare コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)

- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

3. アップグレードの準備が完了するまで待ちます。
4. コンテナーイメージをダウンロードします。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
container_image_prepare
```

18.2. DIRECTOR でデプロイされた CEPH STORAGE と組み合わせたコントローラーノードのアップグレード

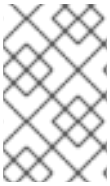
director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、以下の手順を完了する必要があります。

すべてのコントローラーノードを OpenStackPlatform 16.1 にアップグレードします。このプロセスでは、ブートストラップコントローラーノードから始めて各コントローラーノードをアップグレードします。

以下の例では、デフォルトの **overcloud-controller-NODEID** 命名規則を使用してコントローラーノードに名前を付けています。これには、以下の 3 つのコントローラーノードが含まれます。

- **overcloud-controller-0**
- **overcloud-controller-1**
- **overcloud-controller-2**

これらの値は、該当する実際のノード名に置き換てください。



注記

デフォルトのスタック名 **overcloud** を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. コントローラーノードで以下のコマンドを実行し、ブートストラップコントローラーノードを特定します。

```
$ sudo hiera -c /etc/puppet/hiera.yaml pacemaker_short_bootstrap_node_name
```

アンダークラウドからこのコマンドを実行するには、以下の SSH コマンドを実行します。ここで、**CONTROLLER_IP** は環境内の任意のコントローラーノードの IP アドレスに置き換えます。

```
$ ssh heat-admin@CONTROLLER_IP "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

以下の例では、ブートストラップノードの値として **overcloud-controller-0** を使用します。これを実際のブートストラップノードの値に置き換えてください。

3. ブートストラップコントローラーノードをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-controller-0
```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

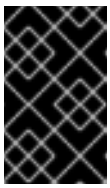
   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。



重要

次のコマンドにより、コントロールプレーンで機能停止が生じます。これ以降の数ステップを実施している間は、標準的なオーバークラウド操作を行うことはできません。

   c. **system_upgrade_transfer_data** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
system_upgrade_transfer_data
```

このコマンドにより、最新バージョンのデータベースが既存のノードからブートストラップノードにコピーされます。

d. **nova_hybrid_state** タグを指定してアップグレードコマンドを実行し、**upgrade_steps_playbook.yaml** Playbook だけを実行します。

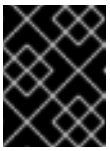
```
$ openstack overcloud upgrade run --stack STACK NAME --playbook
upgrade_steps_playbook.yaml --tags nova_hybrid_state --limit all
```

このコマンドにより、コンピュートノード上の一時的な 16.1 コンテナーが起動します。これにより、後のステップでコンピュートノードをアップグレードする際に、ワークロードの移行が円滑に行われます。

e. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。



重要

このコマンドの処理が完了すると、コントロールプレーンがアクティブになります。再び標準的なオーバークラウド操作を実施することができます。

4. 次のコントローラーノードをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-controller-1
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. 次のコントローラーノードで、**system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-1
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。このノードに加えて、前のステップでアップグレードしたブートストラップノードを **--limit** オプションに含めます。

5. 最後のコントローラーノードをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e ceph_ansible_limit=overcloud-controller-2
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-controller-2
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

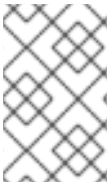
c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1,overcloud-controller-2
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。**--limit** オプションにすべてのコントローラーノードを含めます。

18.3. CEPH STORAGE ノードのオペレーティングシステムのアップグレード

director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、各 Ceph Storage ノードのオペレーティングシステムをアップグレードする必要があります。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. Ceph MON サービスが含まれるノードにログインします。
3. OSD の除外およびリバランスを無効にします。

```
$ sudo podman ps | grep ceph-mon
$ sudo podman exec -it CONTAINER ceph osd set noout
$ sudo podman exec -it CONTAINER ceph osd set norebalance
```

CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

4. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。
5. Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。
   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-0
```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-cephstorage-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-
cephstorage-0
```

■

このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

6. 次の Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

   ```
   $ openstack overcloud external-upgrade run --stack STACK NAME --tags
   ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-1
   ```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
   overcloud-cephstorage-1
   ```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   c. タグを指定せずにアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-
   cephstorage-1
   ```

   このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

7. 最後の Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

   ```
   $ openstack overcloud external-upgrade run --stack STACK NAME --tags
   ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-2
   ```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。

- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-cephstorage-2
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-
cephstorage-2
```

このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

8. すべての HCI ノードをアップグレードしたら、Ceph MON サービスをホストするノードにログインします。

9. OSD の除外およびリバランスを有効にします。

```
$ sudo podman ps | grep ceph-mon
$ sudo podman exec -it CONTAINER ceph osd unset noout
$ sudo podman exec -it CONTAINER ceph osd unset norebalance
```

CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

10. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。

18.4. コンピュートノードのアップグレード

すべてのコンピュートノードを OpenStackPlatform 16.1 にアップグレードします。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. インスタンスを移行します。移行計画についての詳細は、『インスタンス＆イメージガイド』の「コンピュートノード間の仮想マシンインスタンスの移行」を参照してください。

3. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-compute-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

4. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-compute-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

5. 複数のコンピュートノードを並行してアップグレードするには、**--limit** オプションをアップグレードするノードのコンマ区切りリストに設定します。まず **system_upgrade** タスクを実行します。

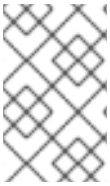
```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-compute-0,overcloud-compute-1,overcloud-compute-2
```

続いて、標準の OpenStack サービスのアップグレードを実施します。

```
$ openstack overcloud upgrade run --stack STACK NAME  --limit overcloud-compute-0,overcloud-compute-1,overcloud-compute-2
```

18.5. オーバークラウドスタックの同期

アップグレードにおいては、スタックのリソース構造およびパラメーターが OpenStack Platform 16.1 の新規デプロイメントと整合するように、オーバークラウドスタックを更新する必要があります。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **containers-prepare-parameter.yaml** ファイルを編集し、以下のパラメーターおよびその値を削除します。

- **ceph3_namespace**

- **ceph3_tag**
- **ceph3_image**
- **name_prefix_stein**
- **name_suffix_stein**
- **namespace_stein**
- **tag_stein**

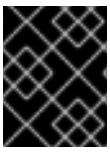
3. アップグレードの最終処理のコマンドを実行します。

```
$ openstack overcloud upgrade converge \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

4. スタックの同期が完了するまで待ちます。



重要

これ以降のデプロイメント操作には、**upgrades-environment.yaml** ファイルは必要ありません。

第19章 外部の CEPH デプロイメントと組み合わせたオーバークラウドのアップグレード

このシナリオでは、外部の Ceph デプロイメントと組み合わせたオーバークラウド環境のアップグレードプロセスの例を説明します。この環境には、以下のノード種別が含まれます。

- コントローラーノード 3 台
- 外部の Ceph Storage クラスター
- 複数のコンピュートノード

19.1. オーバークラウドアップグレード準備タスクの実行

アップグレードを行うには、**openstack overcloud upgrade prepare** コマンドを実行する必要があります。このコマンドにより、以下のタスクが実行されます。

- オーバークラウドのプランを OpenStack Platform 16.1 に更新する。
- アップグレードに向けてノードを準備する。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. upgrade prepare コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)

- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

3. アップグレードの準備が完了するまで待ちます。
4. コンテナーイメージをダウンロードします。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
container_image_prepare
```

19.2. 外部の CEPH デプロイメントと組み合わせたコントローラーノードのアップグレード

外部の Ceph デプロイメントと共にアップグレードする場合は、以下の手順を完了する必要があります。

すべてのコントローラーノードを OpenStackPlatform 16.1 にアップグレードします。このプロセスでは、ブートストラップコントローラーノードから始めて各コントローラーノードをアップグレードします。

以下の例では、デフォルトの **overcloud-controller-NODEID** 命名規則を使用してコントローラーノードに名前を付けています。これには、以下の 3 つのコントローラーノードが含まれます。

- **overcloud-controller-0**
- **overcloud-controller-1**
- **overcloud-controller-2**

これらの値は、該当する実際のノード名に置き換てください。

注記

デフォルトのスタック名 **overcloud** を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. コントローラーノードで以下のコマンドを実行し、ブートストラップコントローラーノードを特定します。

```
$ sudo hiera -c /etc/puppet/hiera.yaml pacemaker_short_bootstrap_node_name
```

アンダークラウドからこのコマンドを実行するには、以下の SSH コマンドを実行します。ここで、**CONTROLLER_IP** は環境内の任意のコントローラーノードの IP アドレスに置き換えます。

```
$ ssh heat-admin@CONTROLLER_IP "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

以下の例では、ブートストラップノードの値として **overcloud-controller-0** を使用します。これを実際のブートストラップノードの値に置き換えてください。

3. ブートストラップコントローラーノードをアップグレードします。

   a. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

重要

次のコマンドにより、コントロールプレーンで機能停止が生じます。これ以降の数ステップを実施している間は、標準的なオーバークラウド操作を行うことはできません。

   b. **system_upgrade_transfer_data** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
system_upgrade_transfer_data
```

   このコマンドにより、最新バージョンのデータベースが既存のノードからブートストラップノードにコピーされます。

   c. **nova_hybrid_state** タグを指定してアップグレードコマンドを実行し、**upgrade_steps_playbook.yaml** Playbook だけを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --playbook
upgrade_steps_playbook.yaml --tags nova_hybrid_state --limit all
```

   このコマンドにより、コンピュートノード上の一時的な 16.1 コンテナーが起動します。これにより、後のステップでコンピュートノードをアップグレードする際に、ワークロードの移行が円滑に行われます。

   d. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

重要

このコマンドの処理が完了すると、コントロールプレーンがアクティブになります。再び標準的なオーバークラウド操作を実施することができます。

4. 次のコントローラーノードをアップグレードします。

   a. 次のコントローラーノードで、**system_upgrade** タグを指定してアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-controller-1
   ```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   b. タグを指定せずにアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1
   ```

   このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。このノードに加えて、前のステップでアップグレードしたブートストラップノードを **--limit** オプションに含めます。

5. 最後のコントローラーノードをアップグレードします。

   a. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-controller-2
   ```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   b. タグを指定せずにアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1,overcloud-controller-2
   ```

   このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。**--limit** オプションにすべてのコントローラーノードを含めます。

19.3. コンピュートノードのアップグレード

すべてのコンピュートノードを OpenStackPlatform 16.1 にアップグレードします。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. インスタンスを移行します。移行計画についての詳細は、『インスタンス＆イメージガイド』の「コンピュートノード間の仮想マシンインスタンスの移行」を参照してください。

3. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-compute-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

4. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-compute-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

5. 複数のコンピュートノードを並行してアップグレードするには、**--limit** オプションをアップグレードするノードのコンマ区切りリストに設定します。まず **system_upgrade** タスクを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-compute-0,overcloud-compute-1,overcloud-compute-2
```

続いて、標準の OpenStack サービスのアップグレードを実施します。

```
$ openstack overcloud upgrade run --stack STACK NAME  --limit overcloud-compute-
0,overcloud-compute-1,overcloud-compute-2
```

19.4. オーバークラウドスタックの同期

アップグレードにおいては、スタックのリソース構造およびパラメーターが OpenStack Platform 16.1 の新規デプロイメントと整合するように、オーバークラウドスタックを更新する必要があります。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **containers-prepare-parameter.yaml** ファイルを編集し、以下のパラメーターおよびその値を削除します。
   - **ceph3_namespace**
   - **ceph3_tag**
   - **ceph3_image**
   - **name_prefix_stein**
   - **name_suffix_stein**
   - **namespace_stein**
   - **tag_stein**
3. アップグレードの最終処理のコマンドを実行します。

```
$ openstack overcloud upgrade converge \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)

- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

4. スタックの同期が完了するまで待ちます。

重要

これ以降のデプロイメント操作には、**upgrades-environment.yaml** ファイルは必要ありません。

第20章 コントローラーが分割されたオーバークラウドのアップグレード

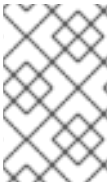
このシナリオでは、コントローラーノードサービスが複数のノードに分割されているオーバークラウドのアップグレードプロセスの例を説明します。この環境には、以下のノード種別が含まれます。

- Pacemaker を使用する複数に分割された高可用性サービス
- 複数に分割されたコントローラーサービス
- Ceph MON ノード 3 台
- Ceph Storage ノード 3 台
- 複数のコンピュートノード

20.1. オーバークラウドアップグレード準備タスクの実行

アップグレードを行うには、**openstack overcloud upgrade prepare** コマンドを実行する必要があります。このコマンドにより、以下のタスクが実行されます。

- オーバークラウドのプランを OpenStack Platform 16.1 に更新する。
- アップグレードに向けてノードを準備する。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. upgrade prepare コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)

- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

3. アップグレードの準備が完了するまで待ちます。
4. コンテナーイメージをダウンロードします。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
container_image_prepare
```

20.2. PACEMAKER ベースのノードのアップグレード

Pacemaker サービスをホストする全ノードを OpenStack Platform 16.1 にアップグレードします。以下のロールに Pacemaker ベースのサービスが含まれます。

- Controller
- Database (MySQL、Galera)
- Messaging (RabbitMQ)
- Load Balancing (HAProxy)
- 以下のサービスが含まれるその他すべてのロール
  - **OS::TripleO::Services::Pacemaker**
  - **OS::TripleO::Services::PacemakerRemote**

このプロセスでは、ブートストラップノードから始めて各ノードをアップグレードします。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. コントローラーベースのノードで以下のコマンドを実行し、ブートストラップノードを特定します。

```
$ ssh heat-admin@CONTROLLER_IP "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

3. ブートストラップノードをアップグレードします。

   a. ノードに Ceph Storage コンテナーが含まれていれば、**ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-controller-0
```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   c. **system_upgrade_transfer_data** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
system_upgrade_transfer_data
```

   このコマンドにより、最新バージョンのデータベースが既存のノードからブートストラップノードにコピーされます。

   d. **nova_hybrid_state** タグを指定してアップグレードコマンドを実行し、**upgrade_steps_playbook.yaml** Playbook だけを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --playbook
upgrade_steps_playbook.yaml --tags nova_hybrid_state --limit all
```

   このコマンドにより、コンピュートノード上の一時的な 16.1 コンテナーが起動します。これにより、後のステップでコンピュートノードをアップグレードする際に、ワークロードの移行が円滑に行われます。

e. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

4. Pacemaker ベースの各ノードをアップグレードします。

a. ノードに Ceph Storage コンテナーが含まれていれば、**ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-database-0
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. 次のノードで、**system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-database-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-
0,overcloud-database-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。このノードに加えて、前のステップでアップグレードしたすべてのノードを **--limit** オプションに含めます。

5. 各 Pacemaker ベースのノードでアップグレードプロセスを繰り返し、すべての Pacemaker ベースのノードをアップグレードします。

20.3. PACEMAKER コントローラーノード以外のアップグレード

Pacemaker ベースのサービスが含まれないノードをすべて OpenStack Platform 16.1 にアップグレードします。これらのノードには、通常特定の OpenStack サービスが含まれます。Pacemaker ベースのサービスが含まれないロールの例は以下のとおりです。

- Networker

- Ironic Conductor
- Object Storage
- 標準のコントローラーノードから分割またはスケーリングしたサービスを持つカスタムロール

このグループには、以下のノードを含めないでください。

- コンピュートノード
- Ceph Storage ノード

このプロセスでは、各ノードのアップグレードを行います。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-networker-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

3. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-networker-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

4. 各ノードでアップグレードプロセスを繰り返し、すべてのコントローラーベースのノードをアップグレードします。

20.4. CEPH MON ノードのオペレーティングシステムのアップグレード

各 Ceph MON ノードのオペレーティングシステムをアップグレードします。ノード間のクォーラムを維持するために、それぞれの Ceph MON ノードを個別にアップグレードすることを推奨します。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. Ceph MON ノードを選択し、オペレーティングシステムをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-cephmon-0
```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-cephmon-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-cephmon-0
```

   このコマンドにより **config-download** Playbook が実行され、Ceph MON ノードでコンポーザブルサービスが設定されます。以下の手順では、Ceph MON ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

3. 次の Ceph MON ノードを選択し、オペレーティングシステムをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-cephmon-1
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-cephmon-1
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-cephmon-1
```

このコマンドにより **config-download** Playbook が実行され、Ceph MON ノードでコンポーザブルサービスが設定されます。以下の手順では、Ceph MON ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

4. 最後の Ceph MON ノードを選択し、オペレーティングシステムをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e ceph_ansible_limit=overcloud-cephmon-2
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-cephmon-2
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。

- Leapp によるアップグレードの一部としてリブートを実施する。

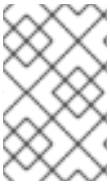
c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-cephmon-2
```

このコマンドにより **config-download** Playbook が実行され、Ceph MON ノードでコンポーザブルサービスが設定されます。以下の手順では、Ceph MON ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

20.5. CEPH STORAGE ノードのオペレーティングシステムのアップグレード

director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、各 Ceph Storage ノードのオペレーティングシステムをアップグレードする必要があります。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. Ceph MON サービスが含まれるノードにログインします。
3. OSD の除外およびリバランスを無効にします。

```
$ sudo podman ps | grep ceph-mon
$ sudo podman exec -it CONTAINER ceph osd set noout
$ sudo podman exec -it CONTAINER ceph osd set norebalance
```

CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

4. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。
5. Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-0
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-cephstorage-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-cephstorage-0
```

このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

6. 次の Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-1
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-cephstorage-1
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-cephstorage-1
```

このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

7. 最後の Ceph Storage ノードを選択し、オペレーティングシステムをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

   ```
   $ openstack overcloud external-upgrade run --stack STACK NAME --tags
   ceph_systemd -e ceph_ansible_limit=overcloud-cephstorage-2
   ```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したノードに制限する。

   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
   overcloud-cephstorage-2
   ```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。

   c. タグを指定せずにアップグレードコマンドを実行します。

   ```
   $ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-
   cephstorage-2
   ```

   このコマンドにより **config-download** Playbook が実行され、Ceph Storage ノードでコンポーザブルサービスが設定されます。このステップでは、Ceph Storage ノードは Red Hat Ceph Storage 4 にアップグレードされません。Red Hat Ceph Storage 4 へのアップグレードは、後の手順で実施されます。

8. すべての HCI ノードをアップグレードしたら、Ceph MON サービスをホストするノードにログインします。

9. OSD の除外およびリバランスを有効にします。

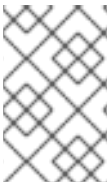
   ```
   $ sudo podman ps | grep ceph-mon
   $ sudo podman exec -it CONTAINER ceph osd unset noout
   $ sudo podman exec -it CONTAINER ceph osd unset norebalance
   ```

   CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

10. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。

20.6. コンピュートノードのアップグレード

すべてのコンピュートノードを OpenStackPlatform 16.1 にアップグレードします。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. インスタンスを移行します。移行計画についての詳細は、『インスタンス＆イメージガイド』の「コンピュートノード間の仮想マシンインスタンスの移行」を参照してください。

3. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-compute-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

4. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-compute-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

5. 複数のコンピュートノードを並行してアップグレードするには、**--limit** オプションをアップグレードするノードのコンマ区切りリストに設定します。まず **system_upgrade** タスクを実行します。

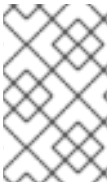
```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-compute-0,overcloud-compute-1,overcloud-compute-2
```

続いて、標準の OpenStack サービスのアップグレードを実施します。

```
$ openstack overcloud upgrade run --stack STACK NAME  --limit overcloud-compute-
0,overcloud-compute-1,overcloud-compute-2
```

20.7. オーバークラウドスタックの同期

アップグレードにおいては、スタックのリソース構造およびパラメーターが OpenStack Platform 16.1 の新規デプロイメントと整合するように、オーバークラウドスタックを更新する必要があります。



注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **containers-prepare-parameter.yaml** ファイルを編集し、以下のパラメーターおよびその値を削除します。
   - **ceph3_namespace**
   - **ceph3_tag**
   - **ceph3_image**
   - **name_prefix_stein**
   - **name_suffix_stein**
   - **namespace_stein**
   - **tag_stein**
3. アップグレードの最終処理のコマンドを実行します。

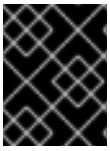
```
$ openstack overcloud upgrade converge \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)

- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

4. スタックの同期が完了するまで待ちます。



重要

これ以降のデプロイメント操作には、**upgrades-environment.yaml** ファイルは必要ありません。

第21章 ハイパーコンバージドインフラストラクチャーを持つオーバークラウドのアップグレード

このシナリオでは、ハイパーコンバージドインフラストラクチャー (HCI) を持つオーバークラウドのアップグレードプロセスの例を説明します。この環境には、以下のノード種別が含まれます。

- コントローラーノード 3 台
- 複数の HCI コンピュートノード (Compute サービスと Ceph OSD サービスが組み合わされてノードに含まれる)

21.1. オーバークラウドアップグレード準備タスクの実行

アップグレードを行うには、**openstack overcloud upgrade prepare** コマンドを実行する必要があります。このコマンドにより、以下のタスクが実行されます。

- オーバークラウドのプランを OpenStack Platform 16.1 に更新する。
- アップグレードに向けてノードを準備する。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. upgrade prepare コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)

- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

3. アップグレードの準備が完了するまで待ちます。
4. コンテナーイメージをダウンロードします。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
container_image_prepare
```

21.2. DIRECTOR でデプロイされた CEPH STORAGE と組み合わせたコントローラーノードのアップグレード

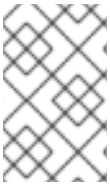
director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、以下の手順を完了する必要があります。

すべてのコントローラーノードを OpenStackPlatform 16.1 にアップグレードします。このプロセスでは、ブートストラップコントローラーノードから始めて各コントローラーノードをアップグレードします。

以下の例では、デフォルトの **overcloud-controller-NODEID** 命名規則を使用してコントローラーノードに名前を付けています。これには、以下の 3 つのコントローラーノードが含まれます。

- **overcloud-controller-0**
- **overcloud-controller-1**
- **overcloud-controller-2**

これらの値は、該当する実際のノード名に置き換てください。



注記

デフォルトのスタック名 **overcloud** を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. コントローラーノードで以下のコマンドを実行し、ブートストラップコントローラーノードを特定します。

```
$ sudo hiera -c /etc/puppet/hiera.yaml pacemaker_short_bootstrap_node_name
```

アンダークラウドからこのコマンドを実行するには、以下の SSH コマンドを実行します。ここで、**CONTROLLER_IP** は環境内の任意のコントローラーノードの IP アドレスに置き換えます。

```
$ ssh heat-admin@CONTROLLER_IP "sudo hiera -c /etc/puppet/hiera.yaml
pacemaker_short_bootstrap_node_name"
```

以下の例では、ブートストラップノードの値として **overcloud-controller-0** を使用します。これを実際のブートストラップノードの値に置き換えてください。

3. ブートストラップコントローラーノードをアップグレードします。

   a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-controller-0
```

   このコマンドにより、以下の操作が行われます。

   - Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
   - **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

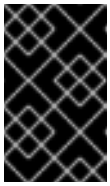
   このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

   b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-0
```

   このコマンドにより、以下のアクションが行われます。

   - Leapp によるオペレーティングシステムのアップグレードを実施する。
   - Leapp によるアップグレードの一部としてリブートを実施する。



重要

次のコマンドにより、コントロールプレーンで機能停止が生じます。これ以降の数ステップを実施している間は、標準的なオーバークラウド操作を行うことはできません。

   c. **system_upgrade_transfer_data** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
system_upgrade_transfer_data
```

このコマンドにより、最新バージョンのデータベースが既存のノードからブートストラップノードにコピーされます。

d. **nova_hybrid_state** タグを指定してアップグレードコマンドを実行し、**upgrade_steps_playbook.yaml** Playbook だけを実行します。

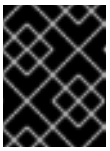
```
$ openstack overcloud upgrade run --stack STACK NAME --playbook
upgrade_steps_playbook.yaml --tags nova_hybrid_state --limit all
```

このコマンドにより、コンピュートノード上の一時的な 16.1 コンテナーが起動します。これにより、後のステップでコンピュートノードをアップグレードする際に、ワークロードの移行が円滑に行われます。

e. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。



重要

このコマンドの処理が完了すると、コントロールプレーンがアクティブになります。再び標準的なオーバークラウド操作を実施することができます。

4. 次のコントローラーノードをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags
ceph_systemd -e ceph_ansible_limit=overcloud-controller-1
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. 次のコントローラーノードで、**system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-controller-1
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。このノードに加えて、前のステップでアップグレードしたブートストラップノードを **--limit** オプションに含めます。

5. 最後のコントローラーノードをアップグレードします。

a. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e ceph_ansible_limit=overcloud-controller-2
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択したコントローラーノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

b. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit overcloud-controller-2
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

c. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-controller-0,overcloud-controller-1,overcloud-controller-2
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。**--limit** オプションにすべてのコントローラーノードを含めます。

21.3. ハイパーコンバージドインフラストラクチャー (HCI) を持つコンピュートノードのアップグレード

HCI コンピュートノードを OpenStack Platform 16.1 にアップグレードします。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. インスタンスを移行します。移行計画についての詳細は、『インスタンス＆イメージガイド』の「コンピュートノード間の仮想マシンインスタンスの移行」を参照してください。

3. Ceph MON サービスが含まれるノードにログインします。

4. OSD の除外およびリバランスを無効にします。

```
$ sudo docker ps | grep ceph-mon
$ sudo docker exec -it CONTAINER ceph osd set noout
$ sudo docker exec -it CONTAINER ceph osd set norebalance
```

CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

5. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。

6. **ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e
ceph_ansible_limit=overcloud-computehci-0
```

このコマンドにより、以下の操作が行われます。

- Podman 管理を使用するために、Ceph Storage コンテナーを制御する systemd ユニットを変更する。
- **ceph_ansible_limit** 変数を使用して、アクションを選択した Ceph Storage ノードに制限する。

このステップは、**leapp** によるアップグレードに向けて Ceph Storage サービスを準備するための予備的な処置です。

7. **system_upgrade** タグを指定してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-computehci-0
```

このコマンドにより、以下のアクションが行われます。

- Leapp によるオペレーティングシステムのアップグレードを実施する。
- Leapp によるアップグレードの一部としてリブートを実施する。

8. タグを指定せずにアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --limit overcloud-computehci-0
```

このコマンドにより、Red Hat OpenStack Platform のアップグレードが実施されます。

9. 複数のコンピュートノードを並行してアップグレードするには、**--limit** オプションをアップグレードするノードのコンマ区切りリストに設定します。まず、**ceph_systemd** タグを指定して外部アップグレードコマンドを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph_systemd -e
ceph_ansible_limit=overcloud-computehci-0
```

次に、**system_upgrade** タスクを実行します。

```
$ openstack overcloud upgrade run --stack STACK NAME --tags system_upgrade --limit
overcloud-computehci-0,overcloud-computehci-1,overcloud-computehci-2
```

続いて、標準の OpenStack サービスのアップグレードを実施します。

```
$ openstack overcloud upgrade run --stack STACK NAME  --limit overcloud-computehci-
0,overcloud-computehci-1,overcloud-computehci-2
```

10. すべての HCI ノードをアップグレードしたら、Ceph MON サービスが含まれるノードにログインします。

11. OSD の除外およびリバランスを有効にします。

```
$ sudo podman ps | grep ceph-mon
$ sudo podman exec -it CONTAINER ceph osd unset noout
$ sudo podman exec -it CONTAINER ceph osd unset norebalance
```

CONTAINER を Ceph MON を実行しているコンテナーの名前に置き換えます。

12. Ceph MON サービスが含まれるノードからログアウトし、アンダークラウドに戻ります。

21.4. オーバークラウドスタックの同期

アップグレードにおいては、スタックのリソース構造およびパラメーターが OpenStack Platform 16.1 の新規デプロイメントと整合するように、オーバークラウドスタックを更新する必要があります。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **containers-prepare-parameter.yaml** ファイルを編集し、以下のパラメーターおよびその値を削除します。

- **ceph3_namespace**
- **ceph3_tag**
- **ceph3_image**
- **name_prefix_stein**
- **name_suffix_stein**
- **namespace_stein**
- **tag_stein**

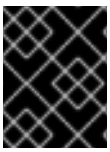
3. アップグレードの最終処理のコマンドを実行します。

```
$ openstack overcloud upgrade converge \
  --stack STACK NAME \
  --templates \
  -e ENVIRONMENT FILE
  …
  -e /home/stack/templates/upgrades-environment.yaml \
  -e /home/stack/templates/rhsm.yaml \
  -e /home/stack/containers-prepare-parameter.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs.yaml \
  …
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- アップグレード固有のパラメーターが含まれる環境ファイル (**upgrades-environment.yaml**) (**-e**)
- 登録およびサブスクリプションに関するパラメーターが含まれる環境ファイル (**rhsm.yaml**) (**-e**)
- 新しいコンテナーイメージの場所を定義した環境ファイル (**containers-prepare-parameter.yaml**) (**-e**)。多くの場合、アンダークラウドが使用する環境ファイルと同じファイルです。
- OVS との互換性を維持するための環境ファイル (**neutron-ovs.yaml**)
- デプロイメントに関連するカスタム設定環境ファイル (**-e**)
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) ファイルを指定します。
- 該当する場合は、**--networks-file** でコンポーザブルネットワーク (**network_data**) ファイルを指定します。
- カスタムのスタック名を使用する場合は、**--stack** オプションでその名前を渡します。

4. スタックの同期が完了するまで待ちます。



重要

これ以降のデプロイメント操作には、**upgrades-environment.yaml** ファイルは必要ありません。

パート IV. オーバークラウドアップグレードの最終処理

第22章 DIRECTOR でデプロイされた CEPH STORAGE クラスターの RED HAT CEPH STORAGE 4 へのアップグレード

director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、本項に記載する手順を完了する必要があります。

注記

外部の Ceph デプロイメントと共にアップグレードする場合は、本項に記載する手順を省略し、次のセクションに進む必要があります。

オーバークラウドをアップグレードしたら、director でデプロイされた Ceph Storage クラスターを Red Hat Ceph Storage クラスターバージョン 4 にアップグレードします。

22.1. CEPH-ANSIBLE のインストール

director を使用してデプロイされた Red Hat Ceph Storage クラスターがデプロイメントで使用される場合は、以下の手順を完了する必要があります。

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

手順

1. Ceph Tools リポジトリーを有効にします。

```
[stack@director ~]$ sudo subscription-manager repos --enable=rhceph-4-tools-for-rhel-8-x86_64-rpms
```

2. **ceph-ansible** パッケージをインストールします。

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

22.2. CEPH STORAGE 4 へのアップグレード

Ceph Storage ノードをバージョン 3 からバージョン 4 にアップグレードします。

注記

デフォルトのスタック名 (**overcloud**) を使用していない場合は、**--stack STACK NAME** オプションでスタック名を設定します。**STACK NAME** は実際のスタック名に置き換えます。

手順

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

```
$ source ~/stackrc
```

2. **ceph** タグを指定して、Ceph Storage の外部アップグレードプロセスを実行します。

```
$ openstack overcloud external-upgrade run --stack STACK NAME --tags ceph
```

3. Ceph Storage のアップグレードが完了するまで待ちます。

第23章 FILESTORE から BLUESTORE への OSD の移行

アップグレードプロセスを完了して検証を行ったら、FileStore OSD を BlueStore に移行する必要があります。1 台ずつノードの移行作業を完了していく必要があります。以下の手順では、移行を完了するのに **ceph-ansible** を使用しています。以下の手順は、Ceph クラスターが director によりデプロイされている場合にのみ適用されます。

23.1. クラスターが FILESTORE を実行している (したがって移行が必要である) ことの確認

手順

1. コントローラーノードやスタンドアロンの Ceph MON ノードなど、Ceph MON コンテナーを持つノードに **heat-admin** ユーザーとしてログインします。たとえば、標準のオーバークラウドデプロイメントでは、**overcloud-controller-1** が Ceph MON コンテナーを使用します。

2. OSD が使用しているドライバーを確認するために、Ceph クラスターに対してクエリーを行います。

```
[heat-admin@overcloud-controller-1 ~]$ sudo -i
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph -f json osd metadata" | jq -c 'sort_by(.hostname) | .[] | ["host", .hostname, "osd_id", .id,
"objectstore", .osd_objectstore]'
[root@overcloud-controller-1 ~]#
```

3. いずれかの行が **"objectstore": "filestore"** を返す場合は、そのノードで OSD を移行する必要があります。

警告

移行の時間はクラスターのサイズによって異なります。非常に大きなクラスターの場合、移行時間はそのクラスター内の OSD の数および保存されるデータ量に比例します。お使いの環境が混合アーキテクチャーのシナリオにならないように、できるだけ早く移行を完了してください。混合アーキテクチャーのシナリオでは、パフォーマンスに影響が及ぶ可能性があります。

警告

Red Hat Ceph Storage (RHCS) 4 バージョンの **ceph-ansible** で FileStore ベースの OSD を管理する構成はサポートされていないため、スタックの更新を実行する前に移行を完了してください。

23.2. FILESTORE から BLUESTORE への OSD の移行

FileStore から BlueStore に移行するのに、director は Ansible を使用してノード上のすべての OSD を削除して再作成します。director は、移行プロセスの前に容量の確認を行います。最後に、director は BlueStore バックエンドを使用して OSD を再デプロイします。

前提条件

- 正常な稼働中の Red Hat Ceph Storage (RHCS) 4 クラスター。コントローラーノードまたはスタンドアロンの Ceph MON ノード上の ceph MON コンテナーにおいて、以下のコマンドを入力してクラスターを確認することができます。

```
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph  -s"
```

手順

1. **CephAnsibleDisksConfig** パラメーターの **osd_objectstore** がデフォルトの **filestore** に設定されていないことを確認します。いずれかのカスタム heat 環境ファイルに **osd_objectstore** パラメーターが存在する場合は、明示的に値を **bluestore** と定義するか、そのパラメーターを削除する必要があります。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
      - /dev/sdb
      - /dev/sdc
      - /dev/sdd
    osd_scenario: lvm
   osd_objectstore: bluestore
```

注記

ジャーナル等に対する特定の FileStore 設定がある場合は、適切に設定を更新するようにしてください。高度な設定についての詳しい情報は、**『コンテナー化された Red Hat Ceph を持つオーバークラウドのデプロイ』**の「Ceph Storage ノードのディスクレイアウトのマッピング」を参照してください。

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

3. **ceph_fstobs** タグを指定して **openstack overcloud external-upgrade run** コマンドを入力します。**<NODE_NAME>** をアップグレードする Ceph OSD ノードの名前に置き換えてください。**openstack server list** コマンドを使用してノード名を把握することができます。

```
[stack@undercloud ~] $ openstack overcloud external-upgrade run --tags ceph_fstobs -e
ceph_ansible_limit=<NODE_NAME> | tee oc-fstobs.log
```

4. 次の OSD の移行に進む前に、Ceph MON サービスが実行されているノードにログインして Ceph クラスターにクエリーを行い、ノードの OSD のステータスを確認してその OSD が移行されていることを確認します。

```
[heat-admin@overcloud-controller-1 ~]$ sudo -i
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph -f json osd metadata" | jq -c '.[] | select(.hostname == "<NODE_NAME>") | ["host",
```

```
.hostname, "osd_id", .id, "objectstore", .osd_objectstore]'
[root@overcloud-controller-1 ~]# exit
```

<NODE_NAME> を移行したノードの名前に置き換えます。出力結果を確認し、OSD が BlueStore を使用することが示されていれば、移行は正常に完了しています。

5. 特定の OSD に関する追加情報を表示するには、以下のコマンドを入力します。

```
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph osd metadata <ID>"
```

<ID> はクエリーを行う OSD の ID に置き換えてください。

6. 次のノードの移行を開始する前に、クラスターが同期するのを待ちます。

```
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph  -s"
```

7. 上記コマンドの出力を確認し、クラスターの健全性が **HEALTH_OK** で PG が **active+clean** の状態にあることを確認します。

8. ストレージクラスター内の各ノードに対して移行プロセスを繰り返します。

FileStore から BlueStore への移行に関する詳細は、Red Hat Ceph Storage の**『管理ガイド』**の「BlueStore」を参照してください。

23.3. FILESTORE から BLUESTORE への移行の確認

OSD のステータスを確認して、移行が正常に完了していることを確認することができます。

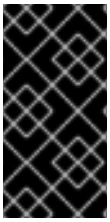
手順

1. いずれかのコントローラーノードがホストする ceph-mon コンテナーに **heat-admin** ユーザーとしてログインします。

2. Ceph クラスターに対してクエリーを行います。

```
[heat-admin@overcloud-controller-1 ~]$ sudo -i
[root@overcloud-controller-1 ~]# podman exec -it ceph-mon-overcloud-controller-1 sh -c
"ceph -f json osd metadata" | jq -c 'sort_by(.hostname) | .[] | ["host", .hostname, "osd_id", .id,
"objectstore", .osd_objectstore]'
[root@overcloud-controller-1 ~]#
```

設定を確認し、クラスター全体のすべての OSD が BlueStore を使用することが示されていれば、移行は正常に完了しています。



重要

推奨されるベストプラクティスは、べき等性を持つスタック更新を実行し、設定の定義と実際の設定が一致するようにすることです。スタック更新の時間はシステムのサイズによって異なります。したがって、ダウンタイムを短縮するため、メンテナンス期間中に移行を完了するよう計画することができます。

第24章 アップグレード後操作の実施

オーバークラウドのアップグレードが完了したら、アップグレード後の設定を実施して、環境が完全にサポートされ、これ以降の操作を行う準備が整っている状態にする必要があります。

24.1. アンダークラウドからの不要なパッケージの削除

Leapp によるアップグレード後に、いくつかの不要なパッケージがアンダークラウドに残ります。これらのパッケージを削除します。

手順

1. 不要なパッケージを削除します。

```
$ sudo dnf -y remove --exclude=python-pycadf-common python2*
```

24.2. アップグレード後の機能検証

post-upgrade 検証グループを実行し、アップグレード後の機能を確認します。

手順

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

```
$ source ~/stackrc
```

2. **--group post-upgrade** オプションを指定して **openstack tripleo validator run** コマンドを実行します。

```
$ openstack tripleo validator run --group post-upgrade
```

3. 検証レポートの結果を確認します。特定の検証からの詳細出力を表示するには、レポートからの特定検証の UUID を指定して **openstack tripleo validator show run** コマンドを実行します。

```
$ openstack tripleo validator show run <UUID>
```

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。

24.3. オーバークラウドイメージのアップグレード

現在のオーバークラウドイメージを新しいバージョンに置き換える必要があります。新しいイメージにより、director は最新バージョンの OpenStack Platform ソフトウェアを使用してノードのイントロスペクションとプロビジョニングを行うことができるようになります。

前提条件

- アンダークラウドが最新バージョンにアップグレードされていること

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 **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

3. オーバークラウドの QCOW2 アーカイブが含まれるパッケージをインストールします。

```
$ sudo dnf install rhosp-director-images rhosp-director-images-ipa
```

4. **stack** ユーザーの **images** ディレクトリー (**/home/stack/images**) から既存のイメージを削除します。

```
$ rm -rf ~/images/*
```

5. アーカイブを展開します。

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-16.1.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-16.1.tar; do tar -xvf $i; done
$ cd ~
```

6. director に最新のイメージをインポートします。

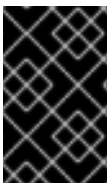
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

7. ノードが新しいイメージを使用するように設定します。

```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

重要

オーバークラウドノードをデプロイする際には、オーバークラウドイメージのバージョンが該当する heat テンプレートバージョンに対応している状態にします。たとえば、OpenStack Platform 16.1 のイメージは、OpenStack Platform 16.1 の heat テンプレートだけに使用してください。



重要

新しい **overcloud-full** イメージは、古い **overcloud-full** イメージを置き換えます。古いイメージに変更を加えた場合、特に今後新規ノードをデプロイする場合には、新しいイメージで変更を繰り返す必要があります。

24.4. CPU ピニングパラメーターの更新

Red Hat OpenStack Platform 16.1 では、CPU ピニングに新たなパラメーターが使用されます。

NovaComputeCpuDedicatedSet

専用の (ピニングされた) CPU を設定します。

NovaComputeCpuSharedSet

共有の (ピニングされていない) CPU を設定します。

Red Hat OpenStack Platform 16.1 へのアップグレードが完了したら、CPU ピニングの設定を **NovaVcpuPinSet** パラメーターから **NovaComputeCpuDedicatedSet** および **NovaComputeCpuSharedSet** パラメーターに移行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. コンピュートノードが同時マルチスレッド (SMT) をサポートするが **hw:cpu_thread_policy=isolate** ポリシーでインスタンスを作成している場合は、以下のオプションのいずれかを実施する必要があります。
   - **hw:cpu_thread_policy** スレッドポリシーの設定を解除し、インスタンスのサイズを変更する。
     i. source コマンドでオーバークラウドの認証ファイルを読み込みます。

     ```
     $ source ~/overcloudrc
     ```

     ii. フレーバーの **hw:cpu_thread_policy** 属性の設定を解除します。

     ```
     (overcloud) $ openstack flavor unset --property hw:cpu_thread_policy <flavor>
     ```

     注記

     - **hw:cpu_thread_policy** 属性の設定を解除すると、ポリシーがデフォルトの **prefer** ポリシーに設定されます。これにより、インスタンスは SMT 対応のコンピュートノードを使用するように設定されます (利用可能な場合)。**hw:cpu_thread_policy** 属性を **require** に設定することもできます。これにより、SMT 対応のコンピュートノードに対するハード要件が設定されます。
     - コンピュートノードに SMT アーキテクチャーがない場合や、スレッドシブリングが利用可能な CPU コアが十分にない場合には、スケジューリングが失敗します。これを回避するには、**hw:cpu_thread_policy** を **require** ではなく **prefer** に設定します。デフォルトの **prefer** ポリシーに設定すると、スレッドシブリングが利用可能な場合に使用されるようになります。
     - **hw:cpu_thread_policy=isolate** を使用する場合は、SMT を無効にするか、SMT をサポートしないプラットフォームを使用する必要があります。

     iii. 新しいスレッドポリシーを使用するようにインスタンスを変換します。

     ```
     (overcloud) $ openstack server resize --flavor <flavor> <server>
     (overcloud) $ openstack server resize confirm <server>
     ```

     hw:cpu_thread_policy=isolated ポリシーを使用するすべての固定されたインスタンスに対して、このステップを繰り返します。

- コンピュートノードからインスタンスを移行して、コンピュートノードの SMT を無効にする。

  i. source コマンドでオーバークラウドの認証ファイルを読み込みます。

  ```
  $ source ~/overcloudrc
  ```

  ii. コンピュートノードが新しい仮想マシンを受け入れるのを無効にします。

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

  iii. コンピュートノードからすべてのインスタンスを移行します。インスタンスの移行についての詳細は、「コンピュートノード間の仮想マシンインスタンスの移行」を参照してください。

  iv. コンピュートノードをリブートし、コンピュートノードの BIOS で SMT を無効にします。

  v. コンピュートノードをブートします。

  vi. コンピュートノードを再度有効にします。

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

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

```
$ source ~/stackrc
```

4. **NovaVcpuPinSet** パラメーターが含まれる環境ファイルを編集します。

5. CPU ピニング設定を **NovaVcpuPinSet** パラメーターから **NovaComputeCpuDedicatedSet** および **NovaComputeCpuSharedSet** に移行します。

   - これまでピニングされたインスタンス用に使用されていたホストの場合には、**NovaVcpuPinSet** の値を **NovaComputeCpuDedicatedSet** に変更します。
   - これまでピニングされていないインスタンス用に使用されていたホストの場合には、**NovaVcpuPinSet** の値を **NovaComputeCpuSharedSet** に変更します。
   - NovaVcpuPinSet の値が設定されていない場合には、ノードでホストするインスタンスの種別に応じて、コンピュートノードのすべてのコアを **NovaComputeCpuDedicatedSet** または **NovaComputeCpuSharedSet** のどちらかに割り当てる必要があります。

   たとえば、以前の環境ファイルに以下のピニング設定が定義されていたとします。

```
parameter_defaults:
  ...
  NovaVcpuPinSet: 1,2,3,5,6,7
  ...
```

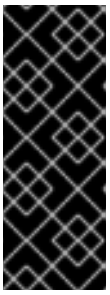
   設定をピニング設定に移行するには、**NovaComputeCpuDedicatedSet** パラメーターを設定し、**NovaVcpuPinSet** パラメーターの設定を解除します。

```
parameter_defaults:
```

```
...
NovaComputeCpuDedicatedSet: 1,2,3,5,6,7
NovaVcpuPinSet: ""
...
```

設定をピニングしない設定に移行するには、**NovaComputeCpuSharedSet** パラメーターを設定し、**NovaVcpuPinSet** パラメーターの設定を解除します。

```
parameter_defaults:
  ...
  NovaComputeCpuSharedSet: 1,2,3,5,6,7
  NovaVcpuPinSet: ""
  ...
```



重要

NovaComputeCpuDedicatedSet または **NovaComputeCpuSharedSet** のいずれかの設定が、**NovaVcpuPinSet** で定義される設定と一致するようにします。**NovaComputeCpuDedicatedSet** または**NovaComputeCpuSharedSet** のいずれかの設定を変更する、またはその両方を設定するには、設定を更新する前にピニング設定のコンピュートノードが 1 つのインスタンスも実行していないようにします。

6. ファイルを保存します。
7. デプロイメントコマンドを実行して、新しい CPU ピニングパラメーターでオーバークラウドを更新します。

```
(undercloud) $ openstack overcloud upgrade converge \
  --stack _STACK NAME_ \
  --templates \
  ...
  -e /home/stack/templates/<compute_environment_file>.yaml
  ...
```

関連資料

- コンピュートノードでの CPU ピニングの設定

24.5. OPEN VIRTUAL NETWORK (OVN) への移行

Open Virtual Network (OVN) は、インスタンスにネットワークサービスを提供する、Open vSwitch をベースとするソフトウェア定義ネットワーク (SDN) ソリューションです。OVN はプラットフォームに依存しない、OpenStack Networking API の完全なサポートを提供します。OVN を使用することで、ゲストインスタンスのグループを L2 または L3 プライベートネットワークにプログラムで接続することができます。OVN は、Red Hat の他のプラットフォームやソリューションに拡張することのできる仮想ネットワークの標準的な方法を採用しています。

Red Hat Enterprise Linux 16.1 にアップグレードしたら、オーバークラウド上の OpenStack Networking (neutron) サービスを ML2/OVS ドライバーから ML2/OVN ドライバーに移行します。ML2/OVN への移行の詳細は、「ML2/OVS から ML2/OVN への移行」 を参照してください。

パート V. トラブルシューティング

第25章 アップグレードに関する問題のトラブルシューティング

アップグレードプロセス中に問題が発生した場合は、本セクションのアドバイスを参照してください。

25.1. 環境ファイルの修正

カスタム環境ファイルのパラメーターを誤って設定していた場合は、環境ファイルを修正して、アップグレード中いつでも **openstack overcloud upgrade prepare** コマンドを実行することができます。このコマンドにより、新しいバージョンのオーバークラウドプランが director にアップロードされ、これにより **config-download** Playbook の新しいセットが生成されます。

upgrades-environment.yaml ファイルのリポジトリー名が間違っている例を以下に示します。

```
parameter_defaults:
  UpgradeLeappEnabled: true
  UpgradeLeappCommandOptions: "--enablerepo rhel-7-for-x86_64-baseos-eus-rpms --enablerepo
rhel-8-for-x86_64-appstream-eus-rpms --enablerepo fast-datapath-for-rhel-8-x86_64-rpms"
  CephAnsibleRepo: rhceph-4-tools-for-rhel-8-x86_64-rpms
```

この間違いにより、コントローラーノードの Leapp アップグレード時に問題が発生します。この問題を正すには、間違いを修正して **openstack overcloud upgrade prepare** コマンドを実行します。

手順

1. ファイルの間違いを修正します。

```
parameter_defaults:
  UpgradeLeappEnabled: true
  UpgradeLeappCommandOptions: "--enablerepo rhel-8-for-x86_64-baseos-eus-rpms --
enablerepo rhel-8-for-x86_64-appstream-eus-rpms --enablerepo fast-datapath-for-rhel-8-
x86_64-rpms"
  CephAnsibleRepo: rhceph-4-tools-for-rhel-8-x86_64-rpms
```

2. 修正したファイルでアップグレードの準備コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
    --stack STACK NAME \
    --templates \
    -e ENVIRONMENT FILE
    …
    -e /home/stack/templates/upgrades-environment.yaml \
    …
```

   オーバークラウドスタックの更新が完了するまで待ちます。

3. 失敗したアップグレードのステップから操作を続行します。