Red Hat OpenStack Platform 13

Red Hat OpenStack Platform のアップグレード

Red Hat OpenStack Platform 環境のアップグレード

Last Updated: 2023-04-10

Red Hat OpenStack Platform 13 Red Hat OpenStack Platform のアップグレード

Red Hat OpenStack Platform 環境のアップグレード

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ドキュメントでは、Red Hat OpenStack Platform 12 (Pike) から 13 (Queens) にアップグレードする複数の異なる方法について説明します。これらの方法は、アップグレード元およびアップグレード先のいずれも Red Hat Enterprise Linux 7 をベースにインストールされた OpenStack デプロイメントであることを前提としています。

目次

第1章 はじめに

本書には、Red Hat OpenStack Platform 環境を最新のメジャーバージョンにアップグレードし、そのバージョンのマイナーリリースで最新状態に維持するために役立つワークフローを記載しています。

本ガイドは、以下のバージョンのアップグレードパスを提供します。

古いオーバークラウドバージョン	新しいオーバークラウドバージョン
Red Hat OpenStack Platform 12	Red Hat OpenStack Platform 13

1.1. ワークフローの概要

以下の表には、アップグレードのプロセスに必要なステップの概要をまとめています。

ステップ	説明
環境の準備	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ラーノードのデータベースおよび設定のバックアップを実行します。最新のマイナーリリースに更新します。環境を検証します。
アンダークラウドのアップグレード	アンダークラウドを OpenStack Platform 12 から OpenStack Platform 13 にアップグレードします。
コンテナーイメージの取得	OpenStack Platform 13 のサービス用のコンテナーイメージの場所が記載された環境ファイルを作成します。
オーバークラウドの準備	オーバークラウドの設定ファイルを OpenStack Platform 13 に移行するための適切なステップを実行します。
コントローラーノードのアップグレード	全コントローラーノードを同時に OpenStack Platform 13 にアップグレードします。
コンピュートノードのアップグレード	選択したコンピュートノードでアップグレードをテストします。テストが成功したら、全コンピュートノードをアップグレードします。
Ceph Storage ノードのアップグレード	全 Ceph Storage ノードをアップグレードします。これには、Red Hat Ceph Storage 3 のコンテナー化されたバージョンへのアップグレードも含まれます。
アップグレードの最終段階	コンバージェンスのコマンドを実行して、オーバークラウドスタックをリフレッシュします。

1.2. リポジトリー

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

表1.1 OpenStack Platform リポジトリー

名前	リポジトリー	要件の説明
Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rpms**	x86_64 システム用ベースオペレーティングシステムのリポジトリー
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	**rhel-7-server-extras-rpms**	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	**rhel-7-server-rh-common-rpms**	Red Hat OpenStack Platform のデプロイと設定用のツールが含まれます。
Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs) x86_64	**rhel-7-server-satellite-tools-6.3-rpms**	Red Hat Satellite Server 6 でのホスト管理ツール。これより新しいバージョンの Satellite Tools リポジトリーを使用すると、アンダークラウドのインストールが失敗する可能性があることに注意してください。
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	**rhel-ha-for-rhel-7-server-rpms**	Red Hat Enterprise Linux の高可用性ツール。コントローラーノードの高可用性に使用します。
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-rpms**	Red Hat OpenStack Platform のコアリポジトリー。Red Hat OpenStack Platform director のパッケージも含まれます。
Red Hat Ceph Storage OSD 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-osd-rpms**	(Ceph Storage ノード向け) Ceph Storage Object Storage デーモンのリポジトリー。Ceph Storage ノードにインストールします。
Red Hat Ceph Storage MON 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-mon-rpms**	(Ceph Storage ノード向け) Ceph Storage Monitor デーモンのリポジトリー。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。

名前	リポジトリー	要件の説明
Red Hat Ceph Storage Tools 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-tools-rpms**	Ceph Storage クラスターと通信するためのノード用のツールを提供します。Ceph Storage クラスターと共にオーバークラウドをデプロイする場合や、オーバークラウドを既存の Ceph Storage クラスターと統合する場合に、すべてのノードでこのリポジトリーを有効にします。
Red Hat OpenStack 13 Director Deployment Tools for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-deployment-tools-rpms**	(Ceph Storage ノード向け) 現在のバージョンの Red Hat OpenStack Platform director に対応したデプロイメントツールのセットを提供します。アクティブな Red Hat OpenStack Platform サブスクリプションがない Ceph ノードにインストールされます。
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	**rhel-7-server-nfv-rpms**	NFV 向けのリアルタイム KVM (RT-KVM) のリポジトリー。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。このリポジトリーは、RT-KVM 対象の全コンピュートノードで有効化する必要があります。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 OpenStack Platform 13 Extended Life Cycle Support for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-els-rpms**	2021 年 6 月 26 日に開始した延長ライフサイクルサポートの更新が含まれています。このリポジトリーを利用するには、"Entitlement for OpenStack 13 Platform Extended Life Cycle Support" (MCT3637) が必要です。

注記

ネットワークがオフラインの Red Hat OpenStack Platform 環境用リポジトリーを設定するには、Red Hat カスタマーポータルで オフライン環境で Red Hat OpenStack Platform Director を設定する のアーティクルを参照してください。

1.3. アップグレードを開始する前に

- アップグレードを実施する前に、ハードウェアに対するファームウェアの更新をすべて適用します。

- 更新時に Open vSwitch (OVS) のメジャーバージョンが変更されると (たとえば 2.9 から 2.11 に)、director はユーザーがカスタマイズした設定ファイルの名前に .rpmsave 拡張子を付けて変更し、デフォルトの OVS 設定をインストールします。
  以前の OVS カスタマイズを維持するには、名前が変更されたファイルに含まれる変更を手動で再適用する必要があります (例: /etc/logrotate.d/openvswitch の logrotate の設定)。この 2 ステップの更新方法により、RPM パッケージの自動更新によってトリガーされるデータプレーンの中断が回避されます。

第2章 OPENSTACK PLATFORM のアップグレードの準備

このプロセスでは、完全な更新に向けて OpenStack Platform 環境を準備します。これには、以下のステップを伴います。

- アンダークラウドのパッケージを更新し、アップグレードコマンドを実行します。
- 新しいカーネルまたはシステムパッケージがインストールされた場合には、アンダークラウドをリブートする。
- overcloud upgrade コマンドでオーバークラウドを更新する。
- 新しいカーネルまたはシステムパッケージがインストールされた場合には、オーバークラウドノードをリブートする。
- アンダークラウドとオーバークラウドの両方で検証のチェックを実行する。

これらの手順により、OpenStack Platform 環境は、アップグレードを開始する前に、最適な状態となります。

2.1. 現在のアンダークラウドおよびオーバークラウドのマイナー更新の実行

OpenStack Platform 13 にアップグレードする前に、既存の環境 (この場合はバージョン 12) を最新のマイナーバージョンに更新する必要があります。既存の OpenStack Platform 12 環境を更新するには、アンダークラウドの最新状態の維持 を参照してください。

2.2. コントローラーノードおよびコンポーザブルノードのリブート

以下の手順では、コントローラーノードと、コンポーザブルロールをベースとするスタンドアロンのノードをリブートします。これには、コンピュートノードと Ceph Storage ノードは含まれません。

手順

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

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

3. ノードをリブートします。

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

4. ノードがブートするまで待ちます。
5. サービスを確認します。以下に例を示します。
   a. ノードが Pacemaker サービスを使用している場合には、ノードがクラスターに再度加わったかどうかを確認します。

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

   b. ノードが Systemd サービスを使用している場合には、すべてのサービスが有効化されていることを確認します。

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

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

2.3. CEPH STORAGE (OSD) クラスターのリブート

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

手順

1. Ceph MON またはコントローラーノードにログインして、Ceph Storage Cluster のリバランスを一時的に無効にします。

```
$ sudo ceph osd set noout
$ sudo ceph osd set norebalance
```

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

3. ノードをリブートします。

```
$ sudo reboot
```

4. ノードがブートするまで待ちます。

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

```
$ sudo ceph -s
```

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

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

7. 完了したら、Ceph MON またはコントローラーノードにログインして、クラスターのリバランスを再度有効にします。

```
$ sudo ceph osd unset noout
$ sudo ceph osd unset norebalance
```

8. 最終のステータスチェックを実行して、クラスターが **HEALTH_OK** を報告していることを確認します。

```
$ sudo ceph status
```

2.4. コンピュートノードのリブート

コンピュートノードをリブートするには、以下のワークフローを実施します。

- リブートするコンピュートノードを選択して無効にし、新規インスタンスをプロビジョニングしないようにする。
- インスタンスのダウンタイムを最小限に抑えるために、インスタンスを別のコンピュートノードに移行する。
- 空のコンピュートノードをリブートして有効にする。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. 再起動するコンピュートノードを特定するには、すべてのコンピュートノードを一覧表示します。

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

3. オーバークラウドから、コンピュートノードを選択し、そのノードを無効にします。

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

4. コンピュートノード上の全インスタンスを一覧表示します。

```
(overcloud) $ openstack server list --host <hostname> --all-projects
```

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

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

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

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

9. コンピュートノードが有効化されていることを確認します。

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

2.5. アンダークラウドの検証

アンダークラウドの機能を確認するステップを以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

2. エラーが発生している Systemd サービスがあるかどうかを確認します。

```
(undercloud) $ sudo systemctl list-units --state=failed 'openstack*' 'neutron*' 'httpd' 'docker'
```

3. アンダークラウドの空き領域を確認します。

```
(undercloud) $ df -h
```

アンダークラウドの要件 を元に、十分な空き容量があるかどうかを判断します。

4. アンダークラウド上に NTP をインストールしている場合には、クロックが同期されていることを確認します。

```
(undercloud) $ sudo ntpstat
```

5. アンダークラウドのネットワークサービスを確認します。

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

全エージェントが **Alive** で、それらの状態が **UP** である必要があります。

6. アンダークラウドの Compute サービスを確認します。

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

全エージェントのステータスが **enabled** で、状態が **up** である必要があります。

関連情報

- OpenStack Orchestration (heat) のデータベースで削除済みとマークされている stack のエントリーを完全削除する方法は、How I can remove old data from my heat database from my Director node のソリューションに記載されています。

2.6. コンテナー化されたオーバークラウドの検証

コンテナー化されたオーバークラウドの機能を確認するステップを以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

2. ベアメタルノードのステータスを確認します。

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

全ノードの電源状態が有効で (**on**)、かつメンテナーンスモードが **false** である必要があります。

3. エラーが発生している Systemd サービスがあるかどうかを確認します。

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo systemctl list-units --state=failed
'openstack*' 'neutron*' 'httpd' 'docker' 'ceph*'" ; done
```

4. エラーが発生しているコンテナー化されたサービスがあるかどうかを確認します。

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'exited=1' --all" ; done
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'status=dead' -f
'status=restarting'" ; done
```

5. 全サービスへの HAProxy 接続をチェックします。コントロールプレーンの仮想 IP アドレスと **haproxy.stats** サービスの認証情報を取得します。

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -
d= -f2); ssh heat-admin@$NODE sudo 'grep "listen haproxy.stats" -A 6 /var/lib/config-
data/puppet-generated/haproxy/etc/haproxy/haproxy.cfg'
```

以下の cURL 要求でそれらの情報を使用します。

```
(undercloud) $ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi
"(frontend|backend)" | cut -d, -f 1,2,18,37,57 | column -s, -t
```

<PASSWORD> および **<IP ADDRESS>** の詳細を、**haproxy.stats** サービスからの実際の詳細に置き換えます。その結果表示される一覧には、各ノード上の OpenStack Platform サービスとそれらの接続ステータスが表示されます。

注記

ノードが Redis サービスを実行している場合、1 つのノードだけがそのサービスを **ON** のステータスで表示します。これは、Redis がアクティブ/パッシブのサービスであり、同時に複数のノードでは実行されないためです。

6. オーバークラウドデータベースのレプリケーションの正常性を確認します。

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks |
cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec
clustercheck clustercheck" ; done
```

7. RabbitMQ クラスターの正常性を確認します。

```
(undercloud) $ for NODE in $(openstack server list --name controller -f value -c Networks |
cut -d= -f2); do echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker exec $(ssh
heat-admin@$NODE "sudo docker ps -f 'name=.*rabbitmq.*' -q") rabbitmqctl
node_health_check" ; done
```

8. Pacemaker リソースの正常性を確認します。

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -
d= -f2); ssh heat-admin@$NODE "sudo pcs status"
```

–

以下の点を確認します。

- 全クラスターノードが **online** であること
- いずれのクラスターノード上でも **stopped** のリソースがないこと
- pacemaker で **failed** のアクションがないこと

9. 各オーバークラウドノードでディスク領域を確認します。

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo df -h --output=source,fstype,avail -
x overlay -x tmpfs -x devtmpfs" ; done
```

10. オーバークラウドの Ceph Storage クラスターの正常性を確認します。以下のコマンドを使用すると、コントローラーノード上で **ceph** ツールが実行されて、クラスターをチェックします。

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -
d= -f2); ssh heat-admin@$NODE "sudo ceph -s"
```

11. Ceph Storage OSD に空き領域があるかどうかを確認します。以下のコマンドを使用すると、コントローラーノード上で **ceph** ツールが実行され、空き領域をチェックします。

```
(undercloud) $ NODE=$(openstack server list --name controller-0 -f value -c Networks | cut -
d= -f2); ssh heat-admin@$NODE "sudo ceph df"
```

12. オーバークラウドノードでクロックが同期されていることを確認します。

```
(undercloud) $ for NODE in $(openstack server list -f value -c Networks | cut -d= -f2); do
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo ntpstat" ; done
```

13. オーバークラウドのアクセス情報を読み込みます。

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

14. オーバークラウドのネットワークサービスを確認します。

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

全エージェントが **Alive** で、それらの状態が **UP** である必要があります。

15. オーバークラウドの Compute サービスを確認します。

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

全エージェントのステータスが **enabled** で、状態が **up** である必要があります。

16. オーバークラウドのボリュームサービスを確認します。

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

全エージェントのステータスが **enabled** で、状態が **up** である必要があります。

関連情報

- How can I verify my OpenStack environment is deployed with Red Hat recommended configurations? の記事を参照してください。この記事には、Red Hat OpenStack Platform 環境を確認して、Red Hat の推奨値に合わせて設定を調整する方法が記載されています。

第3章 アンダークラウドのアップグレード

以下の手順では、アンダークラウドと、そのオーバークラウドのイメージを **Red Hat OpenStack Platform 13** にアップグレードします。

3.1. アンダークラウドを OPENSTACK PLATFORM 13 にアップグレードする手順

この手順では、アンダークラウドのツールセットとコア Heat テンプレートコレクションを **OpenStack Platform 13** リリースにアップグレードします。

手順

1. **stack** ユーザーとして director にログインします。
2. 現在設定されている OpenStack Platform リポジトリーを無効にします。

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
```

3. RHEL のバージョンを RHEL 7.9 に設定します。

```
$ sudo subscription-manager release --set=7.9
```

4. 新しい OpenStack Platform リポジトリーを有効にします。

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
```

5. オーバークラウドのベースイメージへの更新を再度有効にします。

```
$ sudo yum-config-manager --setopt=exclude= --save
```

6. **yum** コマンドを実行して、director の主要なパッケージをアップグレードします。

```
$ sudo yum update -y python-tripleoclient
```

7. 以下のコマンドを実行してアンダークラウドをアップグレードします。

```
$ openstack undercloud upgrade
```

8. アンダークラウドのアップグレードプロセスが完了するまで待ちます。
9. アンダークラウドをリブートして、オペレーティングシステムのカーネルとその他のシステムパッケージを更新します。

```
$ sudo reboot
```

10. ノードがブートするまで待ちます。

アンダークラウドが **OpenStack Platform 13** リリースにアップグレードされました。

3.2. 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. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

2. **stack** ユーザーのホーム下の **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/*
```

3. アーカイブを展開します。

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-13.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-13.0.tar; do tar -xvf $i; done
$ cd ~
```

4. director に最新のイメージをインポートします。

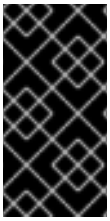
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

5. 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)
```

6. 新規イメージが存在することを確認します。

```
$ openstack image list
$ ls -l /httpboot
```



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、その heat テンプレートバージョンに対応していることを確認してください。たとえば、OpenStack Platform 13 の Heat テンプレートには、OpenStack Platform 13 のイメージのみを使用してください。

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、特に新規ノードを今後デプロイする必要がある場合には、新しいイメージで変更を繰り返す必要があります。

3.3. 以前のテンプレートバージョンの比較

アップグレードプロセスにより、最新のオーバークラウドバージョンに対応したコア Heat テンプレートの新しいセットがインストールされます。Red Hat OpenStack Platform のリポジトリーには、**openstack-tripleo-heat-templates-compat** パッケージ内のコアテンプレートコレクションの以前のバージョンが維持されています。以下の手順では、オーバークラウドのアップグレードに影響する可能性のある変更点を確認することができるように、それらのバージョンを比較する方法について説明します。

手順

1. **openstack-tripleo-heat-templates-compat** パッケージをインストールします。

```
$ sudo yum install openstack-tripleo-heat-templates-compat
```

これにより、Heat テンプレートコレクションの **compat** ディレクトリー (**/usr/share/openstack-tripleo-heat-templates/compat**) に以前のテンプレートがインストールされ、以前のバージョン (**pike**) から命名された **compat** へのリンクが作成されます。これらのテンプレートは、アップグレードされた director との後方互換性があるので、最新バージョンの director を使用して以前のバージョンのオーバークラウドをインストールすることができます。

2. コア Heat テンプレートの一時的なコピーを作成します。

```
$ cp -a /usr/share/openstack-tripleo-heat-templates /tmp/osp13
```

3. 以前のバージョンをそれ独自のディレクトリーに移動します。

```
$ mv /tmp/osp13/compat /tmp/osp12
```

4. 両ディレクトリーのコンテンツに対して **diff** を実行します。

```
$ diff -urN /tmp/osp12 /tmp/osp13
```

このコマンドにより、バージョン間におけるコアテンプレートの変更が表示されます。この内容を確認すると、オーバークラウドのアップグレード中にどのような動作が行われるかがわかります。

3.4. アンダークラウドアップグレード後の注意事項

- **stack** ユーザーのホームディレクトリーでコアテンプレートのローカルセットを使用している場合には、カスタムのコア Heat テンプレートの使用 に記載の推奨ワークフローを使用して、必ずテンプレートを更新してください。オーバークラウドをアップグレードする前に、ローカルコピーを更新する必要があります。

3.5. 次のステップ

アンダークラウドのアップグレードが完了しました。これで、オーバークラウドをアップグレードに向けて準備することができます。

第4章 コンテナーイメージのソースの設定

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

本ガイドには、オーバークラウドを設定してレジストリーを使用するさまざまなユースケースを記載しています。これらのユースケースのいずれかを試みる前に、イメージ準備コマンドの使用方法に習熟しておくことを推奨します。詳しくは、「コンテナーイメージの準備コマンドの使用方法」を参照してください。

4.1. レジストリーメソッド

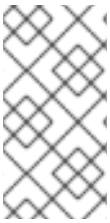
Red Hat OpenStack Platform では、以下のレジストリータイプがサポートされています。

リモートレジストリー

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

ローカルレジストリー

アンダークラウドは、**docker-distribution** サービスを使用してレジストリーとして機能します。これにより、director は **registry.redhat.io** からプルしたイメージを同期し、それを **docker-distribution** レジストリーにプッシュすることができます。オーバークラウドを作成する際に、オーバークラウドはアンダークラウドの **docker-distribution** レジストリーからコンテナーイメージをプルします。この方法では、内部にレジストリーを保管することが可能なので、デプロイメントを迅速化してネットワークの輻輳を軽減することができます。ただし、アンダークラウドは基本的なレジストリーとしてのみ機能し、コンテナーイメージのライフサイクル管理は限定されます。



注記

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

Satellite Server

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

上記のリストから方法を選択し、レジストリー情報の設定を続けます。

注記

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

4.2. コンテナーイメージの準備コマンドの使用方法

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

オーバークラウド用のコンテナーイメージ環境ファイルの生成

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

--output-env-file

作成される環境ファイルの名前を定義します。

以下のスニペットは、このファイルの内容の例を示しています。

```
parameter_defaults:
  DockerAodhApiImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
  DockerAodhConfigImage: registry.redhat.io/rhosp13/openstack-aodh-api:13.0-34
...
```

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

インポート方法に対応したコンテナーイメージ一覧の生成

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

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

--output-images-file

作成されるインポート一覧のファイル名を定義します。

このファイルの内容の例を以下に示します。

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

コンテナーイメージの名前空間の設定

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

--namespace

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

--prefix

イメージ名の前に追加する接頭辞を定義します。

その結果、director は以下のような形式のイメージ名を生成します。

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

コンテナーイメージタグの設定

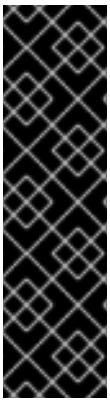
--tag および **--tag-from-label** オプションを併用して、各コンテナーイメージのタグを設定します。

--tag

ソースからの全イメージに特定のタグを設定します。このオプションだけを使用した場合、director はこのタグを使用してすべてのコンテナ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。**--tag** オプションは、デフォルトで **latest** に設定されます。

--tag-from-label

指定したコンテナ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}** を設定すると、director は **version** および **release** ラベルを使用して新しいタグを作成します。あるコンテナーについて、**version** を **13.0** に設定し、**release** を **34** に設定した場合、タグは **13.0-34** となります。



重要

Red Hat コンテナ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、**openstack overcloud container image prepare** コマンドを実行する際には、必ず **--tag-from-label {version}-{release}** を使用する必要があります。コンテナ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 latest** 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。

4.3. 追加のサービス用のコンテナーイメージ

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

-e

追加のコンテナーイメージを有効化するための環境ファイルを指定します。

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

サービス	環境ファイル
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
Collectd	**environments/services-docker/collectd.yaml**

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

以下の項には、追加するサービスの例を記載します。

Ceph Storage

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

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

--set ceph_namespace

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

--set ceph_image

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

--set ceph_tag

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

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

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

OpenStack Bare Metal (ironic)

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

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

OpenStack Data Processing (sahara)

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

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

OpenStack Neutron SR-IOV

オーバークラウドで OpenStack Neutron SR-IOV をデプロイする場合には、director がイメージを準備できるように **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml** 環境ファイルを追加します。デフォルトの Controller ロールおよび Compute ロールは SR-IOV サービスをサポートしないため、**-r** オプションを使用して SR-IOV サービスが含まれるカスタムロールファイルも追加する必要があります。以下のスニペットは、この環境ファイルの追加方法の例を示しています。

```
$ openstack overcloud container image prepare \
  ...
  -r ~/custom_roles_data.yaml
  -e /usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml \
  ...
```

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

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

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

OpenStack Shared File System (manila)

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

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

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

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

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

Red Hat では、オーバークラウドのコンテナーイメージを **registry.redhat.io** でホストしています。リモートレジストリーからイメージをプルするのが最も簡単な方法です。レジストリーはすでに設定済みで、プルするイメージの URL と名前空間を指定するだけで良いからです。ただし、オーバークラウドの作成中には、オーバークラウドノードがリモートリポジトリーからすべてのイメージをプルするので、外部接続で輻輳が生じる場合があります。したがって、実稼働環境ではこの方法は推奨されません。実稼働環境用には、この方法ではなく以下のいずれかの方法を使用してください。

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

手順

1. イメージを直接 **registry.redhat.io** からオーバークラウドデプロイメントにプルするには、イメージパラメーターを指定するための環境ファイルが必要となります。以下のコマンドを実行してコンテナーイメージの環境ファイルを生成します。

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

   - 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
   - カスタムロールファイルを指定するには、**-r** オプションを使用します。
   - Ceph Storage を使用している場合には、Ceph Storage 用のコンテナーイメージの場所を定義する追加のパラメーターを指定します: **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**

2. **overcloud_images.yaml** ファイルを変更し、デプロイメント時に **registry.redhat.io** との間で認証を行うために以下のパラメーターを追加します。

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

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

注記

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

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

レジストリーの設定が完了しました。

4.5. ローカルレジストリーとしてアンダークラウドを使用する方法

アンダークラウド上でローカルレジストリーを設定して、オーバークラウドのコンテナーイメージを保管することができます。

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

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

手順

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

```
<REGISTRY_IP_ADDRESS>:8787
```

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

2. **registry.redhat.io** にログインします。

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

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

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

- 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
- カスタムロールファイルを指定するには、**-r** オプションを使用します。
- Ceph Storage を使用している場合には、Ceph Storage 用のコンテナーイメージの場所を定義する追加のパラメーターを指定します: **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**

4. 次の 2 つのファイルが作成されていることを確認します。

- リモートソースからのコンテナーイメージの情報が含まれている **local_registry_images.yaml**。このファイルを使用して、Red Hat Container Registry (**registry.redhat.io**) からイメージをアンダークラウドにプルします。
- アンダークラウド上の最終的なイメージの場所が記載されている **overcloud_images.yaml**。このファイルをデプロイメントで指定します。

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

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

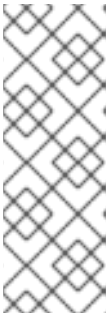
ネットワークおよびアンダークラウドディスクの速度によっては、必要なイメージをプルするのに時間がかかる場合があります。

注記

コンテナーイメージは、およそ 10 GB のディスク領域を使用します。

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

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



注記

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

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

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

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

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

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

レジストリーの設定が完了しました。

4.6. SATELLITE サーバーをレジストリーとして使用する手順

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

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

手順

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

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

- 任意のサービス用の環境ファイルを指定するには、**-e** オプションを使用します。
- カスタムロールファイルを指定するには、**-r** オプションを使用します。

- Ceph Storage を使用している場合には、Ceph Storage 用のコンテナーイメージの場所を定義する追加のパラメーターを指定します: **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**

注記

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

2. これで、コンテナーイメージの情報が含まれた **satellite_images** という名前のファイルが作成されます。このファイルを使用して、コンテナーイメージを Satellite 6 サーバーに同期します。

3. **satellite_images** ファイルから YAML 固有の情報を削除して、イメージ一覧のみが記載されたフラットファイルに変換します。この操作は、以下の **sed** コマンドで実行します。

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

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

4. Satellite 6 の **hammer** ツールがインストールされているシステムに **satellite_images_names** ファイルをコピーします。あるいは、Hammer CLI ガイド に記載の手順に従って、アンダークラウドに **hammer** ツールをインストールします。

5. 以下の **hammer** コマンドを実行して、実際の Satellite 組織に新規製品 (**OSP13 Containers**) を作成します。

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

このカスタム製品に、イメージを保管します。

6. 製品にベースコンテナーイメージを追加します。

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

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

```
$ while read IMAGE; do \
  IMAGENAME=$(echo $IMAGE | cut -d"/" -f2 | sed "s/openstack-//g" | sed "s/:.*//g") ; \
  hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
```

```
  --url https://registry.redhat.io \
  --docker-upstream-name $IMAGE \
  --name $IMAGENAME ; done < satellite_images_names
```

8. コンテナーイメージを同期します。

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

Satellite Server が同期を完了するまで待ちます。

注記

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

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

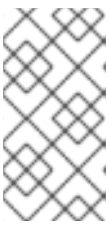
10. **base** イメージに使用可能なタグを確認します。

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

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

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

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



注記

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

このコマンドを実行する際には、以下の情報を含めてください。

- **--namespace**: Satellite サーバー上のレジストリーの URL およびポート。Red Hat Satellite のレジストリーポートは 5000 です。たとえば、**--namespace=satellite6.example.com:5000** のように設定します。

注記

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

- **--prefix=** - この接頭辞は、ラベルの Satellite 6 の命名規則に基づいており、この接頭辞は小文字を使用し、アンダースコアの代わりにスペースを使用します。この接頭辞は、コンテンツビューを使用するかどうかによって異なります。
  - コンテンツビューを使用する場合、設定は **[org]-[environment]-[content view]-[product]-** です。たとえば、**acme-production-myosp13-osp13_containers-** のようになります。
  - コンテンツビューを使用しない場合、設定は **[org]-[product]-** です。たとえば、**acme-osp13_containers-** のようになります。
- **--tag-from-label {version}-{release}**: 各イメージの最新のタグを識別します。
- **-e**: オプションのサービスの環境ファイルを指定します。
- **-r**: カスタムロールファイルを指定します。
- **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**: Ceph Storage を使用する場合には、Ceph Storage のコンテナーイメージの場所を定義する追加のパラメーターを指定します。**ceph_image** に Satellite 固有の接頭辞が追加された点に注意してください。この接頭辞は、**--prefix** オプションと同じ値です。以下に例を示します。

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

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

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

レジストリーの設定が完了しました。

4.7. 次のステップ

コンテナーイメージのソースが記載された **overcloud_images.yaml** 環境ファイルができました。今後のアップグレードとデプロイメントの操作ではすべてこのファイルを追加してください。

これで、オーバークラウドをアップグレードに向けて準備することができます。

第5章 オーバークラウドのアップグレードの準備

以下の手順では、アップグレードプロセスに向けて、オーバークラウドの準備を行います。

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

5.1. オーバークラウドの登録情報の準備

アップグレードプロセスの実行中にオーバークラウドが最新のパッケージを確実に使用するように、オーバークラウドに最新のサブスクリプション情報を提供する必要があります。

前提条件

- 最新の OpenStack Platform リポジトリーが含まれたサブスクリプション
- 登録のためのアクティベーションキーを使用する場合には、新しい OpenStack Platform リポジトリーを含む新規アクティベーションキーを作成すること

手順

1. 登録情報が記載された環境ファイルを編集します。以下に例を示します。

```
$ vi ~/templates/rhel-registration/environment-rhel-registration.yaml
```

2. 以下のパラメーター値を編集します。

rhel_reg_repos

Red Hat OpenStack Platform 13 の新しいリポジトリーを追加するように更新します。

rhel_reg_activation_key

Red Hat OpenStack Platform 13 のリポジトリーにアクセスするためのアクティベーションキーを更新します。

rhel_reg_sat_repo

Red Hat Satellite 6 のより新しいバージョンを使用する場合には、Satellite 6 の管理ツールが含まれているリポジトリーを更新します。

3. 環境ファイルを保存します。

関連情報

- 登録パラメーターの詳細は、**オーバークラウドの高度なカスタマイズ**ガイドの 環境ファイルを使用したオーバークラウドの登録 を参照してください。

5.2. 非推奨パラメーター

以下のパラメーターは非推奨となり、置き換えられた点に注意してください。

旧パラメーター	新規パラメーター
KeystoneNotificationDriver	**NotificationDriver**
controllerExtraConfig	**ControllerExtraConfig**
OvercloudControlFlavor	**OvercloudControllerFlavor**
controllerImage	**ControllerImage**
NovaImage	**ComputeImage**
NovaComputeExtraConfig	**ComputeExtraConfig**
NovaComputeServerMetadata	**ComputeServerMetadata**

NovaComputeSchedulerHints	**ComputeSchedulerHints**
	 
	注記
	カスタムの Compute ロールを使用している場合に、ロール固有の **ComputeSchedulerHints** を使用するには、以下の設定を環境に追加して、非推奨の **NovaComputeSchedulerHints** パラメーターが設定されていても定義されていないことを確認する必要があります。
	`parameter_defaults:`
	`  NovaComputeSchedulerHints: {}`
	カスタムロールを使用する場合は、ロール固有の **_ROLE_SchedulerHints** パラメーターを使用するようにこの設定を追加する必要があります。

NovaComputeIPs	**ComputeIPs**
SwiftStorageServerMetadata	**ObjectStorageServerMetadata**
SwiftStorageIPs	**ObjectStorageIPs**
SwiftStorageImage	**ObjectStorageImage**
OvercloudSwiftStorageFlavor	**OvercloudObjectStorageFlavor**
NeutronDpdkCoreList	**OvsPmdCoreList**
NeutronDpdkMemoryChannels	**OvsDpdkMemoryChannels**

旧パラメーター	新規パラメーター
NeutronDpdkSocketMemory	**OvsDpdkSocketMemory**
NeutronDpdkDriverType	**OvsDpdkDriverType**
HostCpusList	**OvsDpdkCoreList**

新規パラメーターの値には、入れ子状の一重引用符を省き二重引用符を使用します。以下に例を示します。

旧パラメーターおよび値	新規パラメーターおよび値
NeutronDpdkCoreList: "'2,3'"	**OvsPmdCoreList: "2,3"**
HostCpusList: "'0,1'"	**OvsDpdkCoreList: "0,1"**

お使いのカスタム環境ファイルのこれらのパラメーターを更新してください。以下のパラメーターは非推奨となりましたが、現在それと等価なパラメーターはありません。

NeutronL3HA
: L3 高可用性は、分散仮想ルーター (**NeutronEnableDVR**) を使用した設定を除き、すべてのケースで有効です。

CeilometerWorkers
: より新しいコンポーネント (Gnocchi、Aodh、Panko) が優先され、Ceilometer は非推奨となりました。

CinderNetappEseriesHostType
: E-series のサポートは、すべて非推奨となりました。

ControllerEnableSwiftStorage
: 代わりに、**ControllerServices** パラメーターの操作を使用する必要があります。

OpenDaylightPort
: OpenDaylight のデフォルトポートを定義するには、EndpointMap を使用します。

OpenDaylightConnectionProtocol
: このパラメーターの値は、TLS を使用してオーバークラウドをデプロイするかどうかに基づいて決定されるようになりました。

/home/stack ディレクトリーで以下の **egrep** コマンドを実行して、非推奨のパラメーターが含まれる環境ファイルを特定します。

```
$ egrep -r -w
'KeystoneNotificationDriver|controllerExtraConfig|OvercloudControlFlavor|controllerImage|NovaImage|NovaComputeExtraConfig|NovaComputeServerMetadata|NovaComputeSchedulerHints|NovaComputeIPs|SwiftStorageServerMetadata|SwiftStorageIPs|SwiftStorageImage|OvercloudSwiftStorageFlavor|NeutronDpdkCoreList|NeutronDpdkMemoryChannels|NeutronDpdkSocketMemory|NeutronDpdkDriverType|HostCpusList|NeutronDpdkCoreList|HostCpusList|NeutronL3HA|CeilometerWorkers|CinderNetappEseriesHostType|ControllerEnableSwiftStorage|OpenDaylightPort|OpenDaylightConnectionProtocol' *
```

OpenStack Platform 環境で非推奨となったこれらのパラメーターがまだ必要な場合には、デフォルトの **roles_data** ファイルで使用することができます。ただし、カスタムの **roles_data** ファイルを使用していて、オーバークラウドにそれらの非推奨パラメーターが引き続き必要な場合には、**roles_data** ファイルを編集して各ロールに以下の設定を追加することによって、パラメーターへのアクセスを可能にすることができます。

Controller ロール

```
- name: Controller
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  ...
```

Compute ロール

```
- name: Compute
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  disable_upgrade_deployment: True
  ...
```

Object Storage ロール

```
- name: ObjectStorage
  uses_deprecated_params: True
  deprecated_param_metadata: 'SwiftStorageServerMetadata'
  deprecated_param_ips: 'SwiftStorageIPs'
  deprecated_param_image: 'SwiftStorageImage'
  deprecated_param_flavor: 'OvercloudSwiftStorageFlavor'
  disable_upgrade_deployment: True
  ...
```

5.3. 非推奨の CLI オプション

環境ファイルの **parameter_defaults** セクションに追加する Heat テンプレートのパラメーターの使用が優先されるため、一部のコマンドラインオプションは古いか非推奨となっています。以下の表では、非推奨となったオプションと、それに相当する Heat テンプレートのオプションをマッピングしています。

表5.1 非推奨の CLI オプションと Heat テンプレートのパラメーターの対照表

オプション	説明	Heat テンプレートのパラメーター
--control-scale	スケールアウトするコントローラーノード数	**ControllerCount**

オプション	説明	Heat テンプレートのパラメーター
--compute-scale	スケールアウトするコンピュートノード数	**ComputeCount**
--ceph-storage-scale	スケールアウトする Ceph Storage ノードの数	**CephStorageCount**
--block-storage-scale	スケールアウトする Cinder ノード数	**BlockStorageCount**
--swift-storage-scale	スケールアウトする Swift ノード数	**ObjectStorageCount**
--control-flavor	コントローラーノードに使用するフレーバー	**OvercloudControllerFlavor**
--compute-flavor	コンピュートノードに使用するフレーバー	**OvercloudComputeFlavor**
--ceph-storage-flavor	Ceph Storage ノードに使用するフレーバー	**OvercloudCephStorageFlavor**
--block-storage-flavor	Cinder ノードに使用するフレーバー	**OvercloudBlockStorageFlavor**
--swift-storage-flavor	Swift Storage ノードに使用するフレーバー	**OvercloudSwiftStorageFlavor**
--neutron-flat-networks	フラットなネットワークが neutron プラグインで設定されるように定義します。外部ネットワークを作成ができるようにデフォルトは datacentre に設定されています。	**NeutronFlatNetworks**
--neutron-physical-bridge	各ハイパーバイザーで作成する Open vSwitch ブリッジ。デフォルトは br-ex です。通常、このパラメーターを変更する必要はありません。	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	使用する論理ブリッジから物理ブリッジへのマッピング。ホストの外部ブリッジ (br-ex) を物理名 (datacentre) にマッピングするようにデフォルト設定されています。これは、デフォルトの Floating ネットワークに使用されます。	**NeutronBridgeMappings**

オプション	説明	Heat テンプレートのパラメーター
--neutron-public-interface	ネットワークノード向けに br-ex にブリッジするインターフェイスを定義します。	**NeutronPublicInterface**
--neutron-network-type	Neutron のテナントネットワーク種別	**NeutronNetworkType**
--neutron-tunnel-types	neutron テナントネットワークのトンネリング種別。複数の値を指定するには、コンマ区切りの文字列を使用します。	**NeutronTunnelTypes**
--neutron-tunnel-id-ranges	テナントネットワークを割り当てに使用できる GRE トンネリングの ID 範囲	**NeutronTunnelIdRanges**
--neutron-vni-ranges	テナントネットワークを割り当てに使用できる VXLAN VNI の ID 範囲	**NeutronVniRanges**
--neutron-network-vlan-ranges	サポートされる Neutron ML2 および Open vSwitch VLAN マッピングの範囲。デフォルトでは、物理ネットワーク datacentre 上の VLAN を許可するように設定されています。	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	neutron テナントネットワークのメカニズムドライバー。デフォルトは openvswitch です。複数の値を指定するには、コンマ区切りの文字列を使用します。	**NeutronMechanismDrivers**
--neutron-disable-tunneling	VLAN で区切られたネットワークまたは neutron でのフラットネットワークを使用するためにトンネリングを無効化します。	パラメーターのマッピングなし
--validation-errors-fatal	オーバークラウドの作成プロセスでは、デプロイメントの前に一連のチェックが行われます。このオプションは、デプロイメント前のチェックで何らかの致命的なエラーが発生した場合に終了します。どのようなエラーが発生してもデプロイメントが失敗するので、このオプションを使用することを推奨します。	パラメーターのマッピングなし

オプション	説明	Heat テンプレートのパラメーター
--ntp-server	時刻の同期に使用する NTP サーバーを設定します。	**NtpServer**

これらのパラメーターは Red Hat OpenStack Platform から削除されました。CLI オプションは Heat パラメーターに変換して、環境ファイルに追加することを推奨します。

5.4. コンポーザブルネットワーク

Red Hat OpenStack Platform の今回のバージョンでは、コンポーザブルネットワーク向けの新機能が導入されています。カスタムの **roles_data** ファイルを使用する場合には、ファイルを編集して、コンポーザブルネットワークを各ロールに追加します。コントローラーノードの場合の例を以下に示します。

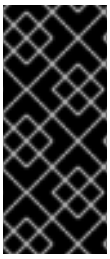
```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

その他の構文例については、デフォルトの **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** ファイルを確認してください。また、ロールの例のスニペットについては、**/usr/share/openstack-tripleo-heat-templates/roles** を確認してください。

カスタムのスタンドアロンロールとコンポーザブルネットワークの対応表を以下に示します。

ロール	必要なネットワーク
Ceph Storage Monitor	**Storage**、**StorageMgmt**
Ceph Storage OSD	**Storage**、**StorageMgmt**
Ceph Storage RadosGW	**Storage**、**StorageMgmt**
Cinder API	**InternalApi**
Compute	**InternalApi**、**Tenant**、**Storage**
Controller	**External**、**InternalApi**、**Storage**、**StorageMgmt**、**Tenant**
Database	**InternalApi**
Glance	**InternalApi**
Heat	**InternalApi**

ロール	必要なネットワーク
Horizon	**InternalApi**
Ironic	必要なネットワークはなし。API には、プロビジョニング/コントロールプレーンネットワークを使用。
Keystone	**InternalApi**
Load Balancer	**External**、**InternalApi**、**Storage**、**StorageMgmt**、**Tenant**
Manila	**InternalApi**
Message Bus	**InternalApi**
Networker	**InternalApi**、**Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**External**、**InternalApi**、**Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**Storage**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**



重要

以前のバージョンでは、***NetName** パラメーター (例: **InternalApiNetName**) によってデフォルトのネットワークの名前が変更されていました。このパラメーターはサポートされなくなりました。カスタムのコンポーザブルネットワークファイルを使用してください。詳細は、**オーバークラウドの高度なカスタマイズ**ガイドの 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。

5.5. 大規模 CEPH クラスターでの再開待機時間の延長

アップグレード中、それぞれの Ceph モニターおよび OSD は順に停止します。停止したものと同じサービスが正常に再開されるまで、移行は続行されません。Ansible は 15 秒間待って (待機) サービスの開始を確認する行為を 5 回繰り返します (リトライ)。サービスが再開されない場合には、移行は停止しオペレーターは手動操作を行う必要があります。

Ceph クラスターのサイズによっては、リトライまたは待機の値を増加しなければならない場合があります。これらのパラメーターの正確な名前およびそのデフォルト値は以下のとおりです。

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

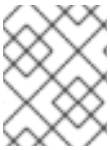
これらのパラメーターのデフォルト値を更新できます。たとえば、40 秒間待って確認する行為を 30 回 (Ceph OSD の場合)、10 秒間待って確認する行為を 20 回 (Ceph MON の場合) 繰り返すようにクラスターを設定するには、**openstack overcloud deploy** コマンドの実行時に **-e** を使用して、**yaml** ファイルの以下のパラメーターを渡します。

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
    health_osd_check_retries: 30
    health_mon_check_retries: 10
    health_mon_check_delay: 20
```

5.6. CEPH のアップグレードの準備

OpenStack Platform 13 で導入された Red Hat Ceph Storage 3 には、CephMgr サービスが必要です。OpenStack Platform 13 のデフォルトテンプレートにより提供されるロールには、CephMgr サービスが含まれています。ただし、コンポーザブルロール機能を使用してロールをカスタマイズし、オーバークラウドで Ceph をデプロイしていた場合には、CephMon サービスが含まれるロールを更新して CephMgr サービスも含める必要があります。

テンプレートを比較する方法の例は、以前のテンプレートバージョンの比較 を参照してください。



注記

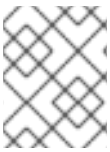
ロールをカスタマイズしてハイパーコンバージドオーバークラウドをデプロイしていた場合には、以下の手順を実施する必要があります。

手順

openstack overcloud deploy コマンドで、オーバークラウドのデプロイメントに **roles_file.yaml** 等のロールファイルを追加することができます。

いずれかの **ロール** ファイルに **OS::TripleO::Services::CephMon** が含まれている場合には、そのロールファイルに CephMgr サービスを含めます。

```
OS::TripleO::Services::CephMon
OS::TripleO::Services::CephMgr
```



注記

全 Ceph Storage ノードのアップグレード で説明されている Ceph のアップグレードを実施する前に、**OS::TripleO::Services::CephMgr** を追加する必要があります。

5.7. ノード固有の CEPH レイアウトの作成

ceph-ansible を使用してノード固有の Ceph レイアウトを作成するには、以下の操作を行います。

```
parameter_defaults:
  NodeDataLookup: |
    {"6E310C04-6186-45DB-B643-54332E76FAD1": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "1E222ACE-56B9-4F14-9DB8-929734C7CAAF": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "C6841D59-9E3E-4875-BC43-FB5F49227CE7": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]}
    }
```

5.8. カスタムの PUPPET パラメーターの確認

Puppet パラメーターのカスタマイズに **ExtraConfig** インターフェイスを使用する場合には、アップグレード中に、Puppet が重複した宣言のエラーを報告する可能性があります。これは、Puppet モジュール自体によって提供されるインターフェイスの変更が原因です。

この手順では、環境ファイル内のカスタムの **ExtraConfig** hieradata パラメーターを確認する方法を説明します。

手順

1. 環境ファイルを選択して、**ExtraConfig** パラメーターが設定されているかどうかを確認します。

   ```
   $ grep ExtraConfig ~/templates/custom-config.yaml
   ```

2. このコマンドの結果に、選択したファイル内のいずれかのロールの **ExtraConfig** パラメーター (例: **ControllerExtraConfig**) が表示される場合には、そのファイルの完全なパラメーター構造を確認してください。

3. **SECTION/parameter** 構文で **value** が続くいずれかの Puppet hieradata がパラメーターに含まれている場合には、実際の Puppet クラスのパラメーターに置き換えられている可能性があります。以下に例を示します。

   ```
   parameter_defaults:
     ExtraConfig:
       neutron::config::dhcp_agent_config:
         'DEFAULT/dnsmasq_local_resolv':
           value: 'true'
   ```

4. director の Puppet モジュールを確認して、パラメーターが Puppet クラス内に存在しているかどうかを確認します。以下に例を示します。

   ```
   $ grep dnsmasq_local_resolv
   ```

   その場合には、新規インターフェイスに変更します。

5. 構文の変更の実例を以下に示します。

   - 例 1:

     ```
     parameter_defaults:
     ```

```
    ExtraConfig:
      neutron::config::dhcp_agent_config:
        'DEFAULT/dnsmasq_local_resolv':
          value: 'true'
```

変更後

```
parameter_defaults:
  ExtraConfig:
    neutron::agents::dhcp::dnsmasq_local_resolv: true
```

- 例 2:

```
parameter_defaults:
  ExtraConfig:
    ceilometer::config::ceilometer_config:
      'oslo_messaging_rabbit/rabbit_qos_prefetch_count':
        value: '32'
```

変更後

```
parameter_defaults:
  ExtraConfig:
    oslo::messaging::rabbit::rabbit_qos_prefetch_count: '32'
```

5.9. ネットワークインターフェイスのテンプレートを新しい構造に変換する方法

以前は、ネットワークインターフェイスの構造は **OS::Heat::StructuredConfig** リソースを使用してインターフェイスを設定していました。

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            [NETWORK INTERFACE CONFIGURATION HERE]
```

テンプレートは現在、**OS::Heat::SoftwareConfig** リソースを設定に使用しています。

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
          params:
```

```
$network_config:
  network_config:
    [NETWORK INTERFACE CONFIGURATION HERE]
```

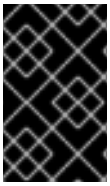
この設定では、**$network_config** 変数に保管されているインターフェイスの設定を取得して、それを **run-os-net-config.sh** スクリプトの一部として挿入します。

警告

ネットワークインターフェイスのテンプレートがこの新しい構造を使用するように更新して、ネットワークインターフェイスのテンプレートが引き続き構文を順守していることを必ず確認する必要があります。この操作を実行しないと、Fast Forward Upgrade のプロセスでエラーが発生する可能性があります。

director の Heat テンプレートコレクションには、お使いのテンプレートをこの新しい形式に変換するためのスクリプトが含まれています。このスクリプトは、**/usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py** にあります。使用方法の例を以下に示します。

```
$ /usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py \
    --script-dir /usr/share/openstack-tripleo-heat-templates/network/scripts \
    [NIC TEMPLATE] [NIC TEMPLATE] ...
```



重要

このスクリプトを使用する場合には、テンプレートにコメント化された行が含まれていないことを確認します。コメント化された行があると、古いテンプレート構造の解析時にエラーが発生する可能性があります。

詳しくは、ネットワーク分離 を参照してください。

注記

コンピュートインスタンスの高可用性 (インスタンス HA) を有効にしている Red Hat OpenStack Platform 12 またはそれ以前で、バージョン 13 またはそれ以降へのアップグレードを実施する場合には、まず手動でインスタンス HA を無効にする必要があります。手順については、以前のバージョンで設定したインスタンス HA の無効化 を参照してください。

5.10. カスタム設定ファイルを使用する BLOCK STORAGE サービスの準備

コンテナー化された環境にアップグレードする際には、**CinderVolumeOptVolumes** パラメーターを使用して docker ボリュームのマウントを追加します。これにより、cinder-volume サービスがコンテナー内で動作している場合には、ホストのカスタム設定ファイルを使用することができます。

以下に例を示します。

```
parameter_defaults:
  CinderVolumeOptVolumes:
```

```
/etc/cinder/nfs_shares1:/etc/cinder/nfs_shares1
/etc/cinder/nfs_shares2:/etc/cinder/nfs_shares2
```

5.11. 事前にプロビジョニングされたノードのアップグレードの準備

事前にプロビジョニングされたノードは、director の管理外で作成されたノードです。事前にプロビジョニングされたノードを使用するオーバークラウドでは、アップグレードの前にいくつかの追加手順が必要です。

前提条件

- オーバークラウドは、事前にプロビジョニングされたノードを使用します。

手順

1. 次のコマンドを実行して、ノード IP アドレスのリストを **OVERCLOUD_HOSTS** 環境変数に保存します。

```
$ source ~/stackrc
$ export OVERCLOUD_HOSTS=$(openstack server list -f value -c Networks | cut -d "=" -f 2 | tr '\n' ' ')
```

2. 次のスクリプトを実行します。

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

3. アップグレードを続行します。

   - 事前にプロビジョニングされたノードで **openstack overcloud upgrade run** コマンドを使用する場合は、**--ssh-user tripleo-admin** パラメーターを含めます。
   - Compute ノードまたは Object Storage ノードをアップグレードする場合は、以下を使用します。
     a. **upgrade-non-controller.sh** スクリプトで **-U** オプションを使用して、**stack** ユーザーを指定します。これは、事前にプロビジョニングされたノードのデフォルトユーザーが **heat-admin** ではなく **stack** であるためです。
     b. **--upgrade** オプションでノードの IP アドレスを使用します。これは、ノードが director の Compute (nova) サービスおよび Bare Metal (ironic) サービスで管理されておらず、ノード名がないためです。
     以下に例を示します。

```
$ upgrade-non-controller.sh -U stack --upgrade 192.168.24.100
```

関連情報

- 事前にプロビジョニングされたノードの詳細については、**director のインストールおよび使用** ガイドの 事前にプロビジョニングされたノードを使用した基本的なオーバークラウドの設定 を参照してください。

5.12. 次のステップ

オーバークラウドの準備段階が完了しました。続いて6章 *オーバークラウドのアップグレード*に記載の手順でオーバークラウドを 13 にアップグレードします。

第6章 オーバークラウドのアップグレード

以下の手順では、オーバークラウドをアップグレードします。

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
- アップグレードでの変更に対応するためのカスタム環境ファイルの準備が完了していること

6.1. OVERCLOUD UPGRADE PREPARE の実行

アップグレードには、**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 13 に更新する
- アップグレードに向けてノードを準備する

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. upgrade prepare コマンドを実行します。

```
$ openstack overcloud upgrade prepare \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e <ENVIRONMENT FILE>
```

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。

- カスタム設定環境ファイル (**-e**)
- 新しいコンテナーイメージの場所が記載された環境ファイル (**-e**)。アップグレードのコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるようになっているので、警告は無視して問題ありません。
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) の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**) のファイルを指定します。

3. アップグレードの準備が完了するまで待ちます。

6.2. コントローラーノードおよびカスタムロールノードのアップグレード

すべてのコントローラーノード、分割されたコントローラーサービス、およびその他のカスタムノードを OpenStack Platform 13 にアップグレードするには、以下のプロセスを使用します。このプロセスでは、**--nodes** オプションを指定して **openstack overcloud upgrade run** コマンドを実行し、操作を選択したノードだけに制限します。

```
$ openstack overcloud upgrade run --nodes [ROLE]
```

[ROLE] をロール名またはロール名のコンマ区切りリストに置き換えます。

オーバークラウドでモノリシックなコントローラーノードが使用されている場合は、**Controller** ロールに対してこのコマンドを実行します。

オーバークラウドで分割されたコントローラーサービスが使用されている場合は、以下のガイドに従ってノードのロールを次の順序でアップグレードします。

- Pacemaker を使用するすべてのロール。たとえば、**ControllerOpenStack**、**Database**、**Messaging**、**Telemetry** 等。
- **Networker** ノード
- その他すべてのカスタムロール

以下のノードはまだアップグレード **しないでください**。

- DPDK ベースまたはハイパーコンバージドインフラストラクチャー (HCI) コンピュートノードなど、あらゆる種別のコンピュートノード
- **CephStorage** ノード

これらのノードは後でアップグレードします。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. モノリシックなコントローラーノードを使用している場合は、**Controller** ロールに対してアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes Controller
```

   - 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. 複数のロールにわたって分割されたコントローラーサービスを使用している場合の操作は以下のとおりです。

   a. Pacemaker サービスを使用するロールのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes ControllerOpenStack
$ openstack overcloud upgrade run --nodes Database
$ openstack overcloud upgrade run --nodes Messaging
$ openstack overcloud upgrade run --nodes Telemetry
```

      - 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。

   b. **Networker** ロールのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes Networker
```

- 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。

c. **Compute** ロールまたは **CephStorage** ロールを除く、残りすべてのカスタムロールのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes ObjectStorage
```

- 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。

6.3. 全コンピュートノードのアップグレード

重要

- ハイパーコンバージドのデプロイメントを使用している場合には、「ハイパーコンバージドノードのアップグレード」でアップグレード方法を確認してください。
- 混在型ハイパーコンバージドのデプロイメントを使用している場合には、「混在型ハイパーコンバージドノードのアップグレード」でアップグレード方法を確認してください。

このプロセスでは、残りのコンピュートノードをすべて OpenStackPlatform 13 にアップグレードします。このプロセスは、**openstack overcloud upgrade run** コマンドに **--nodes Compute** オプションを指定して、操作をコンピュートノードのみに制限して実行する必要があります。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes Compute
```

- カスタムのスタック名を使用している場合には、**--stack** オプションでその名前を渡します。
- カスタムの Compute ロールを使用する場合には、**--nodes** オプションでそのロール名を含めます。

3. コンピュートノードのアップグレードが完了するまで待ちます。

6.4. 全 CEPH STORAGE ノードのアップグレード

重要

- ハイパーコンバージドのデプロイメントを使用している場合には、「ハイパーコンバージドノードのアップグレード」でアップグレード方法を確認してください。
- 混在型ハイパーコンバージドのデプロイメントを使用している場合には、「混在型ハイパーコンバージドノードのアップグレード」でアップグレード方法を確認してください。

このプロセスでは、Ceph Storage ノードをアップグレードします。このプロセスには、以下の操作が必要です。

- **openstack overcloud upgrade run** コマンドに **--nodes CephStorage** オプションを指定して、操作を Ceph Storage ノードのみに制限して実行する
- **openstack overcloud ceph-upgrade run** コマンドを実行し、コンテナー化された Red Hat Ceph Storage 3 クラスターへのアップグレードを実施する

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

```
$ openstack overcloud upgrade run --nodes CephStorage
```

   - カスタムのスタック名を使用する場合には、**--stack** オプションでその名前を渡します。

3. ノードのアップグレードが完了するまで待ちます。
4. Ceph Storage のアップグレードコマンドを実行します。以下に例を示します。

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

   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。

   - カスタム設定環境ファイル (**-e**)。以下に例を示します。
     - コンテナーイメージの場所が記載された環境ファイル (**overcloud_images.yaml**)。アップグレードのコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるようになっているので、警告は無視して問題ありません。
     - Ceph Storage ノード用の関連する環境ファイル
     - お使いの環境に関連する追加の環境ファイル
   - カスタムのスタック名を使用する場合には、**--stack** オプションでその名前を渡します。
   - 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) の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**) のファイルを指定します。

5. Ceph Storage ノードのアップグレードが完了するまで待ちます。

6.5. ハイパーコンバージドノードのアップグレード

ComputeHCI ロールからのハイパーコンバージドノードしか使用しておらず、専用のコンピュートノードまたは Ceph ノードを使用していない場合には、以下の手順を実施してノードをアップグレードします。

手順

1. source コマンドで stackrc ファイルを読み込みます。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

```
$ openstack overcloud upgrade run --roles ComputeHCI
```

カスタムのスタック名を使用している場合には、**--stack** オプションでアップグレードコマンドにその名前を渡します。

3. Ceph Storage のアップグレードコマンドを実行します。以下に例を示します。

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

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。

- カスタム設定環境ファイル (**-e**)。以下に例を示します。
  - コンテナーイメージの場所が記載された環境ファイル (**overcloud_images.yaml**)。アップグレードのコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるようになっているので、警告は無視して問題ありません。
  - Ceph Storage ノード用の関連する環境ファイル
- カスタムのスタック名を使用する場合には、**--stack** オプションでその名前を渡します。
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) の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**) のファイルを指定します。

4. Ceph Storage ノードのアップグレードが完了するまで待ちます。

6.6. 混在型ハイパーコンバージドノードのアップグレード

ComputeHCI ロール等のハイパーコンバージドノードに加えて専用のコンピュートノードまたは Ceph ノードを使用している場合には、以下の手順を実施してノードをアップグレードします。

手順

1. source コマンドで stackrc ファイルを読み込みます。

```
$ source ~/stackrc
```

2. コンピュートノードのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --roles Compute
If using a custom stack name, pass the name with the --stack option.
```

3. ノードのアップグレードが完了するまで待ちます。

4. ComputeHCI ノードのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --roles ComputeHCI
If using a custom stack name, pass the name with the --stack option.
```

5. ノードのアップグレードが完了するまで待ちます。

6. Ceph Storage ノードのアップグレードコマンドを実行します。

```
$ openstack overcloud upgrade run --roles CephStorage
```

7. Ceph Storage ノードのアップグレードが完了するまで待ちます。

8. Ceph Storage のアップグレードコマンドを実行します。以下に例を示します。

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

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。

- カスタム設定環境ファイル (**-e**)。以下に例を示します。
  - コンテナーイメージの場所が記載された環境ファイル (**overcloud_images.yaml**)。アップグレードのコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるようになっているので、警告は無視して問題ありません。
  - Ceph Storage ノード用の関連する環境ファイル
  - お使いの環境に関連する追加の環境ファイル
- カスタムのスタック名を使用する場合には、**--stack** オプションでその名前を渡します。
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) の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**) のファイルを指定します。

9. Ceph Storage ノードのアップグレードが完了するまで待ちます。

6.7. アップグレードの最終処理

アップグレードには、オーバークラウドスタックを更新する最終ステップが必要です。これにより、スタックのリソース構造が OpenStackPlatform 13 の標準のデプロイメントと一致し、今後、通常の **openstack overcloud deploy** の機能を実行できるようになります。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. アップグレードの最終処理コマンドを実行します。

```
$ openstack overcloud upgrade converge \
  --templates \
  -e <ENVIRONMENT FILE>
```

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。

- カスタム設定環境ファイル (**-e**)。
- カスタムのスタック名を使用する場合には、**--stack** オプションでその名前を渡します。
- 該当する場合は、**--roles-file** でカスタムロール (**roles_data**) の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**) のファイルを指定します。

3. アップグレードの最終処理が完了するまで待ちます。

第7章 アップグレード後のステップの実行

このプロセスでは、主要なアップグレードプロセスが完了した後の最終ステップを実行します。

前提条件

- オーバークラウドを最新のメジャーリリースにアップグレードする作業が完了していること

7.1. オーバークラウドのアップグレード後の一般的な考慮事項

以下の項目は、オーバークラウドのアップグレード後の一般的な考慮事項です。

- 必要な場合にはオーバークラウドノードで作成された設定ファイルを確認します。パッケージをアップグレードすると、各サービスのアップレードされたバージョンに適した **.rpmnew** ファイルがインストールされている可能性があります。
- コンピュートノードが **neutron-openvswitch-agent** の問題をレポートする可能性があります。これが発生した場合には、各コンピュートノードにログインして、このサービスを再起動します。以下に例を示します。

```
$ sudo docker restart neutron_ovs_agent
```

- 状況によっては、コントローラーノードのリブート後に IPv6 環境で **corosync** サービスの起動に失敗する可能性があります。これは、コントローラーノードが静的な IPv6 アドレスを設定する前に Corosync が起動してしまうことが原因です。このような場合は、コントローラーノードで Corosync を手動で再起動してください。

```
$ sudo systemctl restart corosync
```