



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

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

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本ドキュメントでは、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	3
1.1. ワークフローの概要	3
1.2. リポジトリ	3
第2章 OPENSTACK PLATFORM のアップグレードの準備	6
2.1. アンダークラウドのバックアップ	6
2.2. コンテナ化されたオーバークラウドのコントロールプレーンサービスのバックアップ	7
2.3. アンダークラウドのマイナーアップデートの実行	9
2.4. コンテナ化されたオーバークラウドのマイナー更新の実行	10
2.5. コントローラーノードおよびコンポーザブルノードの再起動	11
2.6. CEPH STORAGE (OSD) クラスターの再起動	12
2.7. コンピュートノードの再起動	13
2.8. アンダークラウドの検証	14
2.9. コンテナ化されたオーバークラウドの検証	15
第3章 アンダークラウドのアップグレード	18
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	18
3.2. オーバークラウドイメージのアップグレード	18
3.3. 以前のテンプレートバージョンの比較	19
3.4. 次のステップ	20
第4章 コンテナイメージのソースの設定	21
4.1. レジストリーメソッド	21
4.2. CONTAINER IMAGE PREPARE コマンドの使用方法	21
4.3. 追加のサービス用のコンテナイメージ	23
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	25
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	25
4.6. SATELLITE サーバーをレジストリーとして使用する手順	27
4.7. 次のステップ	29
第5章 オーバークラウドのアップグレードの準備	31
5.1. オーバークラウドの登録情報の準備	31
5.2. 非推奨パラメーター	31
5.3. 非推奨の CLI オプション	33
5.4. コンポーザブルネットワーク	35
5.5. カスタムの PUPPET パラメーターの確認	37
5.6. ネットワークインターフェースのテンプレートを新しい構造に変換する方法	38
5.7. 次のステップ	40
第6章 オーバークラウドのアップグレード	41
6.1. OVERCLOUD UPGRADE PREPARE の実行	41
6.2. 全コントローラーノードのアップグレード	41
6.3. 全コンピュートノードのアップグレード	42
6.4. 全 CEPH STORAGE ノードのアップグレード	42
6.5. アップグレードの最終処理	43
第7章 アップグレード後のステップの実行	45
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	45

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の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	アンダークラウドと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てから、最新のマイナーリリースに更新し、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。
コンテナイメージの取得	OpenStack Platform 13 のサービス用のコンテナ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。
コンピューターノードのアップグレード	選択したコンピューター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、全コンピューター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

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

表1.1 OpenStack Platform リポジトリ

名前	リポジトリ	説明
Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rpms	x86_64 システム用ベースオペレーティングシステムのリポジトリ
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	rhel-7-server-extras-rpms	Red Hat OpenStack Platform の依存関係が含まれます。
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	rhel-7-server-rh-common-rpms	Red Hat OpenStack Platform のデプロイと設定ツールが含まれます。
Red Hat Satellite Tools for RHEL 7 Server RPMs x86_64	rhel-7-server-satellite-tools-6.3-rpms	Red Hat Satellite 6 でのホスト管理ツール
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	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。Red Hat OpenStack Platform director のパッケージも含まれます。
Red Hat Ceph Storage OSD 3 for Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rhceph-3-osd-rpms	(Ceph Storage ノード向け) Ceph Storage Object Storage デーモンのリポジトリ。Ceph Storage ノードにインストールします。
Red Hat Ceph Storage MON 3 for Red Hat Enterprise Linux 7 Server (RPMs)	rhel-7-server-rhceph-3-mon-rpms	(Ceph Storage ノード向け) Ceph Storage Monitor デーモンのリポジトリ。Ceph Storage ノードを使用して OpenStack 環境にあるコントローラーノードにインストールします
Red Hat Ceph Storage Tools 2 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。このリポジトリは、Ceph Storage クラスターを使用するオーバークラウドをデプロイする際に、全ノードに有効化する必要があります。

名前	リポジトリ	説明
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	rhel-7-server-nfv-rpms	NFV 向けのリアルタイム KVM (RT-KVM) のリポジトリ。リアルタイムカーネルを有効化するためのパッケージが含まれています。このリポジトリは、RT-KVM 対象の全コンピュータノードで有効化する必要があります。



注記

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

第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 を準備します。これには、以下のステップを伴います。

- アンダークラウドとオーバークラウドの両方をバックアップします。
- 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再起動します。
- `overcloud upgrade` コマンドでオーバークラウドを更新します。
- 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、オーバークラウドノードを再起動します。
- 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します。

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. アンダークラウドのバックアップ

完全なアンダークラウドのバックアップには、以下のデータベースおよびファイルが含まれます。

- アンダークラウドノード上の MariaDB データベース
- (データベースを正確に復元できるように) アンダークラウド上の MariaDB 設定ファイル
- 設定データ: `/etc`
- ログデータ: `/var/log`
- イメージデータ: `/var/lib/glance`
- 証明書生成データ: `/var/lib/certmonger`
- コンテナイメージデータ: `/var/lib/docker`、`/var/lib/registry`
- Swift の全データ: `/srv/node`
- `stack` ユーザーのホームディレクトリー (`/home/stack`) 内の全データ: `/home/stack`



注記

バックアッププロセスを実行する前に、アンダークラウドに利用可能なディスク容量が十分であることを確認します。アーカイブファイルは、少なくとも 3.5 GB となることが予想され、それ以上になる可能性があります。

手順

1. アンダークラウドに **root** ユーザーとしてログインします。
2. データベースをバックアップします。

-

```
# mysqldump --opt --all-databases > /root/undercloud-all-databases.sql
```

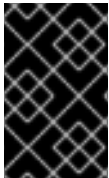
3. データベースのバックアップと設定ファイルをアーカイブします。

```
# sudo tar --xattrs --ignore-failed-read -cf \
    undercloud-backup-`date +%F`.tar \
    /root/undercloud-all-databases.sql \
    /etc \
    /var/log \
    /var/lib/glance \
    /var/lib/certmonger \
    /var/lib/docker \
    /var/lib/registry \
    /srv/node \
    /root \
    /home/stack
```

これで、**undercloud-backup-`<date>`.tar.gz** という名前のファイルが作成されます。**<date>** はシステムの日付です。この **tar** ファイルをセキュアな場所にコピーします。

2.2. コンテナ化されたオーバークラウドのコントロールプレーンサービスのバックアップ

以下の手順では、コンテナ化されたオーバークラウドのデータベースと設定のバックアップを作成します。オーバークラウドのデータベースとサービスのバックアップにより、作業環境のスナップショットが確保されます。スナップショットがあると、操作のエラーが発生してオーバークラウドを元の状態に復元する必要がある場合に役立ちます。



重要

この手順では、不可欠なコントロールプレーンサービスのみが含まれます。コンピュータノードのワークロード、Ceph Storage ノード上のデータ、追加のサービスのバックアップは対象外です。

手順

1. データベースのバックアップを実行します。
 - a. コントロールノードにログインします。オーバークラウドには、アンダークラウドからアクセスできます。

```
$ ssh heat-admin@192.0.2.100
```

- b. **root** ユーザーに変更します。

```
$ sudo -i
```

- c. バックアップを保管するための一時ディレクトリを作成します。

```
# mkdir -p /var/tmp/mysql_backup/
```

- d. データベースのパスワードを取得して、**MYSQLDBPASS** の環境変数に保存します。このパ

スワードは、`/etc/puppet/hieradata/service_configs.json` ファイルの `mysql::server::root_password` の変数に保管されています。以下のコマンドを使用してパスワードを保管します。

```
# MYSQLDBPASS=$(sudo hiera mysql::server::root_password)
```

- e. データベースをバックアップします。

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "select distinct
table_schema from information_schema.tables where engine='innodb'
and table_schema != 'mysql';" | xargs mysqldump -uroot -
p$MYSQLDBPASS --single-transaction --databases >
/var/tmp/mysql_backup/openstack_databases-`date +%F`-`date
+%T`.sql
```

このコマンドにより、`/var/tmp/mysql_backup/openstack_databases-<date>.sql` という名前のデータベースバックアップがダンプされます。**<date>** はシステムの日付と時刻です。このデータベースダンプを安全な場所にコピーします。

- f. ユーザーおよびパーミッションに関する全情報をバックアップします。

```
# mysql -uroot -p$MYSQLDBPASS -s -N -e "SELECT CONCAT('\nSHOW
GRANTS FOR ''',user, '''@''',host, ''';\n') FROM mysql.user where
(length(user) > 0 and user NOT LIKE 'root')" | xargs -n1 mysql -
uroot -p$MYSQLDBPASS -s -N -e | sed 's/$/;/' >
/var/tmp/mysql_backup/openstack_databases_grants-`date +%F`-`date
+%T`.sql
```

このコマンドにより、`/var/tmp/mysql_backup/openstack_databases_grants-<date>.sql` という名前のデータベースバックアップがダンプされます。**<date>** はシステムの日付と時刻です。このデータベースダンプを安全な場所にコピーします。

2. OpenStack Telemetry データベースをバックアップします。

- a. 任意のコントローラーに接続して、MongoDB のプライマリインスタンスの IP を取得します。

```
# MONGOIP=$(sudo hiera mongodb::server::bind_ip)
```

- b. バックアップを作成します。

```
# mkdir -p /var/tmp/mongo_backup/
# mongodump --oplog --host $MONGOIP --out /var/tmp/mongo_backup/
```

- c. `/var/tmp/mongo_backup/` 内のデータベースダンプを安全な場所にコピーします。

3. Redis クラスターをバックアップします。

- a. HAProxy から Redis のエンドポイントを取得します。

```
# REDISIP=$(sudo hiera redis_vip)
```

- b. Redis クラスターのマスターパスワードを取得します。

```
# REDISPASS=$(sudo hiera redis::masterauth)
```

- c. Redis クラスターの接続をチェックします。

```
# redis-cli -a $REDISPASS -h $REDISIP ping
```

- d. Redis データベースをダンプします。

```
# redis-cli -a $REDISPASS -h $REDISIP bgsave
```

このコマンドにより、データベースのバックアップがデフォルトの `/var/lib/redis/` ディレクトリーに保管されます。このデータベースダンプを安全な場所にコピーします。

4. 各コントローラーノードのファイルシステムをバックアップします。

- a. バックアップ用のディレクトリーを作成します。

```
# mkdir -p /var/tmp/filesystem_backup/
```

- b. 以下の **tar** コマンドを実行します。

```
# tar --ignore-failed-read --xattrs \
  -zcvf /var/tmp/filesystem_backup/fs_backup-`date +%Y-%m-%d-%H-%M-%S`.tar.gz \
  /var/lib/config-data \
  /var/log/containers \
  /etc/corosync \
  /etc/logrotate.d \
  /etc/openvswitch \
  /var/log/openvswitch \
  /srv/node \
  /home/heat-admin
```

--ignore-failed-read オプションを使用すると、見つからないディレクトリーは無視されます。これは、特定のサービスが使用されていない場合や、独自のカスタムロール上に分離されている場合に役立ちます。

5. 作成された **tar** ファイルを安全な場所にコピーします。

2.3. アンダークラウドのマイナーアップデートの実行

director では、アンダークラウドノード上のパッケージを更新するためのコマンドが提供されています。これにより、OpenStack Platform 環境の現在のバージョン内のマイナー更新を実行することができます。

手順

1. director に **stack** ユーザーとしてログインします。
2. **python-tripleoclient** パッケージと依存関係を更新し、マイナーバージョンの更新向けの最新のスクリプトを使用できるようにします。

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

3. **director** は **openstack undercloud upgrade** コマンドを使用して、アンダークラウドの環境を更新します。以下のコマンドを実行します。

```
$ openstack undercloud upgrade
```

4. 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。
5. アンダークラウドを再起動して、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
```

6. ノードが起動するまで待ちます。

2.4. コンテナ化されたオーバークラウドのマイナー更新の実行

director では、全オーバークラウドノード上のパッケージを更新するためのコマンドが提供されています。これにより、OpenStack Platform 環境の現在のバージョン内のマイナー更新を実行することができます。

手順

1. コンテナ化されたサービスのイメージの最新タグを確認します。

```
$ openstack overcloud container image tag discover \
  --image registry.access.redhat.com/rhosp12/openstack-base:latest \
  --tag-from-label version-release
```

最も新しいタグを書き留めておきます。

2. **openstack overcloud container image prepare** コマンドで、コンテナイメージのソース用の更新された環境ファイルを作成します。たとえば、**registry.access.redhat.com** からのイメージを使用する場合は、以下のコマンドを実行します。

```
$ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp12 \
  --prefix=openstack- \
  --tag [TAG] \
  --set ceph_namespace=registry.access.redhat.com/rhceph \
  --set ceph_image=rhceph-2-rhel7 \
  --set ceph_tag=latest \
  --env-file=/home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/custom_environment_file.yaml
```

1 **[TAG]** の箇所は、前のステップで取得したタグに置き換えます。

2 **-e** パラメーターで追加の環境ファイルをすべて指定します。**director** は指定されているすべての環境ファイルでカスタムリソースを確認して、コンテナ化されたサービスに必要なコンテナイメージを特定します。

この環境ファイルを異なるソース種別用に生成する方法についての詳しい情報は、『**director のインストールと使用方法**』ガイドの「[コンテナイメージのソースの設定](#)」を参照してください。

3. **openstack overcloud update stack** コマンドを実行して、オーバークラウド内のコンテナイメージの場所を更新します。

```
$ openstack overcloud update stack --init-minor-update \
  --container-registry-file
/home/stack/templates/overcloud_images.yaml
```

--init-minor-update はオーバークラウドスタック内のパラメーターの更新のみを実行します。実際のパッケージやコンテナの更新は行いません。このコマンドが完了するまで待ちます。

4. **openstack overcloud update** コマンドを使用してパッケージとコンテナの更新を実行します。**--nodes** オプションを使用して、各ロールのノードをアップグレードします。たとえば、以下のコマンドは、**Controller** ロールのノードを更新します。

```
$ openstack overcloud update stack --nodes Controller
```

以下の順序で、各ロールグループにこのコマンドを実行します。

- **Controller**
- **CephStorage**
- **Compute**
- **ObjectStorage**
- **Database、MessageBus、Networker** などのカスタムロール

5. 選択したロールの更新プロセスが開始します。director は Ansible playbook を使用して更新を実行し、各タスクの出力を表示します。
6. 次のロールグループを更新します。全ノードの更新が完了するまで作業を繰り返してください。

2.5. コントローラーノードおよびコンポーザブルノードの再起動

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

手順

1. ノードを選択してログインします。
2. ノードを再起動します。

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

3. ノードが起動するまで待ちます。

4. ノードにログインしてサービスをチェックします。以下に例を示します。

- a. ノードが Pacemaker サービスを使用している場合には、ノードがクラスターに再度参加したかどうかを確認します。

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

- b. ノードが Systemd サービスを使用している場合には、全サービスが有効化されているかどうかを確認します。

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

2.6. CEPH STORAGE (OSD) クラスターの再起動

以下の手順では、Ceph Storage (OSD) ノードのクラスターを再起動します。

手順

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

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

2. 再起動する最初の Ceph Storage ノードを選択して、ログインします。
3. ノードを再起動します。

```
$ sudo reboot
```

4. ノードが起動するまで待ちます。
5. ノードにログインして、クラスターのステータスを確認します。

```
$ sudo ceph -s
```

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

6. ノードからログアウトして、次のノードを再起動し、ステータスを確認します。全 Ceph Storage ノードが再起動されるまで、このプロセスを繰り返します。
7. 完了したら、Ceph MON 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
```


2.7. コンピュートノードの再起動

以下の手順では、コンピュートノードを再起動します。OpenStackPlatform 環境内のインスタンスのダウンタイムを最小限に抑えるために、この手順には、選択したコンピュートノードからインスタンスを移行するステップも含まれています。これは、以下のワークフローを伴います。

- 再起動するコンピュー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します。
- インスタンスを別のコンピュートノードに移行します。
- 空のコンピュートノードを再起動して有効化します。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. 全コンピュートノードとその UUID を一覧表示します。

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

再起動するコンピュートノードの UUID を特定します。

3. アンダークラウドから、コンピュートノードを選択し、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. 以下のコマンドの 1 つを使用して、インスタンスを移行します。

- a. 選択した特定のホストにインスタンスを移行します。

```
(overcloud) $ openstack server migrate [instance-id] --live
[target-host]--wait
```

- b. **nova-scheduler** により対象のホストが自動的に選択されるようにします。

```
(overcloud) $ nova live-migration [instance-id]
```

- c. 一度にすべてのインスタンスのライブマイグレーションを行います。

```
$ nova host-evacuate-live [hostname]
```



注記

nova コマンドで非推奨の警告が表示される可能性があります、安全に無視することができます。

6. 移行が完了するまで待ちます。

7. 正常に移行したことを確認します。

```
(overcloud) $ openstack server list --host [hostname] --all-projects
```

8. 選択したコンピューターノードのインスタンスがなくなるまで、移行を続けます。

9. コンピューターノードのログインしてリブートします。

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

10. ノードが起動するまで待ちます。

11. コンピューターノードを再度有効化します。

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

12. コンピューターノードが有効化されているかどうかを確認します。

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

2.8. 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手順を以下に示します。

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
```

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 のエントリを完全削除する方法は <https://access.redhat.com/solutions/2215131> のソリューションに記載されています。

2.9. コンテナ化されたオーバークラウドの検証

コンテナ化されたオーバークラウドの機能を確認する手順を以下に示します。

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)" | awk -F',' '{
print $1" "$2" "$18 }'
```

<PASSWORD> と <IP ADDRESS> は、**haproxy.stats** サービスからのそれぞれの情報に置き換えます。その結果表示される一覧には、各ノード上の OpenStackPlatform サービスとそれらの接続ステータスが表示されます。

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チェックします。

```
(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 環境をチェックして、Red Hat の推奨値に合わせて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 Heat のコアテンプレートを **OpenStack Platform 13** リリースにアップグレードします。

手順

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

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

3. 新しい OpenStack Platform リポジトリを有効にします。

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

4. **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
```

5. 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
```

6. 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。
7. アンダークラウドを再起動して、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
```

8. ノードが起動するまで待ちます。

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ますようになります。

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. **stack** ユーザーのホーム (**/home/stack/images**) にある **images** ディレクトリーから既存のイメージをすべて削除します。

```
$ rm -rf ~/images/*
```

2. 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 ~
```

3. **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/
```

4. 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)
```

5. 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。

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 (**ocata**) 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/osp12/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. 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 コンテナイメージのソースの設定

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

本ガイドでは、レジストリーを使用するためのユースケースをいくつか記載しています。これらのユースケースの1つを試す前には、イメージを準備するコマンドの使用法をよく理解しておくことを推奨します。詳しくは、「[container image prepare コマンドの使用法](#)」を参照してください。

4.1. レジストリーメソッド

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

ローカルレジストリー

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

Satellite サーバー

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

上記のリストから方法を選択し、レジストリー情報の設定を続けます。

4.2. CONTAINER IMAGE PREPARE コマンドの使用法

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

オーバークラウド用のコンテナイメージ環境ファイルの生成

openstack overcloud container image prepare コマンドの主要な用途の1つに、オーバークラウドが使用するイメージの一覧が記載された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.access.redhat.com/rhosp13/openstack-aodh-api:latest
DockerAodhConfigImage: registry.access.redhat.com/rhosp13/openstack-aodh-api:latest
...
```

インポート方法に対応したコンテナイメージ一覧の生成

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

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

--output-images-file

作成されるインポート一覧のファイル名を定義します。

このファイルの内容の例を以下に示します。

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

コンテナイメージの名前空間の設定

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

--namespace

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

--prefix

イメージ名の前に追加するプレフィックスを定義します。

その結果、director は以下のような形式のイメージ名を生成します。

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

コンテナイメージタグの設定

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

--tag-from-label

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

--tag

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

4.3. 追加のサービス用のコンテナイメージ

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

-e

追加のコンテナイメージを有効化するための環境ファイルを指定します。

以下の表は、コンテナイメージを使用する追加のサービスのサンプルー一覧とそれらの対応する環境ファイルがある /usr/share/openstack-tripleo-heat-templates ディレクトリー内の場所をまとめています。

サービス	環境ファイル
Ceph Storage	environments/ceph-ansible/ceph-ansible.yaml
Collectd	environments/services-docker/collectd.yaml
Congress	environments/services-docker/congress.yaml
Fluentd	environments/services-docker/fluentd-client.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/services-docker/manila.yaml
Sensu	environments/services-docker/sensu-client.yaml

以下の項には、追加するサービスの例を記載します。

Ceph Storage

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

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

--set ceph_namespace

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

--set ceph_image

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

--set ceph_tag

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

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

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

OpenStack Bare Metal (ironic)

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

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

OpenStack Data Processing (sahara)

オーバークラウドで OpenStack Bare Metal (ironic) 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 \
...
```

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方法

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

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

手順

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

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

- **-e**: オプションのサービスの環境ファイルを指定します。
 - Ceph Storage を使用している場合は、Ceph Storage コンテナイメージの場所を定義するための追加のパラメーターを含めます: **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**
2. これで、イメージの場所が記載された **overcloud_images.yaml** 環境ファイルがアンダークラウド上に作成されます。このファイルをデプロイメントで指定します。

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イメージを保管することができます。この方法は、以下の操作を伴います。

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

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

手順

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

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

アンダークラウドの IP アドレスを使用します。これは、以前に **undercloud.conf** ファイルの **local_ip** パラメーターで設定されています。以下のコマンドでは、このアドレスが **192.168.24.1:8787** であると仮定しています。

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

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

- **-e**: オプションのサービスの環境ファイルを指定します。
 - Ceph Storage を使用している場合は、Ceph Storage コンテナイメージの場所を定義するための追加のパラメーターを含めます: **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**
3. これで 2 つのファイルが作成されます。
 - リモートソースからのコンテナイメージの情報が含まれている **local_registry_images.yaml**。このファイルを使用して、Red Hat Container Registry (**registry.access.redhat.com**) から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**。このファイルはデプロイメントで指定します。両方のファイルが存在することを確認します。
 4. コンテナイメージを **registry.access.redhat.com** からアンダークラウドにプルします。

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

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



注記

コンテナイメージは、およそ 10 GB のディスク領域を使用します。

レジストリーの設定の準備が整いました。

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

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

手順

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

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

- **-e**: オプションのサービスの環境ファイルを指定します。
- Ceph Storage を使用している場合は、Ceph Storage コンテナイメージの場所を定義するための追加のパラメーターを含めます: **--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.access.redhat.com** のレジストリーをターゲットにしてイメージの一覧を生成します。この後のステップでは、**openstack overcloud container image prepare** コマンドで別の値を使用します。

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

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

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

4. Satellite 6 の **hammer** ツールがインストールされているシステムに **satellite_images_names** ファイルをコピーします。または、『[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、新しい製品 (**OSP13 Containers**) を Satellite の組織に作成します。

■

```
$ 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。

- 製品にベースコンテナイメージを追加します。

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

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

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

- コンテナイメージを同期します。

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

Satellite サーバーが同期を完了するまで待ちます。



注記

設定によっては、**hammer** が Satellite サーバーのユーザー名とパスワードを要求する場合があります。設定ファイルを使用して、**hammer** が自動ログインするように設定することができます。『**Hammer CLI ガイド**』の「[認証](#)」の項を参照してください。

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

- base** イメージに使用可能なタグを確認します。

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

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

11. アンダークラウドに戻り、Satellite サーバー上のイメージ用に環境ファイルを作成します。環境ファイルを作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

上記の **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。これは、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使用します。

このコマンドを実行する際には、以下の情報を含めてください。

- **--namespace**: Satellite サーバー上のレジストリーの URL およびポート。Red Hat Satellite のデフォルトのレジストリーポートは 5000 です。例: **--namespace=satellite6.example.com:5000**
- **--prefix**: プレフィックスは Satellite 6 の命名規則に基づきます。これは、コンテンツビューを使用するかどうかによって異なります。
 - コンテンツビューを使用する場合、構造は **[org]-[environment]-[content view]-[product]** となります (例: **acme-production-myosp13-osp13_containers-**)。
 - コンテンツビューを使用しない場合、構造は **[org]-[product]** となります (例: **acme-osp13_containers-**)。
- **--tag-from-label {version}-{release}**: 各イメージの最新のタグを識別します。
- **-e**: オプションのサービスの環境ファイルを指定します。
- **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**: Ceph Storage を使用する場合は、Ceph Storage コンテナのイメージの場所を定義するための追加パラメーターを含めます。**ceph_image** には現在 Satellite 固有のプレフィックスが付いている点に注意してください。このプレフィックスは、**--prefix** オプションと同じ値です。以下に例を示します。

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

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

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

レジストリーの設定の準備が整いました。

4.7. 次のステップ

コンテナイメージのソースが記載された **overcloud_images.yaml** 環境ファイルができました。今後の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

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して、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使用するようにする必要があります。

前提条件

- 最新の OpenStack Platform リポジトリ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

- 登録パラメータの詳細については、『**Advanced Overcloud Customizations**』ガイドの「[Registering the Overcloud with an Environment File](#)」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、ロール固有のパラメーターに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
<code>controllerExtraConfig</code>	<code>ControllerExtraConfig</code>
<code>OvercloudControlFlavor</code>	<code>OvercloudControllerFlavor</code>
<code>controllerImage</code>	<code>ControllerImage</code>
<code>NovaImage</code>	<code>ComputeImage</code>
<code>NovaComputeExtraConfig</code>	<code>ComputeExtraConfig</code>
<code>NovaComputeServerMetadata</code>	<code>ComputeServerMetadata</code>
<code>NovaComputeSchedulerHints</code>	<code>ComputeSchedulerHints</code>
<code>NovaComputeIPs</code>	<code>ComputeIPs</code>
<code>SwiftStorageServerMetadata</code>	<code>ObjectStorageServerMetadata</code>
<code>SwiftStorageIPs</code>	<code>ObjectStorageIPs</code>
<code>SwiftStorageImage</code>	<code>ObjectStorageImage</code>
<code>OvercloudSwiftStorageFlavor</code>	<code>OvercloudObjectStorageFlavor</code>

カスタムの環境ファイルでこれらのパラメーターを更新します。

OpenStackPlatform 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** ファイルで使うことができます。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、非推奨となったパラメーターと、それに相当する Heat テンプレートのオプションを対照しています。

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
--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

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

オプション	説明	Heat テンプレートのパラメーター
--neutron-vni-ranges	テナントネットワークを割り当てるに使用できる VXLAN VNI の ID 範囲	NeutronVniRanges
--neutron-network-vlan-ranges	サポートされる Neutron ML2 および Open vSwitch VLAN マッピングの範囲。デフォルトでは、物理ネットワーク「datacentre」上の 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。デフォルトでは、「openvswitch」に設定されており、複数の値を指定するにはコンマ区切りの文字列を使用します。	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
--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
```

デフォルトの `/usr/share/openstack-tripleo-heat-templates/roles_data.yaml` ファイルをチェックして、構文のさらなる例を確認してください。ロールスニペットの例は、`/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
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します。
Keystone	InternalApi
Load Balancer	External、InternalApi、Storage、StorageMgmt、Tenant
Manila	InternalApi
Message Bus	InternalApi

ロール	必要なネットワーク
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。詳しくは、『**Advanced Overcloud Customization**』ガイドの「[Using Composable Networks](#)」を参照してください。

5.5. 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** インターフェー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ー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、**ExtraConfig** パラメーターがあるかどうかをチェックします。

```
$ 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**) が表示された場合には、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 Hierdata 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、新規インターフェースに変更します。

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 1:

```
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'
```

5.6. ネットワークインターフェー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ースの構造は **OS::Heat::StructuredConfig** リソースを使用してインターフェー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ー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ー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ー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、「[Isolating Networks](#)」を参照してください。

5.7. 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。次に「[6章 オ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。このコマンドのより、次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 に更新します。
- アップグレードに向けたノードの準備

手順

1. **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 \
  --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、コンテナイメージの環境ファイルには **-e** を使用するの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. **upgrade prepare** が完了するまで待ちます。

6.2. 全コントローラーノードのアップグレード

このプロセスでは、全コントローラーノードを OpenStack Platform 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 **--roles Controller** オプション

を指定して、操作をコントローラ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 Controller
```

- 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. コントローラノードのアップグレードが完了するまで待ちます。

6.3. 全コンピュートノードのアップグレード

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 **--roles 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 Compute
```

- 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. 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

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** コマンドを実行し、**--roles 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します。
- **openstack overcloud ceph-upgrade run** コマンドを実行し、コンテナ化された Red Hat Ceph Storage 3 クラスタへのアップグレードを実行します。

手順

1. **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 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、コンテナイメージの環境ファイルには **-e** を使用するの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. 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** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. 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 \  
  --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

以下の手順では、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グ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。以下のようなコマンドを実行します。

```
$ sudo systemctl restart neutron-openvswitch-agent
```

- 状況によっては、コントローラーノードの再起動後に 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
```