Red Hat OpenStack Platform 13

Fast Forward Upgrade

Red Hat OpenStack Platform 10 から 13 へのロングライフバージョン間のアップグレード

Last Updated: 2023-04-10

Red Hat OpenStack Platform 13 Fast Forward Upgrade

Red Hat OpenStack Platform 10 から 13 へのロングライフバージョン間のアップグレード

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドには、Fast Foward Upgrade のプロセスを記載しています。このプロセスは、OpenStack Platform 環境を 1 つのロングライフバージョンから次のロングライフバージョンにアップグレードします。今回、本書では Red Hat OpenStack Platform 10 (Newton) から 13 (Queens) へのアップグレードに重点を置いています。

目次

第1章 はじめに

本書では、Red Hat OpenStack Platform 環境を最新のロングライフバージョンにアップグレードするために役立つワークフローについて説明します。

1.1. 作業を開始する前に

以下の点に注意してください。

- バージョン 7 または 8 の Red Hat OpenStack Platform 環境を最初にデプロイした場合には、XFS ファイルシステムの古いバージョンが原因でアップグレードパスとコンテナー化されたサービスのデプロイに問題が生じる場合があることに注意してください。問題およびその解決方法についての詳細は、アーティクル XFS ftype=0 prevents upgrading to a version of OpenStack Director with containers 参照してください。

- デプロイメントに Red Hat Ceph Storage (RHCS) ノードが含まれる場合、各 Ceph オブジェクトストレージデーモン (OSD) の配置グループ (PG) の数は、デフォルトでは 250 を超えることができません。OSD ごとの PG 数が上限を超える Ceph ノードをアップグレードすると、警告状態になりアップグレードプロセスが失敗する可能性があります。アップグレードプロセスを開始する前に、OSD ごとの PG 数を増やすことができます。この問題の診断およびトラブルシューティングに関する詳細は、アーティクル OpenStack FFU from 10 to 13 times out when Ceph allocated in one or more OSDs more than 250 PGs を参照してください。

- **prevent_arp_spoofing** が False に設定されているすべてのポートを見つけます。これらのポートについて、ポートセキュリティーが無効になっていることを確認します。アップグレードの一環として、**prevent_arp_spoofing** オプションは削除され、その機能はポートセキュリティーによって制御されます。

- 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。

- デプロイされたオーバークラウド (アプリケーションのパスワードを含む) を手動で変更した場合、アップグレードの失敗を避けるために、これらの変更内容を使用して director デプロイメントテンプレートを更新する必要があります。ご不明な点がございましたら、 Red Hat テクニカルサポート までお問い合わせください。

1.2. FAST FORWARD UPGRADE

Red Hat OpenStack Platform には **Fast Forward Upgrade** 機能が実装されました。この機能は、複数のバージョンを経由するオーバークラウドのアップグレードパスを提供します。この機能は、**ロングライフバージョン** とされている特定の OpenStack のバージョンの使用を継続し、次のロングライフバージョンが提供された際にアップグレードする機会を提供することを目的としています。

本ガイドは、以下のバージョンの Fast Forward Upgrade パスを提供します。

旧バージョン	新バージョン
Red Hat OpenStack Platform 10	Red Hat OpenStack Platform 13

1.3. ワークフローの概要

以下の表には、Fast Forward Upgrade プロセスに必要なステップの概要と共に、アップグレードプロセスの各ステップに要する推定時間およびその影響をまとめています。

注記

以下の表に示す時間は内部テストに基づく最短の推定値であり、すべての実稼働環境には当てはまらない可能性があります。各タスクのアップグレード時間を正確に測定するには、実稼働環境と類似したハードウェアを持つテスト環境でこれらの手順を実施してください。

表1.1 Fast Forward Upgrade プロセスのステップ概要と影響

ステップ	説明	所要時間
環境の準備	アンダークラウドノー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、リブートします。環境を検証します。	このステップに要する時間は、デプロイメントのサイズにより異なる可能性があります。
アンダークラウドのアップグレード	OpenStack Platform 10 から OpenStack Platform 13 まで、アンダークラウドのバージョンを 1 つずつ順番にアップグレードします。	アンダークラウドアップグレードの推定時間は、約 60 分です。なお、アップグレード中、アンダークラウドにダウンタイムが発生します。 アンダークラウドのアップグレードステップ中、オーバークラウドは引き続き機能します。
コンテナーイメージの取得	さまざまな OpenStack 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。	コンテナーイメージソース設定の推定期間は、約 10 分です。
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。	アップグレードに向けたオーバークラウド準備の推定時間は、約 20 分です。
Fast Forward Upgrade の実行	OpenStack Platform director の最新のテンプレートセットを使用して、オーバークラウドプランをアップグレードします。パッケージとデータベースのバージョンを 1 つずつ順番にアップグレードして、データベーススキーマを OpenStack Platform 13 にアップグレードできる状態にします。	オーバークラウドのアップグレード実行の推定時間は、約 30 分です。なお、アップグレード中、オーバークラウドサービスにダウンタイムが発生します。 機能停止時間中 OpenStack の操作を行うことはできません。

ステップ	説明	所要時間
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。	コントローラーノードアップグレードの推定期間は、約 50 分です。 コントローラーノードのアップグレード中、オーバークラウドサービスに短時間のダウンタイムが発生します。
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。	コンピュートノードアップグレードの推定期間は、ノード 1 台につき約 25 分です。 コンピュートノードのアップグレード中、ワークロードのダウンタイムは予想されません。
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。	Ceph Storage ノードアップグレードの推定期間は、ノード 1 台につき約 25 分です。 Ceph Storage ノードのアップグレード中、ダウンタイムは予想されません。
アップグレードの最終処理	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。	オーバークラウドのコンバージ実行の推定時間は、最短でも 1 時間です。ただし、環境によってさらに時間がかかる場合があります。

1.4. アップグレード前の CEPH クラスターステータスの確認

環境をアップグレードする前に、Ceph クラスターがアクティブであり、想定どおりに機能していることを確認する必要があります。

手順

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

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

3. クラスターの健全性ステータスが **HEALTH_OK** であり、すべてのオブジェクトストレージデーモン (OSD) がアクティブであることを確認します。

第2章 OPENSTACK PLATFORM アップグレードの準備

このプロセスでは、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。

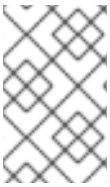
- アンダークラウドとオーバークラウドの両方をバックアップします。
- アンダークラウドを OpenStack Platform 10 の最新のマイナーバージョンに更新します (最新の Open vSwitch を含む)。
- 新しいカーネルまたはシステムパッケージがインストールされた場合には、アンダークラウドをリブートします。
- オーバークラウドを OpenStack Platform 10 の最新のマイナーバージョンに更新します (最新の Open vSwitch を含む)。
- 新しいカーネルまたはシステムパッケージがインストールされた場合には、オーバークラウドノードをリブートします。
- 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. ベアメタル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リストアできるように) 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 (SSL を使用している場合): **/var/lib/certmonger**
- コンテナーイメージデータ: **/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に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. データベースのバックアップを作成します。

```
[root@director ~]# mysqldump --opt --all-databases > /root/undercloud-all-databases.sql
```

3. **backup** ディレクトリーを作成して、そのディレクトリーを所有するユーザーを **stack** ユーザーに変更します。

```
[root@director ~]# mkdir /backup
[root@director ~]# chown stack: /backup
```

このディレクトリーを使用して、アンダークラウドのデータベースおよびファイルシステムを含むアーカイブを保存します。

4. **バックアップ** ディレクトリーに移動します。

```
[root@director ~]# cd /backup
```

5. 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。

```
[root@director ~]# tar --xattrs --xattrs-include='*.*' --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
```

- **--ignore-failed-read** オプションを指定すると、アンダークラウドに適用されないディレクトリーはスキップされます。
- **--xattrs** および **--xattrs-include='.'** オプションには、Object Storage (swift) および SELinux のメタデータを保存するために必要な拡張属性が含まれます。

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>** はシステムの日付になります。この **tar** ファイルを安全な場所にコピーします。

関連情報

- アンダークラウドのバックアップをリストアする必要がある場合には、付録A *アンダークラウドの復元*を参照してください。

2.2. 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、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、稼働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。コンピュート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. コントローラー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ーを作成します。

      ```
      # 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 -c /etc/puppet/hiera.yaml mysql::server::root_password)
      ```

   e. データベースのバックアップを作成します。

      ```
      # 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になります。このデータベースダンプを安全な場所にコピーします。

   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$MYSQLDBPASS -s -N -e "SELECT CONCAT('\"SHOW GRANTS FOR '',user,''@'',host,'';\"') FROM mysql.user where (length(user) > 0 and user NOT LIKE 'root')" | xargs -n1 mysql -uroot -p$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になります。このデータベースダンプを安全な場所にコピーします。

2. Pacemaker の設定をバックアップします。
   a. コントローラーノードにログインします。

b. 以下のコマンドを実行し、現在の Pacemaker 設定のアーカイブを作成します。

```
# sudo pcs config backup pacemaker_controller_backup
```

c. 作成されたアーカイブ (**pacemaker_controller_backup.tar.bz2**) を安全な場所にコピーします。

3. 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 のプライマリーインスタンスの IP を取得します。

```
# MONGOIP=$(sudo hiera -c /etc/puppet/hiera.yaml 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。

4. 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 -c /etc/puppet/hiera.yaml 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 -c /etc/puppet/hiera.yaml 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。

5. 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 --acls --ignore-failed-read --xattrs --xattrs-include='*.*' \
    -zcvf /var/tmp/filesystem_backup/`hostname`-filesystem-`date '+%Y-%m-%d-%H-%M-
%S'`.tar \
```

```
/etc \
/srv/node \
/var/log \
/var/lib/nova \
--exclude /var/lib/nova/instances \
/var/lib/glance \
/var/lib/keystone \
/var/lib/cinder \
/var/lib/heat \
/var/lib/heat-config \
/var/lib/heat-cfntools \
/var/lib/rabbitmq \
/var/lib/neutron \
/var/lib/haproxy \
/var/lib/openvswitch \
/var/lib/redis \
/var/lib/os-collect-config \
/usr/libexec/os-apply-config \
/usr/libexec/os-refresh-config \
/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。

c. 作成された **tar** ファイルを安全な場所にコピーします。

6. オーバークラウドで削除された行をアーカイブします。

a. アーカイブされた削除済みインスタンスを確認します。

```
$ source ~/overcloudrc
$ nova list --all-tenants --deleted
```

b. アーカイブされた削除済みインスタンスがない場合は、オーバークラウドのコントローラーノードの1つで以下のコマンドを入力して、削除済みインスタンスをアーカイブします。

```
# su - nova -s /bin/bash -c "nova-manage --debug db archive_deleted_rows --max_rows 1000"
```

削除されたすべてのインスタンスをアーカイブするまで、このコマンドを再実行します。

c. オーバークラウドのコントローラーノードの1つで以下のコマンドを入力して、アーカイブされた削除済みインスタンスをすべてパージします。

```
# su - nova -s /bin/bash -c "nova-manage --debug db purge --all --all-cells"
```

d. アーカイブされた削除済みインスタンスが残っていないことを確認します。

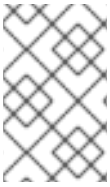
```
$ nova list --all-tenants --deleted
```

関連情報

- オーバークラウドのバックアップをリストアする必要がある場合には、付録B *オーバークラウドの復元*を参照してください。

2.3. 現在のアンダークラウドパッケージの OPENSTACK PLATFORM 10.Z の更新

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 環境の現行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。これは、**OpenStack Platform 10** 内でのマイナー更新です。



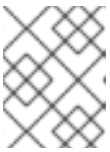
注記

このステップにより、アンダークラウドのオペレーティングシステムも Red Hat Enterprise Linux 7 の最新バージョンに更新され、Open vSwitch はバージョン 2.9 となります。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. 主要な OpenStack Platform サービスを停止します。

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```



注記

これにより、アンダークラウドで短時間のダウンタイムが生じます。アンダークラウドのアップグレード中もオーバークラウドは引き続き機能します。

3. RHEL のバージョンを RHEL 7.7 に設定します。

```
$ sudo subscription-manager release --set=7.7
```

4. **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
```

5. **openstack undercloud upgrade** コマンドを実行します。

```
$ openstack undercloud upgrade
```

6. コマンドの実行が完了するまで待ちます。
7. 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
```

8. ノードがブートするまで待ちます。
9. アンダークラウドに **stack** ユーザーとしてログインします。

アンダークラウドのパッケージの更新に加えて、オーバークラウドのイメージを最新の状態に維持して、イメージの設定が最新の **openstack-tripleo-heat-template** パッケージと同期することを推奨します。これにより、現在の準備段階と実際の Fast Foward Upgrade の間に実行されるデプロイメントとスケーリングの操作が正常に実行されるようになります。次の項では、このシナリオでイメージを更新する方法について説明します。環境を準備した直後に環境のアップグレードを行う予定の場合には、次の項はスキップできます。

2.4. NFV 対応環境の更新準備

お使いの環境でネットワーク機能仮想化 (NFV) が有効化されている場合には、アンダークラウドの更新後およびオーバークラウドの更新前に以下のステップを実行します。

手順

1. カスタムの環境ファイル (例: **network-environment.yaml**) で、vhost ユーザーソケットディレクトリーを変更します。

```
parameter_defaults:
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

2. **openstack overcloud deploy** コマンドに **ovs-dpdk-permissions.yaml** ファイルを追加して、qemu グループの設定値を OVS-DPDK 向けに **hugetlbfs** に設定します。

```
-e environments/ovs-dpdk-permissions.yaml
```

3. すべてのインスタンスの vHost ユーザーポートは、必ず **dpdkvhostuserclient** モードに設定してください。詳細は、Manually changing the vhost user port mode を参照してください。

2.5. 現在のオーバークラウドイメージの OPENSTACK PLATFORM 10.Z の更新

アンダークラウドの更新プロセスで、**rhosp-director-images** および **rhosp-director-images-ipa** パッケージから新規イメージアーカイブがダウンロードされる可能性があります。このプロセスにより、Red Hat OpenStack Platform 10内のアンダークラウドでそれらのイメージが更新されます。

前提条件

- アンダークラウドを現行バージョンの最新のマイナーリリースに更新済みであること

手順

1. **yum** ログをチェックして、新規イメージのアーカイブが利用可能かどうかを確認します。

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

2. 新規アーカイブが利用可能な場合には、現在のイメージを新規イメージに置き換えてください。新しいイメージをインストールするには、最初に **stack** ユーザーの **images** ディレクトリー (**/home/stack/images**) から既存のイメージを削除します。

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

3. アンダークラウドノードにおいて、source コマンドでアンダークラウドの認証情報を読み込みます。

```
$ source ~/stackrc
```

4. アーカイブを展開します。

```
$ cd ~/images
$ for i in /usr/share/rhosp-director-images/overcloud-full-latest-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

5. 最新のイメージを director にインポートして、ノードがこれらの新規イメージを使用するように設定します。

```
$ cd ~/images
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f csv --quote none | sed "1d" | paste -s -d " ")
```

6. 新規イメージがあるかどうかをチェックして、イメージ更新の最終処理を行います。

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

director は古いイメージを保持して、それらが更新された時のタイムスタンプを使用して名前を変更します。これらのイメージが必要でなくなったら、削除してください。

director が更新され、最新のイメージを使用するようになりました。この更新の後にはサービスを再起動する必要はありません。

アンダークラウドでは、更新された OpenStack Platform 10 のパッケージが使用されるようになりました。次にオーバークラウドを最新のマイナーリリースに更新します。

2.6. 現在のオーバークラウドパッケージの OPENSTACK PLATFORM 10.Z の更新

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 環境の現行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。これは、**Red Hat OpenStack Platform 10** 内でのマイナー更新です。

注記

このステップにより、オーバークラウドノードのオペレーティングシステムも Red Hat Enterprise Linux 7 の最新バージョンに更新され、Open vSwitch はバージョン 2.9 となります。

前提条件

- アンダークラウドを現行バージョンの最新のマイナーリリースに更新済みであること
- オーバークラウドのバックアップを実行済みであること

手順

1. サブスクリプション管理の設定で **rhel_reg_release** パラメーターを確認します。このパラメーターが設定されていない場合は、そのパラメーターを追加してバージョン 7.7 に設定する必要があります。

```
parameter_defaults:
  ...
  rhel_reg_release: "7.7"
```

オーバークラウドのサブスクリプション管理用環境ファイルに加えた変更を、必ず保存してください。

2. 元の **openstack overcloud deploy** コマンドに **--update-plan-only** オプションを追加して、現在のプランを更新します。以下に例を示します。

```
$ openstack overcloud deploy --update-plan-only \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/rhel-registration/environment-rhel-registration.yaml \
  [-e <environment_file>|...]
```

--update-plan-only のオプションを指定すると、director に保管されているオーバークラウドのプランのみが更新されます。**-e** オプションを使用して、オーバークラウドと関連のある環境ファイルとその更新パスを追加します。後で実行される環境ファイルで定義されているパラメーターとリソースが優先されることになるため、環境ファイルの順序は重要となります。以下の一覧は、環境ファイルの順序の例です。

- Heat テンプレートコレクションの初期化ファイル (**environments/network-isolation.yaml**) を含むネットワーク分離ファイルと、次にカスタムの NIC 設定ファイル
- 外部のロードバランシングの環境ファイル
- ストレージの環境ファイル
- Red Hat CDN または Satellite 登録用の環境ファイル
- その他のカスタム環境ファイル

3. オーバークラウドの静的なインベントリーファイルを作成します。

```
$ tripleo-ansible-inventory --ansible_ssh_user heat-admin --static-yaml-inventory
~/inventory.yaml
```

デフォルトのオーバークラウド名 **overcloud** 以外のオーバークラウド名を使用する場合は、**--plan** オプションを使用して実際のオーバークラウドの名前を設定します。

4. すべてのノードで、オペレーティングシステムのバージョンを Red Hat Enterprise Linux 7.7 に設定するタスクが含まれる Playbook を作成します。

```
$ cat > ~/set_release.yaml <<'EOF'
- hosts: all
  gather_facts: false
  tasks:
    - name: set release to 7.7
```

```
    command: subscription-manager release --set=7.7
    become: true
EOF
```

5. set_release.yaml Playbook を実行します。

```
$ ansible-playbook -i ~/inventory.yaml -f 25 ~/set_release.yaml --limit
undercloud,Controller,Compute
```

すべての Red Hat OpenStack Platform ノードにコンテンツを適用するには、**--limit** オプションを使用します。

6. **openstack overcloud update** コマンドを使用して、全ノードでパッケージの更新を実行します。

```
$ openstack overcloud update stack -i overcloud
```

-i のオプションを指定すると、各ノードは対話モードで順次に更新されます。更新プロセスによりノードの更新が完了すると、スクリプトにより、確認のためのブレークポイントが提供されます。**-i** オプションを指定しないと、最初のブレークポイントで更新が一時停止されたままになります。したがって、**-i** オプションを含めることが必須です。

スクリプトは以下の機能を実行します。

a. スクリプトはノード上で 1 つずつ実行します。

   i. コントローラーノードの場合は、これにより全パッケージが更新されます。

   ii. その他のノードの場合には、これにより Puppet モジュールのみが更新されます。

b. Puppet は全ノードで一度に実行されます。

   i. コントローラーノードの場合には、Puppet 実行により設定が同期されます。

   ii. その他のノードの場合には、Puppet 実行により残りのパッケージが更新され、設定が同期されます。

7. 更新のプロセスが開始します。このプロセス中に、director は **IN_PROGRESS** のステータスを報告して、ブレークポイントをクリアするように定期的に要求します。以下に例を示します。

```
starting package update on stack overcloud
IN_PROGRESS
IN_PROGRESS
WAITING
on_breakpoint: [u'overcloud-compute-0', u'overcloud-controller-2', u'overcloud-controller-1',
u'overcloud-controller-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear 49913767-e2dd-4772-
b648-81e198f5ed00), no=cancel update, C-c=quit interactive mode:
```

Enter を押すと、**on_breakpoint** 一覧の最後のノードからブレークポイントをクリアします。これで、そのノードの更新が開始します。

8. スクリプトはノードの更新順序を自動的に事前定義します。

- 各コントローラーノードを個別に事前定義

- 各コンピュートノードを個別に事前定義
- 各 Ceph Storage ノードを個別に事前定義
- その他の全ノードを個別に事前定義

更新を成功させるには、特に以下の順序で作業を行うことを推奨します。

a. 各コントローラーノードのブレークポイントを個別にクリアします。更新後にノードのサービスを再起動する必要がある場合のために、各コントローラーノードには、個別のパッケージ更新が必要です。これにより、他のコントローラーノードの高可用性サービスの中断が抑えられます。

b. コントローラーノードの更新後に、各コンピュートノードのブレークポイントをクリアします。また、特定のノード上でコンピュートノードの名前をタイプしてブレークポイントをクリアしたり、Python ベースの正規表現を使用して複数のコンピュートノード上のブレークポイントを一度にクリアしたりすることもできます。

c. 各 Ceph Storage ノードのブレークポイントをクリアします。また、特定のノード上で Ceph Storage ノードの名前をタイプしてブレークポイントをクリアしたり、Python ベースの正規表現を使用して複数の Ceph Storage ノード上のブレークポイントを一度にクリアしたりすることもできます。

d. 残りのブレークポイントをクリアして、その他のノードを更新します。特定のノードでノード名をタイプしてブレークポイントをクリアしたり、Python ベースの正規表現を使用して複数のノード上のブレークポイントを一度にクリアしたりすることもできます。

e. すべてのノードの更新が完了するまで待ちます。

9. 更新が完了すると、コマンドにより **COMPLETE** のステータスが報告されます。

```
...
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

10. コントローラーノードにフェンシングを設定している場合には、更新プロセスによってその設定が無効になる場合があります。更新プロセスが完了したら、コントローラーノードの 1 つで以下のコマンドを実行してフェンシングを再度有効にします。

```
$ sudo pcs property set stonith-enabled=true
```

更新プロセスを実行しても、オーバークラウド内のノードは自動的には再起動しません。カーネルおよびその他のシステムッケージを更新した場合には、リブートが必要です。各ノードの **/var/log/yum.log** ファイルをチェックして、**kernel** または **openvswitch** パッケージのメジャーまたはマイナーバージョンが更新されているかどうかを確認します。更新されている場合には、以下の手順に従って各ノードをリブートします。

2.7. 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

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。

2.8. 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
```

2.9. 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
```

2.10. システムパッケージの確認

アップグレードの前には、アンダークラウドノードと全オーバークラウドノードが、以下のパッケージの最新バージョンを使用している必要があります。

パッケージ	バージョン
openvswitch	最小 2.9
qemu-img-rhev	最小 2.10
qemu-kvm-common-rhev	最小 2.10
qemu-kvm-rhev	最小 2.10
qemu-kvm-tools-rhev	最小 2.10

手順

1. ノードにログインします。
2. **yum** を実行して、システムパッケージを確認します。

```
$ sudo yum list qemu-img-rhev qemu-kvm-common-rhev qemu-kvm-rhev qemu-kvm-tools-rhev openvswitch
```

3. **ovs-vsctl** を実行して、現在実行中のバージョンを確認します。

```
$ sudo ovs-vsctl --version
```

4. すべてのノードでこれらのステップを繰り返します。

アンダークラウドは、更新された OpenStack Platform 10 パッケージを使用するようになりました。次の手順で、システムが稼動状態かどうかを確認します。

2.11. OPENSTACK PLATFORM 10 アンダークラウドの検証

Red Hat OpenStack Platform 10のアンダークラウドをアップグレードする前に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。

```
$ 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。

```
$ 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。

```
$ 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。

```
$ 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。

```
$ 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。

2.12. OPENSTACK PLATFORM 10 オーバークラウドの検証

Red Hat OpenStack Platform 10のオーバークラウドをアップグレードする前に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。

```
$ 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。

```
$ 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. 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。

```
$ 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 /etc/haproxy/haproxy.cfg'
```

5. 前の手順で取得した接続および認証情報を使用して、RHOSP サービスの接続ステータスを確認します。
SSL が有効になっていない場合は、次の cURL リクエストでこれらの詳細を使用します。

```
$ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

SSL が有効になっている場合は、次の cURL リクエストでこれらの詳細を使用します。

```
curl -s -u admin:<PASSWORD> "https://<HOSTNAME>:1993/;csv" | egrep -vi "
(frontend|backend)" | awk -F',' '{ print $1" "$2" "$18 }'
```

<PASSWORD> および **<IP ADDRESS>** または **<HOSTNAME>** の値は、**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 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。

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。

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

7. RabbitMQ クラスターの正常性を確認します。

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

8. Pacemaker リソースの正常性を確認します。

```
$ 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。

```
$ 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。

```
$ 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。

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

重要

各 Ceph オブジェクトストレージデーモン (OSD) の配置グループ (PG) の数は、デフォルトでは 250 を超えることができません。OSD ごとの PG 数が上限を超える Ceph ノードをアップグレードすると、警告状態になりアップグレードプロセスが失敗する可能性があります。アップグレードプロセスを開始する前に、OSD ごとの PG 数を増やすことができます。この問題の診断およびトラブルシューティングに関する詳細は、アーティクル OpenStack FFU from 10 to 13 times out when Ceph allocated in one or more OSDs more than 250 PGs を参照してください。

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。

```
$ 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。

```
$ 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。

```
$ 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。

```
$ 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。

```
$ 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設定を調整する方法が記載されています。
- Database Size Management for Red Hat Enterprise Linux OpenStack Platform の記事を参照して、オーバークラウド上の OpenStack Platform サービスの未使用のデータベースレコードをチェックしてクリーニングします。

2.13. NFV 対応環境の更新の最終処理

お使いの環境でネットワーク機能仮想化 (NFV) が有効化されている場合には、アンダークラウドとオーバークラウドを更新した後に、以下のステップを実行する必要があります。

手順

既存の OVS-DPDK インスタンスを移行して、OVS ポートで vhost ソケットモードが **dkdpvhostuser** から **dkdpvhostuserclient** に変わることを確認します。既存のインスタンスのスナップショットを作成して、そのスナップショットイメージをベースに新規インスタンスを再ビルドすることを推奨します。インスタンスのスナップショットに関する詳細は、インスタンスのスナップショットの管理 を参照してください。

インスタンスのスナップショットを作成して、そのスナップショットから新規インスタンスを起動するには、以下の手順を実行します。

1. オーバークラウドのアクセス情報を読み込みます。

```
$ source ~/overcloudrc
```

2. スナップショットを作成するインスタンスのサーバー ID を確認します。

```
$ openstack server list
```

3. スナップショットを作成する前に、元のインスタンスをシャットダウンして、全データがディスクにフラッシュされるようにしてください。

```
$ openstack server stop SERVER_ID
```

4. インスタンスのスナップショットイメージを作成します。

```
$ openstack image create --id SERVER_ID SNAPSHOT_NAME
```

5. このスナップショットイメージで新規インスタンスを起動します。

```
$ openstack server create --flavor DPDK_FLAVOR --nic net-id=DPDK_NET_ID--image
SNAPSHOT_NAME INSTANCE_NAME
```

6. オプションとして、新規インスタンスのステータスが **ACTIVE** であることを確認します。

```
$ openstack server list
```

スナップショットを作成する必要のある全インスタンスでこの手順を繰り返してから、再起動します。

2.14. YUM 履歴の保持

オーバークラウドのマイナー更新が完了したら、**yum** 履歴を保持します。ロールバック操作のために yum トランザクションを元に戻す必要がある場合に、この情報が役立ちます。

手順

1. それぞれのノードで以下のコマンドを実行して、ノードでの全 **yum** 履歴をファイルに保存します。

```
$ sudo yum history list all > /home/heat-admin/$(hostname)-yum-history-all
```

2. それぞれのノードで以下のコマンドを実行して、最後の **yum** 履歴項目の ID を保存します。

```
$ sudo yum history list all | head -n 5 | tail -n 1 | awk '{print $1}' > /home/heat-
admin/$(hostname)-yum-history-all-last-id
```

3. これらのファイルを安全な場所にコピーします。

2.15. 次のステップ

準備段階が完了したので、次に3章 *アンダークラウドのアップグレード*に記載のステップに従って、アンダークラウドを 10 から 13 にアップグレードします。

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、アンダークラウドを **Red Hat OpenStack Platform 13**にアップグレードします。これは、OpenStack Platform 10 から OpenStack Platform 13 までのアンダークラウドのバージョンを 1 つずつ順番にアップグレードしていくことによって実行します。

3.1. アンダークラウドを OPENSTACK PLATFORM 11 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 11** 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-10-rpms
```

3. 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-11-rpms
```

4. オーバークラウドのベースイメージの更新を無効にします。

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. 主要な OpenStack Platform サービスを停止します。

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```

注記

これにより、アンダークラウドで短時間のダウンタイムが生じます。アンダークラウドのアップグレード中もオーバークラウドは引き続き機能します。

6. デフォルトのプロビジョニング/コントロールプレーンネットワークが **192.0.2.0/24** から **192.168.24.0/24** に変わりました。以前の **undercloud.conf** ファイルで、デフォルトのネットワーク値を使用していた場合には、プロビジョニング/コントロールプレーンネットワークは **192.0.2.0/24** に設定されます。これは、**192.0.2.0/24** ネットワークを引き続き使用するには、**undercloud.conf** ファイルに特定のパラメーターを設定する必要があることを意味します。これらのパラメーターは以下のとおりです。
   - **local_ip**
   - **network_gateway**
   - **undercloud_public_vip**
   - **undercloud_admin_vip**
   - **network_cidr**

- **masquerade_network**
- **dhcp_start**
- **dhcp_end**

ネットワークの値を **undercloud.conf** に設定して、今後アップグレードを実行する際に **192.0.2.0/24** CIDR を引き続き使用するようにします。**openstack undercloud upgrade** コマンドを実行する前に、ネットワークの設定が正しく設定されていることを確認してください。

7. **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 instack-undercloud openstack-puppet-modules openstack-tripleo-common python-tripleoclient
```

8. 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
```

9. 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。

アンダークラウドを **OpenStack Platform 11** リリースにアップグレードする手順が完了しました。

3.2. アンダークラウドを OPENSTACK PLATFORM 12 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 12** 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-11-rpms
```

3. 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-12-rpms
```

4. オーバークラウドのベースイメージの更新を無効にします。

```
$ sudo yum-config-manager --setopt=exclude=rhosp-director-images* --save
```

5. **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
```

6. **/home/stack/undercloud.conf** ファイルを編集して、**enabled_drivers** パラメーターに **pxe_ssh** ドライバーが含まれていないことを確認します。Virtual Baseboard Management Controller (VBMC) が推奨されるようになったため、このドライバーは非推奨となり、Red Hat

OpenStack Platform から削除されました。この新しいドライバーと移行手順の詳細は、**director のインストールと使用方法**の付録 Virtual Baseboard Management Controller (VBMC) を参照してください。

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。

アンダークラウドが **OpenStack Platform 12** リリースにアップグレードされました。

3.3. 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。

3.4. アンダークラウドでの非推奨サービスの無効化

アンダークラウドをアップグレードしたら、非推奨の **openstack-glance-registry** および **mongod** サービスを無効にする必要があります。

手順

1. アンダークラウドに **stack** ユーザーとしてログインします。
2. **openstack-glance-registry** サービスを停止して無効にします。

```
$ sudo systemctl stop openstack-glance-registry
$ sudo systemctl disable openstack-glance-registry
```

3. **mongod** サービスを停止して無効にします。

```
$ sudo systemctl stop mongod
$ sudo systemctl disable mongod
```

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。これでコンテナーイメージのソースを設定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。

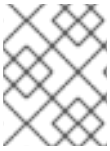
注記

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。

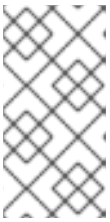
手順

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。

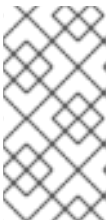
```
$ while read IMAGE; do \
  IMAGENAME=$(echo $IMAGE | cut -d"/" -f2 | sed "s/openstack-//g" | sed "s/:.*//g") ; \
  hammer repository create \
  --organization "ACME" \
  --product "OSP13 Containers" \
  --content-type docker \
```

```
  --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、アップグレードに向けてオーバークラウドを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

本項では、アップグレードのプロセスに備えてオーバークラウドを準備します。本項のすべてのステップが、お使いのオーバークラウドに適用されるわけではありません。ただし、各ステップをチェックして、アップグレードのプロセスが開始する前にオーバークラウドで追加の設定が必要かどうかを確認しておくことを推奨します。

5.1. オーバークラウドサービスのダウンタイムの準備

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

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

オーバークラウドのアップグレードによる影響を受ける項目

- OpenStack Platform サービス

オーバークラウドのアップグレードによる影響を受けない項目

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

5.2. アップグレードテスト用のコンピュートノードの選択

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

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

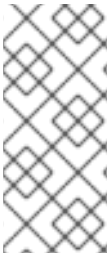
オーバークラウドのアップグレードプロセスを円滑にするには、全コンピュートノードをアップグレードする前に、環境内にある個々のコンピュートノードのいくつかでアップグレードをテストすると役立ちます。これにより、アップグレード中に大きな問題が発生しなくなり、ワークロードのダウンタイムを最小限に抑えることができます。

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

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

6章*オーバークラウドのアップグレード*の手順では、全コンピュートノードでアップグレードを実行する前の、アップグレードプロセスのテスト用のコンピュートノードの例として、**compute-0** を使用しています。

次のステップでは、**roles_data** ファイルを更新して、新しいコンポーザブルサービスがお使いの環境内の適切なロールに追加されるようにします。既存の **roles_data** ファイルを手動で編集するには、以下に記載する OpenStack Platform 13 のロール向けの新規コンポーザブルサービスの一覧を使用してください。



注記

Red Hat OpenStack Platform 12 以前のバージョンでコンピュートインスタンス向けの高可用性 (インスタンス HA) を有効化していて、バージョン 13 以降のバージョンへの Fast Forward Upgrade を実行する場合には、最初にインスタンス HA を手動で無効にする必要があります。手順については、以前のバージョンからのインスタンス HA の無効化 を参照してください。

5.3. 新規コンポーザブルサービス

Red Hat OpenStack Platform の今回のバージョンには、新たなコンポーザブルサービスが含まれています。独自のロールにカスタムの **roles_data** ファイルを使用する場合には、これらの新しい必須サービスを該当するロールに追加してください。

全ロール

以下の新規サービスは全ロールに適用されます。

OS::TripleO::Services::MySQLClient

他のコンポーザブルサービス用のデータベース設定を提供する MariaDB クライアントをノード上で設定します。このサービスは、スタンドアロンのコンポーザブルサービスを使用する全ロールに追加してください。

OS::TripleO::Services::CertmongerUser

オーバークラウドが Certmonger から証明書を要求できるようにします。TLS/SSL 通信を有効にしている場合にのみ使用されます。

OS::TripleO::Services::Docker

コンテナー化されたサービスを管理するために **docker** をインストールします。

OS::TripleO::Services::ContainersLogrotateCrond

コンテナーログ用の **logrotate** サービスをインストールします。

OS::TripleO::Services::Securetty

ノード上で **securetty** を設定できるようにします。**environments/securetty.yaml** 環境ファイルで有効化済みです。

OS::TripleO::Services::Tuned

Linux のチューニングデーモン (**tuned**) を有効化して設定します。

OS::TripleO::Services::AuditD

auditd デーモンを追加して、ルールを設定します。デフォルトでは無効になっています。

OS::TripleO::Services::Collectd

collectd デーモンを追加します。デフォルトでは無効になっています。

OS::TripleO::Services::Rhsm

Ansible ベースの方法を使用してサブスクリプションを設定します。デフォルトでは無効になっています。

OS::TripleO::Services::RsyslogSidecar

ロギング用のサイドカーコンテナーを設定します。デフォルトでは無効になっています。

特定のロール

以下の新規サービスは特定のロールに適用されます。

OS::TripleO::Services::NovaPlacement

OpenStack Compute (nova) Placement API を設定します。現在のオーバークラウドでスタンドアロンの Nova API ロールを使用している場合には、そのロールにこのサービスを追加します。そうでない場合には、このサービスを Controller ロールに追加してください。

OS::TripleO::Services::PankoApi

OpenStack Telemetry Event Storage (panko) サービスを設定します。現在のオーバークラウドでスタンドアロンの Telemetry ロールを使用している場合には、このサービスをそのロールに追加します。そうでない場合には、このサービスを Controller ロールに追加してください。

OS::TripleO::Services::Clustercheck

Controller またはスタンドアローンの **Database** ロールなどの **OS::TripleO::Services::MySQL** サービスも使用するロールに必要です。

OS::TripleO::Services::Iscsid

Controller ロール、**Compute** ロール、**BlockStorage** ロールで、**iscsid** サービスを設定します。

OS::TripleO::Services::NovaMigrationTarget

コンピュート ノード上で移行ターゲットサービスを設定します。

OS::TripleO::Services::Ec2Api

コントローラー ノードで OpenStack Compute (nova) EC2-API サービスを有効化します。デフォルトでは無効になっています。

OS::TripleO::Services::CephMgr

コントローラー ノードで Ceph Manager サービスを有効にします。 **ceph-ansible** 設定の一部として有効化されています。

OS::TripleO::Services::CephMds

コントローラー ノードで Ceph Metadata Service (MDS) を有効化します。デフォルトでは無効になっています。

OS::TripleO::Services::CephRbdMirror

RADOS Block Device (RBD) ミラーリングサービスを有効化します。デフォルトでは無効になっています。

上記に加えて、特定のカスタムロール向けサービスの最新の一覧は、**オーバークラウドの高度なカスタマイズ**の サービスアーキテクチャー: スタンドアロンロール の項を参照してください。

新規コンポーザブルサービスに加えて、OpenStack Platform 13 以降で非推奨になったサービスについても注意してください。

5.4. 非推奨のコンポーザブルサービス

カスタムの **roles_data** ファイルを使用する場合には、該当するロールから以下のサービスを削除してください。

OS::TripleO::Services::Core

このサービスは、その他の Pacemaker サービスのコア依存関係として機能していました。このサービスは、高可用性コンポーザブルサービスに対応するために削除されました。

OS::TripleO::Services::VipHosts

このサービスは、ノードのホスト名と IP アドレスで /etc/hosts ファイルを設定していました。このサービスは、director の Heat テンプレートに直接統合されるようになりました。

OS::TripleO::Services::FluentdClient

このサービスは、**OS::TripleO::Services::Fluentd** サービスに置き換えられました。

OS::TripleO::Services::ManilaBackendGeneric

Manila の汎用バックエンドはサポートされなくなりました。

カスタムの **roles_data** ファイルを使用する場合には、各ロールから以下のサービスを削除してください。

上記に加えて、特定のカスタムロール向けサービスの最新の一覧は、**オーバークラウドの高度なカスタマイズ**の サービスアーキテクチャー: スタンドアロンロール の項を参照してください。

5.5. コンテナー化されたサービスへの切り替え

Fast Forward Upgrade プロセスにより、特定の Systemd サービスがコンテナー化されたサービスに変換されます。**/usr/share/openstack-tripleo-heat-templates/environments/** からのデフォルトの環境ファイルを使用する場合には、この変換は自動的に行われます。

カスタム環境ファイルを使用してオーバークラウドのサービスを有効にする場合には、環境ファイルの **resource_registry** セクションで、登録したコンポーザブルサービスがすべてコンポーザブルサービスにマッピングされていることを確認します。

手順

1. カスタム環境ファイルを表示します。

```
$ cat ~/templates/custom_environment.yaml
```

2. ファイルコンテンツの **resource_registry** セクションを確認します。
3. **resource_registry** セクションのコンポーザブルサービスを確認します。以下の名前空間を使用するコンポーザブルサービス。

```
OS::TripleO::Services
```

たとえば、以下のコンポーザブルサービスは、OpenStack Bare Metal サービス (ironic) API に関するものです。

```
OS::TripleO::Services::IronicApi
```

4. コンポーザブルサービスが Puppet 固有の Heat テンプレートにマッピングされているかどうかを確認します。以下に例を示します。

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/puppet/services/ironic-api.yaml
```

5. コンテナー化バージョンの Heat テンプレートが **/usr/share/openstack-triple-heat-template/docker/services/** にあるかどうかを確認し、サービスをコンテナー化バージョンに再マッピングします。

```
resource_registry:
  OS::TripleO::Services::IronicApi: /usr/share/openstack-triple-heat-
template/docker/services/ironic-api.yaml
```

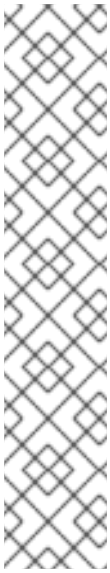
あるいは、**/usr/share/openstack-tripleo-heat-templates/environments/** にあるサービスの更新された環境ファイルを使用します。たとえば、OpenStack Bare Metal サービス (ironic) を有効にする最新の環境ファイルは **/usr/share/openstack-tripleo-heat-templates/environments/services/ironic.yaml** で、ここにはコンテナー化されたサービスへのマッピングが含まれています。

カスタムサービスがコンテナー化されたサービスを使用しない場合には、マッピングを Puppet 固有の Heat テンプレートのままにします。

5.6. 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**

旧パラメーター	新規パラメーター
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**
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"**

旧パラメーターおよび値	新規パラメーターおよび値
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
  ...
```

5.7. 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**
--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**

オプション	説明	Heat テンプレートのパラメーター
--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**
--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**

オプション	説明	Heat テンプレートのパラメーター
--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
--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。

これらの新たなパラメーターを含んだ **deprecated_cli_options.yaml** ファイルの例を以下に示します。

```
parameter_defaults:
  ControllerCount: 3
  ComputeCount: 3
  CephStorageCount: 3
  ...
```

本ガイドの後半には、これらの新しいパラメーターを含む **deprecated_cli_options.yaml** 環境ファイルの例を記載しています。

5.8. 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**

ロール	必要なネットワーク
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。
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。詳しい情報は、**オーバークラウドの高度なカスタマイズ**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。

5.9. CEPH STORAGE または HCI ノードのアップグレードの準備

コンテナー化されたサービスにアップグレードされたため、Ceph Storage ノードのインストールと更新の方法が変わりました。Ceph Storage の設定では、**ceph-ansible** パッケージ内の Playbook のセットを使用するようになりました。このパッケージはアンダークラウドにインストールします。

重要な影響

- 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。

手順

1. director の管理する Ceph Storage クラスターまたは外部の Ceph Storage クラスターを使用している場合には、**ceph-ansible** パッケージをインストールします。
   a. アンダークラウドで Ceph Tools リポジトリーを有効にします。

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

   b. **ceph-ansible** パッケージをアンダークラウドにインストールします。

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

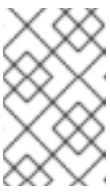
2. Ceph 固有の環境ファイルを確認し、Ceph 固有の heat リソースがコンテナー化されたサービスを使用する状態にします。
   - director が Ceph Storage クラスターを管理する場合には、**resource_register** のリソースが **docker/services/ceph-ansible** のテンプレートをポイントする状態にします。

   ```
   resource_registry:
     OS::TripleO::Services::CephMgr: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-mgr.yaml
     OS::TripleO::Services::CephMon: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-mon.yaml
     OS::TripleO::Services::CephOSD: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-osd.yaml
     OS::TripleO::Services::CephClient: /usr/share/openstack-tripleo-heat-templates/docker/services/ceph-ansible/ceph-client.yaml
   ```

   重要

   この設定は、**/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** の環境ファイルに記載されています。このファイルは、**-e** を使用して今後すべてのデプロイメントコマンドに追加することができます。

   注記

   環境で使用する環境ファイルまたはテンプレートファイルが **/usr/share** ディレクトリーにない場合は、ファイルへの絶対パスを含める必要があります。

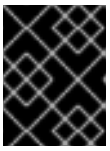
   - 外部 Ceph Storage クラスターの場合には、**resource_register** のリソースが **docker/services/ceph-ansible** のテンプレートをポイントする状態にします。

```
resource_registry:
  OS::TripleO::Services::CephExternal: /usr/share/openstack-tripleo-heat-
templates/docker/services/ceph-ansible/ceph-external.yaml
```

重要

この設定は、**/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml** の環境ファイルに記載されています。このファイルは、**-e** を使用して今後すべてのデプロイメントコマンドに追加することができます。

3. director が管理する Ceph Storage クラスターの場合には、新しい **CephAnsibleDisksConfig** パラメーターを使用して、ディスクのマッピングの方法を定義します。以前のバージョンの Red Hat OpenStack Platform では、**ceph::profile::params::osds** hieradata を使用して OSD レイアウトを定義していました。この hieradata を **CephAnsibleDisksConfig** パラメーターの設定に変換します。以下の例で、Ceph ジャーナルディスクが共存する場合と共存しない場合について、hieradata を **CephAnsibleDisksConfig** パラメーターの設定に変換する方法を説明します。



重要

osd_scenario を設定する必要があります。**osd_scenario** を設定しないままにすると、デプロイメントに失敗する場合があります。

- Ceph ジャーナルディスクが共存するケースで、hieradata が以下のようであれば、

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb': {}
      '/dev/sdc': {}
      '/dev/sdd': {}
```

CephAnsibleDisksConfig パラメーターを使用して、以下のように hieradata を変換し、**ceph::profile::params::osds** を **{}** に設定します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    journal_size: 512
    osd_scenario: collocated
  ExtraConfig:
      ceph::profile::params::osds: {}
```

- ジャーナルがより高速な専用のデバイスにあり共存しないケースで、hieradata が以下のようであれば、

```
parameter_defaults:
  ExtraConfig:
```

```
  ceph::profile::params::osd_journal_size: 512
  ceph::profile::params::osds:
    '/dev/sdb':
      journal: '/dev/sdn'
    '/dev/sdc':
      journal: '/dev/sdn'
    '/dev/sdd':
      journal: '/dev/sdn'
```

CephAnsibleDisksConfig パラメーターを使用して、以下のように hieradata を変換し、**ceph::profile::params::osds** を **{}** に設定します。

```
parameter_defaults:
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    dedicated_devices:
    - /dev/sdn
    - /dev/sdn
    - /dev/sdn
    journal_size: 512
    osd_scenario: non-collocated
  ExtraConfig:
    ceph::profile::params::osds: {}
```

ceph-ansible に使用する OSD ディスクレイアウトオプションの完全な一覧は、**/usr/share/ceph-ansible/group_vars/osds.yml.sample** のサンプルファイルを参照してください。

4. 今後のデプロイメントコマンドでは、**-e** オプションを使用して新しい Ceph の設定環境ファイルを指定します。これには以下のファイルが含まれます。

   - director の管理する Ceph Storage の場合:
     - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml**
     - Ansible ベースのディスクマッピングが含まれる環境ファイル
     - Ceph Storage のカスタマイズに関するその他の環境ファイル
   - 外部 Ceph Storage の場合:
     - **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible-external.yaml**
     - Ceph Storage のカスタマイズに関するその他の環境ファイル

5.10. ディスク設定が異なる CEPH または HCI ノードの環境変数の更新

HCI ノードの場合には、Compute サービスのアップグレードにはディスク定義に古い構文を使用し、ストレージサービスのアップグレードにはディスク定義に新しい構文を使用します。「Ceph Storage または HCI ノードのアップグレードの準備」を参照してください。ただし、ディスク設定が異なる場合も構文を更新しなければならない場合があります。

アップグレードするノードのディスクが同一ではない場合には、異なるディスク設定となります。たとえば、HCI ノードと Ceph Storage ノードが混在するケースでは、ディスク設定が異なります。

OpenStack Platform 12 から ceph-ansible が使用されるようになり、ディスク設定が異なる混在ノードを更新する際の構文が変更されています。つまり、OpenStack Platform 12 以降、ディスクを定義するために **RoleExtraConfig** のコンポーザブルロール構文を使用することはできません。以下の例を参照してください。

以下の例は、OpenStack Platform 12 以降では機能しません。

```
CephStorageExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
    '/dev/sdc'
    '/dev/sdd'

ComputeHCIExtraConfig:
  ceph::profile::params::osds:
    '/dev/sda'
    '/dev/sdb'
```

OpenStack Platform 12 以降は、アップグレードの前にテンプレートを更新する必要があります。異種ディスク設定のテンプレート更新方法に関する詳細は、**コンテナー化された Red Hat Ceph を持つオーバークラウドのデプロイ** の 異なる設定の Ceph Storage ノードへのディスクレイアウトのマッピング を参照してください。

5.11. 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
```

5.12. ストレージバックエンドの準備

一部のストレージバックエンドは、設定フックではなく、独自のコンポーザブルサービスを使用するように変更されました。カスタムストレージバックエンドを使用する場合には、**environments** ディレクトリーで関連する環境ファイルに新規パラメーターとリソースが含まれているかどうかを確認してください。バックエンド用のカスタム環境ファイルを更新します。以下に例を示します。

- NetApp Block Storage (cinder)バックエンドの場合は、デプロイメント内の新しい **environments/cinder-netapp-config.yaml** を使用してください。
- Dell EMC Block Storage (cinder)バックエンドの場合は、デプロイメント内の新しい **environments/cinder-dellsc-config.yaml** を使用してください。
- Dell EqualLogic Block Storage (cinder)バックエンドの場合は、デプロイメント内の新しい **environments/cinder-dellps-config.yaml** を使用してください。

たとえば、NetApp Block Storage (cinder)バックエンドは、それぞれのバージョンに以下のリソースを使用していました。

- OpenStack Platform 10 以前: **OS::TripleO::ControllerExtraConfigPre: ../puppet/extraconfig/pre_deploy/controller/cinder-netapp.yaml**
- OpenStack Platform 11 以降: **OS::TripleO::Services::CinderBackendNetApp: ../puppet/services/cinder-backend-netapp.yaml**

今回の変更の結果、このバックエンドには新しい **OS::TripleO::Services::CinderBackendNetApp** リソースと、関連付けられたサービステンプレートを使用するようになりました。

5.13. SSL/TLS を介してアンダークラウドのパブリック API にアクセスするための準備

オーバークラウドは、アップグレード中にアンダークラウドの OpenStack Object Storage (swift) のパブリック API にアクセスする必要があります。アンダークラウドで自己署名証明書を使用している場合には、アンダークラウドの認証局を各オーバークラウドノードに追加する必要があります。

前提条件

- アンダークラウドで、パブリック API に SSL/TLS を使用していること

手順

1. director の動的な Ansible スクリプトが OpenStack Platform 12 バージョンに更新され、オーバークラウドプラン内の **RoleNetHostnameMap** Heat パラメーターを使用してインベントリーを定義するようになりました。ただし、オーバークラウドは現在 OpenStack Platform 11 のテンプレートバージョンを使用しており、これには **RoleNetHostnameMap** パラメーターがありません。これは、一時的な静的インベントリーファイルを作成する必要があることを意味します。このファイルは、以下のコマンドを実行すると生成することができます。

```
$ openstack server list -c Networks -f value | cut -d"=" -f2 > overcloud_hosts
```

2. 以下の内容を記述した Ansible Playbook (**undercloud-ca.yml**) を作成します。

```
---
- name: Add undercloud CA to overcloud nodes
  hosts: all
```

```
user: heat-admin
become: true
vars:
  ca_certificate: /etc/pki/ca-trust/source/anchors/cm-local-ca.pem
tasks:
  - name: Copy undercloud CA
    copy:
      src: "{{ ca_certificate }}"
      dest: /etc/pki/ca-trust/source/anchors/
  - name: Update trust
    command: "update-ca-trust extract"
  - name: Get the swift endpoint
    shell: |
      sudo hiera swift::keystone::auth::public_url | awk -F/ '{print $3}'
    register: swift_endpoint
    delegate_to: 127.0.0.1
    become: yes
    become_user: stack
  - name: Verify URL
    uri:
      url: https://{{ swift_endpoint.stdout }}/healthcheck
      return_content: yes
    register: verify
  - name: Report output
    debug:
      msg: "{{ ansible_hostname }} can access the undercloud's Public API"
    when: verify.content == "OK"
```

この Playbook には複数のタスクが含まれており、各ノードで以下の操作を実行します。

- アンダークラウドの認証局ファイルをオーバークラウドノードにコピーします。アンダークラウドによって生成された場合には、デフォルトの場所は **/etc/pki/ca-trust/source/anchors/cm-local-ca.pem** です。
- オーバークラウドノードで、認証局トラストデータベースを更新するコマンドを実行します。
- オーバークラウドノードから、アンダークラウドの Object Storage パブリック API をチェックして、成功したかどうかを報告します。

3. 以下のコマンドで Playbook を実行します。

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml
```

ここでは、一時インベントリーを使用して、Ansible にオーバークラウドノードを指定します。

カスタムの認証局ファイルを使用している場合は、**ca_certificate** 変数で場所を変更することができます。以下に例を示します。

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml -e
ca_certificate=/home/stack/ssl/ca.crt.pem
```

4. その結果、Ansible の出力には、ノードのデバッグメッセージが表示されます。以下に例を示します。

```
ok: [192.168.24.100] => {
    "msg": "overcloud-controller-0 can access the undercloud's Public API"
}
```

関連情報

- オーバークラウドでの Ansible 自動化の実行に関する詳細は、**director のインストールと使用方法**の 動的インベントリースクリプトの実行 を参照してください。

5.14. FAST FORWARD UPGRADE の登録の設定

Fast Forward Upgrade プロセスでは、リポジトリーの切り替えに新しい方法を採用しています。このため、デプロイメントのコマンドから以前の **rhel-registration** 環境ファイルを削除する必要があります。以下に例を示します。

- environment-rhel-registration.yaml
- rhel-registration-resource-registry.yaml

Fast Forward Upgrade のプロセスでは、アップグレードの各段階でスクリプトを使用してリポジトリーを変更します。このスクリプトは、**OS::TripleO::Services::TripleoPackages** コンポーザブルサービス (**puppet/services/tripleo-packages.yaml**) の一部として含まれ、**FastForwardCustomRepoScriptContent** パラメーターを使用します。スクリプトの内容は以下のとおりです。

```
#!/bin/bash
set -e
case $1 in
  ocata)
    subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
    ;;
  pike)
    subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
    ;;
  queens)
    subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
    subscription-manager release --set=7.9
    subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-osd-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-mon-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-tools-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-13-deployment-tools-rpms
    ;;
  *)
    echo "unknown release $1" >&2
    exit 1
esac
```

director は、スクリプトに対して、OpenStack Platform バージョンのアップストリームのコード名を渡します。

コード名	バージョン
ocata	OpenStack Platform 11
pike	OpenStack Platform 12
queens	OpenStack Platform 13

queens に変更を加えると、Ceph Storage 2 のリポジトリーも無効となり、Ceph Storage 3 MON と Tools のリポジトリーが有効になります。この変更では、Ceph Storage 3 OSD リポジトリーはコンテナー化されたため、有効になりません。

状況によっては、カスタムのスクリプトを使用する必要がある場合があります。以下に例を示します。

- カスタムのリポジトリー名で Red Hat Satellite を使用する場合
- カスタムの名前で接続されていないリポジトリーを使用する場合
- 各段階に追加のコマンドを実行する場合

このような状況では、**FastForwardCustomRepoScriptContent** パラメーターを設定してカスタムスクリプトを追加します。

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    [INSERT UPGRADE SCRIPT HERE]
```

たとえば、以下のスクリプトは Satellite 6 アクティベーションキーのセットを使用して、リポジトリーを変更します。

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    set -e
    URL="satellite.example.com"
    case $1 in
      ocata)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp11 --
org=Default_Organization
        ;;
      pike)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp12 --
org=Default_Organization
        ;;
      queens)
        subscription-manager register --baseurl=https://$URL --force --activationkey=rhosp13 --
org=Default_Organization
        ;;
      *)
        echo "unknown release $1" >&2
        exit 1
    esac
```

本ガイドの後半には、カスタムスクリプトを含む **custom_repositories_script.yaml** 環境ファイルについて記載しています。

5.15. 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'
```

5.16. 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] ...
```

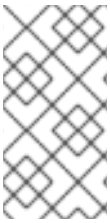
重要

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。

詳細は、ネットワーク分離 を参照してください。

5.17. DPDK および SR-IOV 設定の確認

本項は、Data Plane Development Kit (DPDK) 統合および Single Root Input/Output Virtualization (SR-IOV) 等の NFV 技術を使用するオーバークラウドに関するものです。お使いのオーバークラウドがこれらの機能を使用していない場合には、本項を無視してください。



注記

Red Hat OpenStack Platform 10 では、第一ブートスクリプトファイルを OpenStack Platform 13 用のテンプレートである **host-config-and-reboot.yaml** に置き換える必要はありません。アップグレードプロセスの開始から完了まで第一ブートスクリプトを維持することで、新たなリブートを回避します。

5.17.1. DPDK 環境のアップグレード

DPDK を使用する環境では、コンテナー化環境に正しく移行するように特定のサービスマッピングを確認します。

手順

1. コンテナー化されたサービスへの変換により、DPDK サービスの Fast Forward Upgrade は自動的に実施されます。DPDK 用のカスタム環境ファイルを使用している場合には、これらの環境ファイルを手動で調整してコンテナー化されたサービスにマッピングします。

```
OS::TripleO::Services::ComputeNeutronOvsDpdk:
  /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-agent.yaml
```

注記

あるいは、最新の NFV 環境ファイル **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-ovs-dpdk.yaml** を使用します。

2. OpenStack Network (Neutron) エージェントサービスを適切なコンテナー化されたテンプレートにマッピングします。

   - DPDK にデフォルトの **Compute** ロールを使用している場合には、**ComputeNeutronOvsAgent** サービスをコア Heat テンプレートコレクションの **docker/services** ディレクトリーの **neutron-ovs-dpdk-agent.yaml** ファイルにマッピングします。

     ```
     resource_registry:
       OS::TripleO::Services::ComputeNeutronOvsAgent:
         /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-ovs-dpdk-
     agent.yaml
     ```

   - DPDK にカスタムロールを使用している場合には、**ComputeNeutronOvsDpdkAgentCustom** 等のカスタムコンポーザブルサービスが存在しているはずです。このサービスを docker ディレクトリーの **neutron-ovs-dpdk-agent.yaml** ファイルにマッピングします。

3. 以下のサービスおよび追加パラメーターを DPDK のロール定義に追加します。

   ```
   RoleParametersDefault:
     VhostuserSocketGroup: "hugetlbfs"
     TunedProfileName: "cpu-paritioning"

   ServicesDefault:
     - OS::TripleO::Services::ComputeNeutronOvsDPDK
   ```

4. 以下のサービスを削除します。

   ```
   ServicesDefault:
     - OS::TripleO::Services::NeutronLinuxbridgeAgent
     - OS::TripleO::Services::NeutronVppAgent
     - OS::TripleO::Services::Tuned
   ```

5.17.2. SR-IOV 環境のアップグレード

SR-IOV を使用する環境では、コンテナー化環境に正しく移行するように以下のサービスマッピングを確認します。

手順

1. コンテナー化されたサービスへの変換により、SR-IOV サービスの Fast Forward Upgrade は自動的に実施されます。SR-IOV 用のカスタム環境ファイルを使用している場合には、これらのサービスをコンテナー化されたサービスに正しくマッピングします。

```
  OS::TripleO::Services::NeutronSriovAgent:
    /usr/share/openstack-tripleo-heat-templates/docker/services/neutron-sriov-agent.yaml

OS::TripleO::Services::NeutronSriovHostConfig:
    /usr/share/openstack-tripleo-heat-templates/puppet/services/neutron-sriov-host-
config.yaml
```

注記

あるいは、最新の NFV 環境ファイル **/usr/share/openstack-tripleo-heat-templates/environments/services/neutron-sriov.yaml** を使用します。

2. **roles_data.yaml** ファイルに必要な SR-IOV サービスを含めます。
SR-IOV に **デフォルト** の **Compute** ロールを使用している場合には、適切なサービスを OpenStack Platform 13 のこのロールに含めます。

   - **roles_data.yaml** ファイルを **/usr/share/openstack-tripleo-heat-templates** からお使いのカスタムテンプレートディレクトリー (例: **/home/stack/templates**) にコピーします。
   - 以下のサービスをデフォルトの Compute ロールに追加します。
     - OS::TripleO::Services::NeutronSriovAgent
     - OS::TripleO::Services::NeutronSriovHostConfig
   - 以下のサービスをデフォルトの Compute ロールから削除します。
     - OS::TripleO::Services::NeutronLinuxbridgeAgent
     - OS::TripleO::Services::Tuned
       SR-IOV に **カスタム** の **Compute** ロールを使用している場合には、**NeutronSriovAgent** サービスが存在しているはずです。Red Hat OpenStack Platform 13 で導入された **NeutronSriovHostConfig** サービスを追加します。

注記

この後のセクションで、**ffwd-upgrade**、**prepare**、**converge** コマンドを実行する際に、**roles_data.yaml** ファイルを追加する必要があります。

5.18. 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。

5.19. 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 オーバークラウドのアップグレード*に記載のステップに従って、オーバークラウドを 10 から 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

本項ではオーバークラウドをアップグレードします。これには、以下のワークフローが含まれます。

- Fast Forward Upgrade の prepare コマンドの実行
- fast forward upgrade コマンドの実行
- コントローラーノードのアップグレード
- コンピュートノードのアップグレード
- Ceph Storage ノードのアップグレード
- Fast Forward Upgrade の最終処理

このワークフローを一旦開始すると、全ステップを完了するまでオーバークラウドの OpenStack サービスは完全には制御できなくなることを認識しておいてください。これは、全ノードが OpenStack Platform 13 に正常にアップグレードされるまで、ワークロードは管理できないことを意味します。ワークロード自体は影響を受けず、稼働を続けます。オーバークラウドのワークロードへの変更または追加は、Fast Forward Upgrade が完了するまで待つ必要があります。

6.1. FAST FORWARD UPGRADE の コマンド

Fast Forward Upgrade プロセスには、プロセスの特定の段階で実行するさまざまなコマンドが含まれます。以下の一覧は、各コマンドに関する基本的な情報の一部を示しています。

重要

この一覧には、各コマンドに関する情報のみが含まれます。これらのコマンドは特定の順序で実行し、オーバークラウドに固有のオプションを指定する必要があります。適切なステップでこれらのコマンドを実行する手順を受け取るまで待ちます。

openstack overcloud ffwd-upgrade prepare

このコマンドにより、オーバークラウドのアップグレードの初期準備のステップが実行されます。これには、アンダークラウド上の現在のオーバークラウドプランを新しい OpenStack Platform 13 オーバークラウドプランおよび更新された環境ファイルに置き換えることが含まれます。このコマンドは、**openstack overcloud deploy** コマンドと同じように機能し、同じオプションを多用します。

openstack overcloud ffwd-upgrade run

このコマンドは、Fast Forward Upgrade プロセスを実行します。director は、新しい OpenStack Platform 13 オーバークラウドプランに基づいて Ansible Playbook のセットを作成し、オーバークラウド全体で Fast Forward タスクを実行します。これには、OpenStack Platform の 10 から 13 までの各バージョンでアップグレードプロセスを実行することが含まれます。

openstack overcloud upgrade run

このコマンドは、ロールの単一ノードまたは複数のノードに対して、ノード固有のアップグレード設定を実行します。director は、オーバークラウドのプランに基づいて Ansible Playbook のセットを作成し、選択したノードに対してタスクを実行します。これにより、OpenStack Platform 13 の適切な設定でノードが設定されます。このコマンドは、ロールごとに更新を実施する方法も提供します。たとえば、以下のコマンドを実行してコントローラーノードを最初にアップグレードしてから、再度コマンドを実行してコンピュートノードと Ceph Storage ノードをアップグレードします。

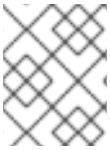
openstack overcloud ceph-upgrade run

このコマンドにより、Ceph Storage バージョンのアップグレードが実行されます。Ceph Storage ノードに対して **openstack overcloud upgrade run** を実行した後に、このコマンドを実行します。director は **ceph-ansible** を使用して Ceph Storage バージョンのアップグレードを実行します。

openstack overcloud ffwd-upgrade converge

このコマンドにより、オーバークラウドのアップグレードの最終ステップが実行されます。この最終ステップでは、オーバークラウドの Heat スタックを OpenStack Platform 13 のオーバークラウドプランおよび更新された環境ファイルと同期します。これにより、作成されるオーバークラウドが新規の OpenStack Platform 13 オーバークラウドの設定と一致します。このコマンドは、**openstack overcloud deploy** コマンドと同じように機能し、同じオプションを多用します。

これらのコマンドは、特定の順序で実行する必要があります。本章の残りの項に従って、これらのコマンドを使用して Fast Forward Upgrade を実行します。



注記

オーバークラウドにカスタム名を使用する場合には、各コマンドに **--stack** オプションを使用してカスタム名を設定します。

6.2. オーバークラウドの FAST FORWARD UPGRADE の実行

Fast Forward Upgrade には、以下のタスクを実行する 2 つのコマンドが必要です。

- オーバークラウドのプランを OpenStack Platform 13 に更新します。
- Fast Forward Upgrade に備えてノードを準備します。
- Fast Forward Upgrade の対象となる各バージョンのアップグレードステップを順番に実行します。以下の作業が含まれます。
  - 各 OpenStack Platform サービスのバージョン固有のタスク
  - リポジトリーの変更。Fast Forward Upgrade の対象となる OpenStack Platform バージョンを 1 つずつ順番に切り替える
  - データベースのアップグレードに必要な特定のパッケージを更新する
  - データベースのバージョンを 1 つずつ順番にアップグレードする
- OpenStack Platform 13 への最終アップグレードに向けてオーバークラウドを準備します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. お使いのデプロイメントに適したすべての該当するオプションおよび環境ファイルと共に、Fast Forward Upgrade の prepare コマンドを実行します。

```
$ openstack overcloud ffwd-upgrade prepare \
  --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/deprecated_cli_options.yaml \
  -e /home/stack/templates/custom_repositories_script.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
```

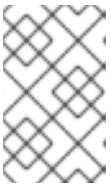
```
ansible.yaml \
  -e /home/stack/templates/ceph-customization.yaml \
  -e <ENVIRONMENT FILE>
```

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** の使用が推奨されるようになっているので、この警告は無視して問題ありません。
  - 該当する場合は、非推奨になった CLI オプションを Heat パラメーターにマッピングする環境ファイル。**deprecated_cli_options.yaml** を使用します。
  - 該当する場合は、カスタムリポジトリーのスクリプトを指定する環境ファイル。**custom_repositories_script.yaml** を使用します。
  - Ceph Storage ノードを使用する場合には、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。

重要

ffwd-upgrade コマンドの実行を確認するプロンプトが表示されます。**yes** と入力してください。



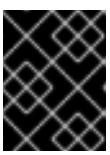
注記

openstack ffwd-upgrade prepare コマンドは複数回実行できます。コマンドが失敗した場合は、テンプレートの問題を修正してから、コマンドを再実行できます。

3. オーバークラウドプランが OpenStack Platform 13 バージョンに更新されます。Fast Forward Upgrade の準備が完了するまで待ちます。
4. アップグレードを行う前に、オーバークラウドのスナップショットまたはバックアップを作成します。
5. Fast Forward Upgrade のコマンドを実行します。

```
$ openstack overcloud ffwd-upgrade run
```

- 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。



重要

ffwd-upgrade コマンドの実行を確認するプロンプトが表示されます。**yes** と入力してください。

注記

openstack ffwd-upgrade run コマンドは、複数回実行できます。コマンドが失敗した場合は、テンプレートの問題を修正してから、コマンドを再実行できます。

6. Fast Forward Upgrade が完了するまで待ちます。

この段階では、

- ワークロードは引き続き稼働中です。
- オーバークラウドのデータベースは OpenStack Platform 12 バージョンにアップグレードされました。
- オーバークラウドのサービスがすべて無効化されます。
- Ceph Storage ノードはまだバージョン 2 のままです。

これは、オーバークラウドが、OpenStack Platform 13 に達するための標準のアップグレードステップを実行できる状態にあることを意味します。

6.3. 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。

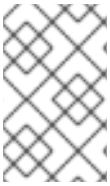
オーバークラウドで分割されたコントローラーサービスが使用されている場合は、以下のガイドに従ってノードのロールを次の順序でアップグレードします。

- 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。



注記

OpenStack Platform サービスはオーバークラウド上では非アクティブな状態で検証できないため、以下の手順のコマンドでは **--skip-tags validation** のオプションを使用しています。

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 --skip-tags validation
```

   - 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 --skip-tags validation
$ openstack overcloud upgrade run --nodes Database --skip-tags validation
$ openstack overcloud upgrade run --nodes Messaging --skip-tags validation
$ openstack overcloud upgrade run --nodes Telemetry --skip-tags validation
```

      - 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 --skip-tags validation
```

      - 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 --skip-tags validation
```

      - 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。

この段階では、

- ワークロードは引き続き稼働中です。
- オーバークラウドのデータベースが OpenStack Platform 13 バージョンにアップグレードされました。
- コントローラーノードが OpenStack Platform 13 にアップグレードされました。
- すべてのコントローラーサービスが有効化されました。

- コンピュートノードは、まだアップグレードする必要があります。
- Ceph Storage ノードはバージョン 2 のままなので、アップグレードする必要があります。

警告

コントローラーサービスは有効化されていますが、コンピュートノードと Ceph Storage サービスが無効になるまではワークロードの操作は実行しないでください。ワークロードを操作すると、仮想マシンが孤立してしまう可能性があります。環境全体がアップグレードされるまで待ってください。

6.4. テスト用コンピュートノードのアップグレード

このプロセスは、テスト用に選択したコンピュートノードをアップグレードします。このプロセスでは、**openstack overcloud upgrade run** コマンドに **--nodes** オプションを指定して、操作をテスト用ノードのみに制限して実行する必要があります。この手順では、コマンドで **--nodes compute-0** を例として使用しています。

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-0 --skip-tags validation
```

注記

OpenStack Platform サービスはオーバークラウド上では非アクティブな状態で検証できないため、上記のコマンドには **--skip-tags validation** を使用しています。

- 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. テスト用ノードのアップグレードが完了するまで待ちます。

6.5. 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 --skip-tags validation
```

注記

OpenStack Platform サービスはオーバークラウド上では非アクティブな状態で検証できないため、上記のコマンドには **--skip-tags validation** を使用しています。

- 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。

この段階では、

- ワークロードは引き続き稼働中です。
- コントローラーノードとコンピュートノードが OpenStack Platform 13 にアップグレードされました。
- Ceph Storage ノードはバージョン 2 のままなので、アップグレードする必要があります。

6.6. 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 --skip-tags validation
```

注記

OpenStack Platform サービスはオーバークラウド上では非アクティブな状態で検証できないため、上記のコマンドには **--skip-tags validation** を使用しています。

- 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 <ENVIRONMENT FILE> \
    -e /home/stack/templates/overcloud_images.yaml \
    -e /home/stack/templates/deprecated_cli_options.yaml \
    -e /home/stack/templates/custom_repositories_script.yaml
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
    -e /home/stack/templates/ceph-customization.yaml \
    --ceph-ansible-playbook '/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml,/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml'
```

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** の使用が推奨されるようになっているので、この警告は無視して問題ありません。
  - 該当する場合は、非推奨になった CLI オプションを Heat パラメーターにマッピングする環境ファイル。**deprecated_cli_options.yaml** を使用します。
  - 該当する場合は、カスタムリポジトリーのスクリプトを指定する環境ファイル。**custom_repositories_script.yaml** を使用します。
  - 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。
- 以下の Ansible Playbook
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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。

6.7. 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** の使用が推奨されるようになっているので、この警告は無視して問題ありません。

- 該当する場合は、非推奨になった CLI オプションを Heat パラメーターにマッピングする環境ファイル。**deprecated_cli_options.yaml** を使用します。
- 該当する場合は、カスタムリポジトリーのスクリプトを指定する環境ファイル。**custom_repositories_script.yaml** を使用します。
- 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。
- 以下の Ansible Playbook
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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。

6.8. 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** の使用が推奨されるようになっているので、この警告は無視して問題ありません。
  - 該当する場合は、非推奨になった CLI オプションを Heat パラメーターにマッピングする環境ファイル。**deprecated_cli_options.yaml** を使用します。
  - 該当する場合は、カスタムリポジトリーのスクリプトを指定する環境ファイル。**custom_repositories_script.yaml** を使用します。
  - 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。
- 以下の Ansible Playbook
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

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。

この段階では、

- 全ノードが OpenStack Platform 13 にアップグレードされ、ワークロードは引き続き稼働しています。

環境はアップグレードされましたが、最後のステップを 1 つ実行して、アップグレードの最終処理を行う必要があります。

6.9. FAST FORWARD UPGRADE の最終処理

Fast Forward Upgrade 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。

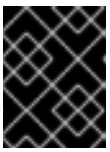
```
$ source ~/stackrc
```

2. Fast Forward Upgrade の最終処理のコマンドを実行します。

```
$ openstack overcloud ffwd-upgrade converge \
    --templates \
    -e /home/stack/templates/overcloud_images.yaml \
    -e /home/stack/templates/deprecated_cli_options.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /home/stack/templates/ceph-customization.yaml \
    -e <OTHER ENVIRONMENT FILES>
```

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** の使用が推奨されるようになっているので、この警告は無視して問題ありません。
  - 該当する場合は、非推奨になった CLI オプションを Heat パラメーターにマッピングする環境ファイル。**deprecated_cli_options.yaml** を使用します。
  - Ceph Storage ノードを使用する場合には、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。



重要

ffwd-upgrade コマンドの実行を確認するプロンプトが表示されます。**yes** と入力してください。

3. Fast Forward Upgrade の最終処理が完了するまで待ちます。

6.10. 次のステップ

オーバークラウドのアップグレードが完了しました。これで、*8章 アップグレード後のステップの実行*に記載のステップに従って、アップグレード後のオーバークラウドの設定を行うことができます。今後のデプロイメント操作では、OpenStack Platform 13 環境に関連する全環境ファイルを必ず指定してください。これには、アップグレード中に新規作成または変換した環境ファイルが含まれます。

第7章 アップグレード後のオーバークラウドのリブート

Red Hat OpenStack 環境のアップグレード後に、オーバークラウドをリブートします。リブートにより、関連付けられたカーネル、システムレベル、およびコンテナーコンポーネントの更新と共にノードが更新されます。これらの更新により、パフォーマンスとセキュリティー上のメリットが得られます。

ダウンタイムを計画して、以下のリブート手順を実行します。

7.1. 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。

7.2. 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
```

7.3. 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
```

7.4. コンピュート HCI ノードのリブート

以下の手順では、コンピュートハイパーコンバージドインフラストラクチャー (HCI) ノードをリブートします。

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. アンダークラウドに **stack** ユーザーとしてログインします。

3. 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
```

リブートするコンピュートノードの UUID を特定します。

4. 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
```

5. 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
```

6. 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する。

   ```
   (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する。

   ```
   (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う。

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

注記

nova コマンドで非推奨の警告が表示される可能性がありますが、無視して問題ありません。

7. 移行が完了するまで待ちます。

8. 移行が正常に完了したことを確認します。

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

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

10. 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。

11. コンピュート HCI ノードをリブートします。

```
$ sudo reboot
```

12. ノードがブートするまで待ちます。

13. コンピュートノードを再度有効化します。

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

14. 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
```

15. ノードからログアウトして、次のノードをリブートし、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。

16. 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
```

17. 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
```

第8章 アップグレード後のステップの実行

以下のステップでは、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。これには、Fast Forward Upgrade プロセス終了後のイメージの変更、追加の設定ステップ、考慮事項などが含まれます。

8.1. 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。

8.2. 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 環境をチェックして、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。

8.3. オーバークラウドイメージのアップグレード

現在のオーバークラウドイメージを新しいバージョンに置き換える必要があります。新しいイメージにより、director は最新バージョンの OpenStack Platform ソフトウェアを使用してノードのイントロスペクションとプロビジョニングを行うことができるようになります。

前提条件

- アンダークラウドが最新バージョンにアップグレードされていること

手順

1. アンダークラウド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。

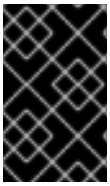
```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

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。

8.4. デプロイメントのテスト

オーバークラウドはアップグレードされましたが、今後のデプロイメント操作が正常に実行されるようにするには、テストデプロイメントを実行することを推奨します。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. お使いのオーバークラウドに関連する全環境ファイルを指定して、デプロイのコマンドを実行します。

```
$ openstack overcloud deploy \
    --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。

- **-e** を使用したカスタム設定環境ファイル。
- 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。

3. デプロイメントが完了するまで待ちます。

8.5. 結果

これで Fast Forward Upgrade のプロセスが完了しました。

付録A アンダークラウドの復元

以下の復元の手順は、お使いのアンダークラウドノードでエラーが発生して、回復不能な状態であることを前提としています。この手順では、新規インストール環境でデータベースおよびクリティカルなファイルシステムの復元を行う必要があります。以下が前提条件です。

- Red Hat Enterprise Linux 7 の最新版を再インストール済みであること
- ハードウェアレイアウトが同じであること
- マシンのホスト名とアンダークラウドの設定が同じであること
- バックアップアーカイブが **root** ディレクトリーにコピー済みであること

手順

1. お使いのアンダークラウドに **root** ユーザーとしてログインします。
2. コンテンツ配信ネットワークにシステムを登録します。プロンプトが表示されたら、カスタマーポータルのユーザー名とパスワードを入力します。

```
[root@director ~]# subscription-manager register
```

3. Red Hat OpenStack Platform のエンタイトルメントをアタッチします。

```
[root@director ~]# subscription-manager attach --pool=Valid-Pool-Number-123456
```

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

```
[root@director ~]# subscription-manager repos --disable=*
[root@director ~]# subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --enable=rhel-7-server-openstack-10-rpms
```

5. システムで更新を実行して、ベースシステムパッケージを最新の状態にします。

```
[root@director ~]# yum update -y
[root@director ~]# reboot
```

6. アンダークラウドの時刻が同期されていることを確認します。以下に例を示します。

```
[root@director ~]# yum install -y ntp
[root@director ~]# systemctl start ntpd
[root@director ~]# systemctl enable ntpd
[root@director ~]# ntpdate pool.ntp.org
[root@director ~]# systemctl restart ntpd
```

7. アンダークラウドのバックアップアーカイブをアンダークラウドの **root** ディレクトリーにコピーします。これ以降のステップでは、ファイル名に **undercloud-backup-$TIMESTAMP.tar** を使用しています。ここで、$TIMESTAMP はアーカイブのタイムスタンプの Bash 変数です。
8. データベースサーバーとクライアントツールをインストールします。

```
[root@director ~]# yum install -y mariadb mariadb-server
```

9. データベースを起動します。

```
[root@director ~]# systemctl start mariadb
[root@director ~]# systemctl enable mariadb
```

10. データベースのバックアップのサイズに対応するように、許可されるパケット数を増やします。

```
[root@director ~]# mysql -uroot -e"set global max_allowed_packet = 1073741824;"
```

11. アーカイブからデータベースおよびデータベース設定を抽出します。

```
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar
etc/my.cnf.d/*server*.cnf
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar root/undercloud-all-
databases.sql
```

12. データベースのバックアップをリストアします。

```
[root@director ~]# mysql -u root < /root/undercloud-all-databases.sql
```

13. root 設定ファイルの一時バージョンを抽出します。

```
[root@director ~]# tar -xvf undercloud-backup-$TIMESTAMP.tar root/.my.cnf
```

14. データベースの古い root パスワードを取得します。

```
[root@director ~]# OLDPASSWORD=$(sudo cat root/.my.cnf | grep -m1 password | cut -d'='
-f2 | tr -d """)
```

15. データベースの root パスワードをリセットします。

```
[root@director ~]# mysqladmin -u root password "$OLDPASSWORD"
```

16. root 設定ファイルを一時ディレクトリーから **root** ディレクトリーに移動します。

```
[root@director ~]# mv ~/root/.my.cnf ~/.
[root@director ~]# rmdir ~/root
```

17. 古いユーザー権限の一覧を取得します。

```
[root@director ~]# mysql -e 'select host, user, password from mysql.user;'
```

18. リストされた各ホストの古いユーザー権限を削除します。以下に例を示します。

```
[root@director ~]# HOST="192.0.2.1"
[root@director ~]# USERS=$(mysql -Nse "select user from mysql.user WHERE user !=
\"root\" and host = \"$HOST\";" | uniq | xargs)
[root@director ~]# for USER in $USERS ; do mysql -e "drop user \"$USER\"@\"$HOST\"" ||
```

```
true ;done
[root@director ~]# for USER in $USERS ; do mysql -e "drop user $USER" || true ;done
[root@director ~]# mysql -e 'flush privileges'
```

ホスト IP および任意のホスト (“%") からアクセスするすべてのユーザーについて、この手順を実施します。

注記

HOST パラメーターの IP アドレスは、コントロールプレーン内のアンダークラウドの IP アドレスです。

19. データベースを再起動します。

```
[root@director ~]# systemctl restart mariadb
```

20. **stack** ユーザーを作成します。

```
[root@director ~]# useradd stack
```

21. ユーザーのパスワードを設定します。

```
[root@director ~]# passwd stack
```

22. **sudo** を使用する場合にパスワードを要求されないようにします。

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

23. **stack** ユーザーのホームディレクトリーをリストアします。

```
# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar home/stack
```

24. **policycoreutils-python** パッケージをインストールします。

```
[root@director ~]# yum -y install policycoreutils-python
```

25. **openstack-glance** パッケージをインストールして、そのデータおよびファイルパーミッションをリストアします。

```
[root@director ~]# yum install -y openstack-glance
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-
$TIMESTAMP.tar var/lib/glance/images
[root@director ~]# chown -R glance: /var/lib/glance/images
[root@director ~]# restorecon -R /var/lib/glance/images
```

26. **openstack-swift** パッケージをインストールして、そのデータおよびファイルパーミッションをリストアします。

```
[root@director ~]# yum install -y openstack-swift
[root@director ~]# tar --xattrs --xattrs-include='*.*' -xvC / -f undercloud-backup-
$TIMESTAMP.tar srv/node
```

```
[root@director ~]# chown -R swift: /srv/node
[root@director ~]# restorecon -R /srv/node
```

27. **openstack-keystone** パッケージをインストールして、その設定データをリストアします。

```
[root@director ~]# yum -y install openstack-keystone
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/keystone
[root@director ~]# restorecon -R /etc/keystone
```

28. **openstack-heat** をインストールして、設定をリストアします。

```
[root@director ~]# yum install -y openstack-heat*
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/heat
[root@director ~]# restorecon -R /etc/heat
```

29. Puppet をインストールして、その設定データをリストアします。

```
[root@director ~]# yum install -y puppet hiera
[root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/puppet/hieradata/
```

30. アンダークラウドで SSL を使用する場合は、CA 証明書を更新します。アンダークラウドの設定に応じて、ユーザー提供の証明書の手順または自動生成された証明書の手順のいずれかを使用します。

- アンダークラウドがユーザー提供の証明書で設定されている場合は、次の手順を実行します。

  a. 証明書を抽出します。

  ```
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/instack-certs/undercloud.pem
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-trust/source/anchors/*
  ```

  b. SELinux コンテキストを復元し、ファイルシステムのラベル付けを管理します。

  ```
  [root@director ~]# restorecon -R /etc/pki
  [root@director ~]# semanage fcontext -a -t etc_t "/etc/pki/instack-certs(/.*)?"
  [root@director ~]# restorecon -R /etc/pki/instack-certs
  ```

  c. 証明書を更新します。

  ```
  [root@director ~]# update-ca-trust extract
  ```

- **certmonger** を使用してアンダークラウドの証明書を自動生成する場合は、以下の手順を実行します。

  a. 証明書、CA 証明書、および certmonger ファイルを抽出します。

  ```
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar var/lib/certmonger/*
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/tls/*
  [root@director ~]# tar -xvC / -f undercloud-backup-$TIMESTAMP.tar etc/pki/ca-trust/source/anchors/*
  ```

b. SELinux コンテキストを復元します。

```
[root@director ~]# restorecon -R /etc/pki
[root@director ~]# restorecon -R /var/lib/certmonger
```

c. **/var/lib/certmonger/lock** ファイルを削除します。

```
[root@director ~]# rm -f /var/lib/certmonger/lock
```

31. **stack** ユーザーに切り替えます。

```
[root@director ~]# su - stack
[stack@director ~]$
```

32. **python-tripleoclient** パッケージをインストールします。

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

33. アンダークラウドのインストールコマンドを実行します。このコマンドは、**stack** ユーザーのホームディレクトリーから実行するようにしてください。

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

インストールが完了すると、アンダークラウドは、オーバークラウドへの接続を自動的にリストアします。ノードは、保留中のタスクに対して、OpenStack Orchestration (heat) のポーリングを続けます。

付録B オーバークラウドの復元

B.1. オーバークラウドのコントロールプレーンサービスの復元

以下の手順では、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リストアします。このような場合には、ターミナルのウィンドウを 3 つ開いて、特定の操作を 3 つのコントローラーノードすべてで同時に実行できるようにすることを推奨します。また、高可用性の操作を実行するコントローラーノードを 1 台選択することもお勧めします。この手順では、このコントローラーノードを **ブートストラップコントローラーノード** と呼びます。

重要

この手順では、コントロールプレーンサービスのみを復元します。コンピュートノードのワークロードや Ceph Storage ノード上のデータの復元は含まれません。

注記

Red Hat は、Open vSwitch (OVS) およびデフォルトの Open Virtual Network (OVN) などのネイティブ SDN を使用する Red Hat OpenStack Platform のバックアップをサポートします。サードパーティーの SDN についての詳細は、サードパーティーの SDN ドキュメントを参照してください。

手順

1. Pacemaker を停止し、コンテナー化されたすべてのサービスを削除します。
   a. ブートストラップコントローラーノードにログインし、Pacemaker クラスターを停止します。

   ```
   # sudo pcs cluster stop --all
   ```

   b. クラスターが完全にシャットダウンするまで待ちます。

   ```
   # sudo pcs status
   ```

   c. すべてのコントローラーノードで、OpenStack サービスのコンテナーをすべて削除します。

   ```
   # docker stop $(docker ps -a -q)
   # docker rm $(docker ps -a -q)
   ```

2. メジャーバージョンのアップグレードの失敗からリストアする場合には、全ノードで実行された **yum** トランザクションをすべて元に戻さなければならない場合があります。これには、各ノードで以下の操作が必要です。
   a. 以前のバージョンのリポジトリーを有効化します。以下に例を示します。

   ```
   # sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
   # sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
   # sudo subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
   ```

   b. 以下の Ceph リポジトリーを有効にします。

```
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-tools-rpms
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-2-mon-rpms
```

c. **yum** 履歴をチェックします。

```
# sudo yum history list all
```

アップグレードプロセス中に発生したトランザクションを特定します。これらの操作の大半は、コントローラーノードの1台 (アップグレード中にブートストラップノードとして選択されていたコントローラーノード) で発生しているはずです。特定のトランザクションを確認する必要がある場合は、**history info** サブコマンドで表示してください。

```
# sudo yum history info 25
```

注記

yum history list all で各トランザクションから実行したコマンドを表示するように強制するには、**yum.conf** ファイルで **history_list_view=commands** を設定します。

d. アップグレード以降に発生した **yum** トランザクションをすべて元に戻します。以下に例を示します。

```
# sudo yum history undo 25
# sudo yum history undo 24
# sudo yum history undo 23
...
```

最後のトランザクションから開始して、降順に操作を継続するようにしてください。また、**rollback** オプションを使用すると、複数のトランザクションを1回の実行で元に戻すこともできます。たとえば、以下のコマンドは最後のトランザクションから 23 までのトランザクションをロールバックします。

```
# sudo yum history rollback 23
```

重要

各トランザクションの取り消しを確認できるようにするには、**rollback** ではなく **undo** を使用することを推奨します。

e. 関連する **yum** トランザクションが取り消されたら、全ノードで元の OpenStack Platform リポジトリーのみを有効化します。以下に例を示します。

```
# sudo subscription-manager repos --disable=rhel-7-server-openstack-*-rpms
# sudo subscription-manager repos --enable=rhel-7-server-openstack-10-rpms
```

f. 以下の Ceph リポジトリーを無効にします。

```
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
# sudo subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
```

3. データベースを復元します。

   a. ブートストラップコントローラーノードにデータベースのバックアップをコピーします。

   b. 全コントローラーノード上でデータベースポートへの外部接続を停止します。

   ```
   # MYSQLIP=$(hiera -c /etc/puppet/hiera.yaml mysql_bind_host)
   # sudo /sbin/iptables -I INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
   ```

   これにより、ノードへのデータベーストラフィックがすべて分離されます。

   c. 一時的にデータベースのレプリケーションを無効にします。すべてのコントローラーノードで **/etc/my.cnf.d/galera.cnf** ファイルを編集します。

   ```
   # vi /etc/my.cnf.d/galera.cnf
   ```

   以下の変更を加えます。

   - **wsrep_cluster_address** パラメーターをコメントアウトする
   - **wsrep_provider** を **none** に設定する

   d. **/etc/my.cnf.d/galera.cnf** ファイルを保存します。

   e. すべてのコントローラーノードで MariaDB データベースが無効になっていることを確認します。OpenStack Platform 13 へのアップグレード中に、以前の手順で無効にした MariaDB サービスがコンテナー化されたサービスに移行します。ホスト上でもサービスがプロセスとして実行されていないことを確認してください。

   ```
   # mysqladmin -u root shutdown
   ```

   注記

   HAProxy から、データベースが無効になったという警告が表示される可能性があります。

   f. 既存の MariaDB データディレクトリーを移動し、全コントローラーノード上で新規データディレクトリーを準備します。

   ```
   # mv /var/lib/mysql/ /var/lib/mysql.old
   # mkdir /var/lib/mysql
   # chown mysql:mysql /var/lib/mysql
   # chmod 0755 /var/lib/mysql
   # mysql_install_db --datadir=/var/lib/mysql --user=mysql
   # chown -R mysql:mysql /var/lib/mysql/
   # restorecon -R /var/lib/mysql
   ```

   g. すべてのコントローラーノードでデータベースを手動で起動します。

   ```
   # mysqld_safe --skip-grant-tables --skip-networking --wsrep-on=OFF &
   ```

   h. 古いパスワードを取得し、すべてのコントローラーノードでデータベースのパスワードを再設定します。

```
# OLDPASSWORD=$(sudo cat .my.cnf | grep -m1 password | cut -d'=' -f2 | tr -d "'")
# mysql -uroot -e"use mysql;update user set
password=PASSWORD($OLDPASSWORD)"
```

i. すべてのコントローラーノードでデータベースを停止します。

```
# /usr/bin/mysqladmin -u root shutdown
```

j. ブートストラップコントローラーノードで、**--skip-grant-tables** オプションを指定せずに手動でデータベースを起動します。

```
# mysqld_safe --skip-networking --wsrep-on=OFF &
```

k. ブートストラップコントローラーノードで、OpenStack データベースを復元します。この後の手順で、このデータベースが他のコントローラーノードに複製されます。

```
# mysql -u root < openstack_database.sql
```

l. ブートストラップコントローラーノードで、ユーザーとパーミッションを復元します。

```
# mysql -u root < grants.sql
```

m. 以下のコマンドを使用して、ブートストラップコントローラーノードをシャットダウンします。

```
# mysqladmin shutdown
```

n. データベースのレプリケーションを有効にします。すべてのコントローラーノードで **/etc/my.cnf.d/galera.cnf** ファイルを編集します。

```
# vi /etc/my.cnf.d/galera.cnf
```

以下の変更を加えます。

- **wsrep_cluster_address** パラメーターをコメント解除する
- **wsrep_provider** を **/usr/lib64/galera/libgalera_smm.so** に設定する

o. **/etc/my.cnf.d/galera.cnf** ファイルを保存します。

p. ブートストラップノードでデータベースを実行します。

```
# /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm:// &
```

--wsrep-cluster-address オプションにノードを含めないと、Galera により新しいクラスターが作成され、ブートストラップノードがマスターノードになります。

q. ノードのステータスを確認します。

```
# clustercheck
```

このコマンドにより、**Galera cluster node is synced.** という出力が表示されるはずです。エラーが発生した場合には、**/var/log/mysql_cluster.log** ファイルを確認してください。

r. 残りのコントローラーノードでデータベースを起動します。

```
$ /usr/bin/mysqld_safe --pid-file=/var/run/mysql/mysqld.pid --
socket=/var/lib/mysql/mysql.sock --datadir=/var/lib/mysql --log-
error=/var/log/mysql_cluster.log  --user=mysql --open-files-limit=16384 --wsrep-cluster-
address=gcomm://overcloud-controller-0,overcloud-controller-1,overcloud-controller-2 &
```

--wsrep-cluster-address オプションにノードを含めると、新しいクラスターにノードが追加され、マスターからの内容と同期されます。

s. 定期的に各ノードのステータスを確認します。

```
# clustercheck
```

全ノードの同期操作が完了したら、このコマンドにより、それぞれのノードについて **Galera cluster node is synced.** という出力が表示されるはずです。

t. すべてのノードのデータベースを停止します。

```
$ mysqladmin shutdown
```

u. 各ノードからファイアウォールルールを削除して、サービスがデータベースへのアクセスを回復するようにします。

```
# sudo /sbin/iptables -D INPUT -d $MYSQLIP -p tcp --dport 3306 -j DROP
```

4. Pacemaker の設定を復元します。

a. Pacemaker のアーカイブをブートストラップノードにコピーします。

b. ブートストラップノードにログインします。

c. 設定の復元コマンドを実行します。

```
# pcs config restore pacemaker_controller_backup.tar.bz2
```

5. ファイルシステムを復元します。

a. 各コントローラーノードのバックアップ **tar** ファイルを一時ディレクトリーにコピーして、圧縮された全データを展開します。

```
# mkdir /var/tmp/filesystem_backup/
# cd /var/tmp/filesystem_backup/
# mv <backup_file>.tar.gz .
# tar --xattrs --xattrs-include='*.*' -xvzf <backup_file>.tar.gz
```

注記

/ ディレクトリーには直接展開しないでください。直接展開すると、現在のファイルシステムが上書きされてしまいます。ファイルを一時ディレクトリーで抽出することを推奨します。

b. **os-*-config** ファイルをリストアし、**os-collect-config** を再起動します。

```
# cp -rf /var/tmp/filesystem_backup/var/lib/os-collect-config/* /var/lib/os-collect-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-apply-config/* /usr/libexec/os-apply-config/.
# cp -rf /var/tmp/filesystem_backup/usr/libexec/os-refresh-config/* /usr/libexec/os-refresh-config/.
# systemctl restart os-collect-config
```

c. Puppet hieradata ファイルを復元します。

```
# cp -r /var/tmp/filesystem_backup/etc/puppet/hieradata /etc/puppet/hieradata
# cp -r /var/tmp/filesystem_backup/etc/puppet/hiera.yaml /etc/puppet/hiera.yaml
```

d. 設定ファイルが必要な場合には、このディレクトリーを保持します。

6. redis リソースを復元します。

a. redis ダンプを各コントローラーノードにコピーします。

b. redis ダンプを各コントローラー上の本来の場所に移動します。

```
# mv dump.rdb /var/lib/redis/dump.rdb
```

c. redis ディレクトリーへのアクセス権限を復元します。

```
# chown -R redis: /var/lib/redis
```

7. 以下のすべてのディレクトリーの内容を削除します。

```
# rm -rf /var/lib/config-data/puppet-generated/*
# rm /root/.ffu_workaround
```

8. OpenStack Object Storage (swift) サービスのアクセス権限を復元します。

```
# chown -R swift: /srv/node
# chown -R swift: /var/lib/swift
# chown -R swift: /var/cache/swift
```

9. アンダークラウドにログインし、お使いの OpenStack Platform 10 デプロイメントから元の **openstack overcloud deploy** コマンドを実行します。元のデプロイメントに関連するすべての環境ファイルを必ず含めてください。

10. デプロイメントが完了するまで待ちます。

11. オーバークラウドのコントロールプレーンのデータを復元した後には、関連する各サービスが有効化されて正しく実行されていることを確認します。

a. コントローラーノード上の高可用性サービスの場合:

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

b. コントローラーおよびコンピュートノード上のシステムサービスの場合:

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

以下の項には、有効にすべきサービスについての参考情報を記載します。

B.2. 高可用性サービスの復元

復元の後に OpenStack Platform 10 のコントローラーノードでアクティブにする必要のある高可用性サービスの一覧は以下のとおりです。これらのサービスのいずれかが無効化されている場合には、以下のコマンドで有効化します。

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

コントローラーサービス
galera
haproxy
openstack-cinder-volume
rabbitmq
redis

B.3. コントローラーサービスの復元

復元の後に OpensTack Platform 10 のコントローラーノードでアクティブにする必要のある Systemd のコアサービスの一覧は以下のとおりです。これらのサービスのいずれかが無効化されている場合には、以下のコマンドで有効化します。

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

コントローラーサービス
httpd
memcached
neutron-dhcp-agent
neutron-l3-agent
neutron-metadata-agent

コントローラーサービス
neutron-openvswitch-agent
neutron-ovs-cleanup
neutron-server
ntpd
openstack-aodh-evaluator
openstack-aodh-listener
openstack-aodh-notifier
openstack-ceilometer-central
openstack-ceilometer-collector
openstack-ceilometer-notification
openstack-cinder-api
openstack-cinder-scheduler
openstack-glance-api
openstack-glance-registry
openstack-gnocchi-metricd
openstack-gnocchi-statsd
openstack-heat-api-cfn
openstack-heat-api-cloudwatch
openstack-heat-api
openstack-heat-engine
openstack-nova-api
openstack-nova-conductor
openstack-nova-consoleauth

コントローラーサービス
openstack-nova-novncproxy
openstack-nova-scheduler
openstack-swift-account-auditor
openstack-swift-account-reaper
openstack-swift-account-replicator
openstack-swift-account
openstack-swift-container-auditor
openstack-swift-container-replicator
openstack-swift-container-updater
openstack-swift-container
openstack-swift-object-auditor
openstack-swift-object-expirer
openstack-swift-object-replicator
openstack-swift-object-updater
openstack-swift-object
openstack-swift-proxy
openvswitch
os-collect-config
ovs-delete-transient-ports
ovs-vswitchd
ovsdb-server
pacemaker

B.4. オーバークラウドの COMPUTE サービスの復元

復元の後に OpensTack Platform 10 のコンピュートノードでアクティブにする必要のある Systemd のコアサービスの一覧は以下のとおりです。これらのサービスのいずれかが無効化されている場合には、以下のコマンドで有効化します。

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Compute サービス
neutron-openvswitch-agent
neutron-ovs-cleanup
ntpd
openstack-ceilometer-compute
openstack-nova-compute
openvswitch
os-collect-config