



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

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

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

概要

本ガイドには、Fast Forward 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	4
1.1. 作業を開始する前に	4
1.2. FAST FORWARD UPGRADE	4
1.3. ワークフローの概要	4
第2章 OPENSTACK PLATFORM アップグレードの準備	6
2.1. アンダークラウドのバックアップ	6
2.2. オーバークラウドのコントロールプレーンサービスのバックアップ	7
2.3. NFV 対応環境の更新準備	10
2.4. 現在のアンダークラウドパッケージの OPENSTACK PLATFORM 10.Z の更新	10
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	11
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	12
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再起動	14
2.8. CEPH STORAGE (OSD) クラスターの再起動	15
2.9. コンピュートノードの再起動	15
2.10. システムパッケージの確認	17
2.11. OPENSTACK PLATFORM 10 アンダークラウドの検証	17
2.12. OPENSTACK PLATFORM 10 オーバークラウドの検証	18
2.13. NFV 対応環境の更新の最終処理	21
2.14. 次のステップ	21
第3章 アンダークラウドのアップグレード	22
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	22
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	23
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	24
3.4. 次のステップ	25
第4章 コンテナイメージのソースの設定	26
4.1. レジストリーメソッド	26
4.2. CONTAINER IMAGE PREPARE コマンドの使用方法	26
4.3. 追加のサービス用のコンテナイメージ	28
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	30
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	30
4.6. SATELLITE サーバーをレジストリーとして使用する手順	32
4.7. 次のステップ	35
第5章 オーバークラウドのアップグレードの準備	36
5.1. オーバークラウドサービスのダウンタイムの準備	36
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	36
5.3. 新規コンポーザブルサービス	37
5.4. 非推奨のコンポーザブルサービス	38
5.5. 非推奨パラメーター	39
5.6. 非推奨の CLI オプション	40
5.7. コンポーザブルネットワーク	43
5.8. CEPH STORAGE ノードのアップグレードの準備	45
5.9. ストレージバックエンドの準備	46
5.10. 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	47
5.11. FAST FORWARD UPGRADE の登録の設定	49
5.12. カスタムの PUPPET パラメーターの確認	50
5.13. ネットワークインターフェースのテンプレートを新しい構造に変換する方法	52
5.14. 次のステップ	53

第6章 オーバークラウドのアップグレード	54
6.1. オーバークラウドの FAST FORWARD UPGRADE の実行	54
6.2. 全コントローラーノードのアップグレード	56
6.3. テスト用コンピュートノードのアップグレード	57
6.4. 全コンピュートノードのアップグレード	57
6.5. 全 CEPH STORAGE ノードのアップグレード	58
6.6. FAST FORWARD UPGRADE の最終処理	60
6.7. 次のステップ	61
第7章 アップグレード後のステップの実行	62
7.1. アンダークラウドの検証	62
7.2. コンテナ化されたオーバークラウドの検証	62
7.3. オーバークラウドイメージのアップグレード	65
7.4. デプロイメントのテスト	66
7.5. 結果	66
付録A アンダークラウドの復元	67
付録B オーバークラウドの復元	71
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	71
B.2. 復元した高可用性サービス	75
B.3. コントローラーサービスの復元	76
B.4. オーバークラウドの COMPUTE サービスの復元	78
付録C NFV の更新用の YAML ファイルのサンプル	80
C.1. RED HAT OPENSTACK PLATFORM 10 OVS 2.9 の更新ファイル	80
C.1.1. post-install.yaml	80

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。

- Fast Forward Upgrade のワークフローは、現在開発中です。特に **ffwd-upgrade** CLI コマンドの起動は、最初は開発/テスト環境に限定すべきです。Fast Forward Upgrade を実稼働環境で使用するにしたい場合には、実稼働レベルの Fast Forward Upgrade の実行を試みる前に、Customer Experience and Engagement チーム (<https://access.redhat.com/support>) に連絡してサポートを受けてください。
- バージョン 7 または 8 を使用する 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でアップグレードパスとコンテナ化されたサービスのデプロイを妨げる問題があることに注意してください。この問題に関する詳しい情報と解決方法については、以下の記事を参照してください。
 - [「XFS filetype=0 prevents upgrading to a version of OpenStack Director with containers](#)

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。目的は、ユーザーが **ロングライフバージョン** とされる特定の OpenStack バージョンを使用し続け、次のロングライフバージョンが提供された時点でアップグレードできるようにすることです。

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をまとめています。

ステップ	説明
環境の準備	アンダークラウドノードとオーバークラウドのコントローラーノードのデータベースおよび設定のバックアップを実行してから、最新のマイナーリリースに更新して、再起動し、環境を検証します。
アンダークラウドのアップグレード	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。

ステップ	説明
コンテナイメージの取得	さまざまな OpenStack サービス用のコンテナイメージの場所が記載された環境ファイルを作成します。
オーバークラウドの準備	オーバークラウドの設定ファイルを OpenStack Platform 13 に移行するための適切なステップを実行します。
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。
コントローラーノードのアップグレード	全コントローラーノードを同時に OpenStack Platform 13 にアップグレードします。
コンピュートノードのアップグレード	選択したコンピュートノードでアップグレードをテストします。テストが成功したら、全コンピュートノードをアップグレードします。
Ceph Storage ノードのアップグレード	全 Ceph Storage ノードをアップグレードします。これには、Red Hat Ceph Storage 3 のコンテナ化されたバージョンへのアップグレードも含まれます。
アップグレードの最終段階	コンバージェンスのコマンドを実行して、オーバークラウドスタックをリフレッシュします。

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再起動します。
- 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再起動します。
- アンダークラウドとオーバークラウドの両方で検証のチェックを実行します。

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

2.1. アンダークラウドのバックアップ

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

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



注記

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

手順

1. アンダークラウドに **root** ユーザーとしてログインします。

2. データベースをバックアップします。

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

3. データベースのバックアップと設定ファイルをアーカイブします。

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

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

関連情報

- アンダークラウドのバックアップを復元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、作業環境のスナップショットが確保されます。スナップショットがあると、操作のエラーが発生してオーバークラウドを元の状態に復元する必要がある場合に役立ちます。



重要

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

手順

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

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

- b. **root** ユーザーに変更します。

```
$ sudo -i
```

- c. バックアップを保管するための一時ディレクトリーを作成します。

```
# 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** 変数に保管されています。以下のコマンドを使用してパスワードを保管します。

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

- e. データベースをバックアップします。

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

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

- f. ユーザーおよびパーミッションに関する全情報をバックアップします。

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

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

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

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

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

- b. バックアップを作成します。

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

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

3. Redis クラスターをバックアップします。

- a. HAProxy から Redis のエンドポイントを取得します。

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

- b. Redis クラスターのマスターパスワードを取得します。

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

- c. Redis クラスターの接続をチェックします。

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

- d. Redis データベースをダンプします。

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

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

4. 各コントローラーノードのファイルシステムをバックアップします。

- a. バックアップ用のディレクトリーを作成します。

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

- b. 以下の **tar** コマンドを実行します。

```
# tar --ignore-failed-read --xattrs \
  -zcvf /var/tmp/filesystem_backup/`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 \
  /usr/libexec/os-apply-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。

5. 作成された **tar** ファイルを安全な場所にコピーします。

- オーバークラウドのバックアップを復元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. NFV 対応環境の更新準備

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する前に以下のステップを実行する必要があります。

手順

1. この [post-install.yaml](#) のサンプルファイルの内容を既存の **post-install.yaml** ファイルのいずれかに追加します。
2. 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を変更します。

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

3. **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
```

2.4. 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 環境の現在のバージョン内のマイナー更新を実行することができます。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. **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
```

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

```
$ openstack undercloud upgrade
```

5. コマンドの実行が完了するまで待ちます。
6. アンダークラウドを再起動して、オペレーティングシステムのカーネルとその他のシステムパッケージを更新します。

```
$ sudo reboot
```

7. ノードが起動するまで待ちます。
8. 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 **openstack-tripleo-heat-template** パッケージと同期するように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 Forward 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。

2.5. 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 (**/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-10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-latest-10.0.tar; do tar -xvf $i; done
```

- 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 ~
$ 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 " ")
```

- 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
```

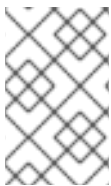
director は古いイメージを保持して、それらが更新された時のタイムスタンプを使用して名前を変更します。必要がなくなったイメージは削除してください。

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 環境の現在のバージョン内のマイナー更新を実行することができます。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

- 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
2. **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。以下に例を示します。

```
$ 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、確認のためのブレイクポイントが提供されます。**-i** オプションを使用しなかった場合には、最初のブレイクポイントで更新が一時停止されたままとなるため、**-i** オプションの指定は必須です。



注記

全ノードで並行して更新を実行すると問題が発生する可能性があります。たとえば、パッケージの更新には、サービスの再起動が必要となる場合があります。その操作によって他のノードが中断される可能性があります。そのため、このプロセスでは、一連のブレイクポイントを設けて、ノードごとに更新します。1つのノードでパッケージの更新が完了すると、更新プロセスは次のノードに移ります。

3. 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、ブレイクポイントを通過するように定期的に要求します。以下に例を示します。

```
not_started: [u'overcloud-controller-0', u'overcloud-controller-1',
u'overcloud-controller-2']
on_breakpoint: [u'overcloud-compute-0']
Breakpoint reached, continue? Regexp or Enter=proceed, 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イクポイントを通過します。

これで、そのノードの更新が開始します。また、ノード名を入力して特定のノードでブレイクポイントを通過したり、複数のノードで一度にブレイクポイントを通過するための Python ベースの正規表現を入力することも可能です。ただし、複数のコントローラーノードで同時にブレイクポイントを通過することはお勧めしません。全ノードが更新を完了するまで、このプロセスを継続します。

4. 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
```

5. 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。更新プロセスの完了時には、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再起動しません。カーネルおよびその他のシステムパッケージを更新した場合には、再起動が必要です。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** のパッケージのメジャー/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再起動します。

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再起動

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

手順

1. ノードを選択してログインします。
2. ノードを再起動します。

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

3. ノードが起動するまで待ちます。
4. ノードにログインしてサービスをチェックします。以下に例を示します。
 - a. ノードが Pacemaker サービスを使用している場合には、ノードがクラスターに再度参加したかどうかを確認します。

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

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

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

2.8. CEPH STORAGE (OSD) クラスターの再起動

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

手順

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

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

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

```
$ sudo reboot
```

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

```
$ sudo ceph -s
```

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

6. ノードからログアウトして、次のノードを再起動し、ステータスを確認します。全 Ceph Storage ノードが再起動されるまで、このプロセスを繰り返します。
7. 完了したら、Ceph MON またはコントローラーノードにログインして、クラスターのリバランスを再度有効にします。

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

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

```
$ sudo ceph status
```

2.9. コンピュートノードの再起動

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

- 再起動するコンピュートノードを選択して無効にし、新規インスタンスをプロビジョニングしないようにします。
- インスタンスを別のコンピュートノードに移行します。
- 空のコンピュートノードを再起動して有効化します。

手順

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

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

再起動するコンピュートノードのUUID を特定します。

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

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

4. コンピュートノード上の全インスタンスを一覧表示します。

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

5. 以下のコマンドの 1 つを使用して、インスタンスを移行します。

- a. 選択した特定のホストにインスタンスを移行します。

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

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

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

- c. 一度にすべてのインスタンスのライブマイグレーションを行います。

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



注記

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

6. 移行が完了するまで待ちます。
7. 正常に移行したことを確認します。

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

8. 選択したコンピュー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。
9. コンピュートノードのログインしてリブートします。

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

10. ノードが起動するまで待ちます。

11. コンピュートノードを再度有効化します。

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

12. コンピュートノードが有効化されているかどうかを確認します。

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

2.10. システムパッケージの確認

アップグレードを行う前に、全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

各ノードで以下の手順を実行して、パッケージのバージョンを確認します。

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
```

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2つの手順で、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手順を以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

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

```
$ 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
```

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

2.12. 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手順を以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

2. ベアメタルノードのステータスを確認します。

```
$ openstack baremetal node list
```

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

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

```
$ 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'
```

以下の cURL 要求でそれらの情報を使用します。

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

<PASSWORD> と <IP ADDRESS> は、**haproxy.stats** サービスからのそれぞれの情報に置き換えます。その結果表示される一覧には、各ノード上の 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。

5. オーバークラウドデータベースのレプリケーションの正常性をチェックします。

```
$ 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
```

6. 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
```

7. 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

8. 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
```

9. 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"
```

10. 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"
```

11. 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
```

12. オーバークラウドのアクセス情報を読み込みます。

```
$ source ~/overcloudrc
```

13. 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。

14. 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。

15. 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 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

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しい説明は、「[Manage Instance Snapshots](#)」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. 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
```

2. 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
```

3. 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
```

4. 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
```

5. 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が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. 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](#)」に記載の手順で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 Heat のコアテンプレートを **OpenStack Platform 11** リリースにアップグレードします。

手順

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

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
```

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

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
```

4. 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。

5. 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、**undercloud.conf** ファイルの特定のパラメーターを設定して、**192.0.2.0/24** ネットワークを引き続き使用する必要があることを意味します。そ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間に **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、ネットワークの構成が正しく設定されていることを確認してください。

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 instack-undercloud openstack-puppet-modules
openstack-tripleo-common 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。

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 Heat のコアテンプレートを **OpenStack Platform 12** リリースにアップグレードします。

手順

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

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

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

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

4. オーバークラウドが Ceph Storage とともに構成されている場合には、**ceph-ansible** パッケージをインストールします。

```
$ sudo yum install -y ceph-ansible
```

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の手順については、『[director のインストールと使用方法](#)』ガイドの「[Virtual Baseboard Management Controller \(VBMC\)](#)」の項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。

アンダークラウドを **OpenStack Platform 12** リリースにアップグレードする手順が完了しました。

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 Heat のコアテンプレートを **OpenStack Platform 13** リリースにアップグレードします。

手順

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

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

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

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

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

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

5. 以下のコマンドを実行してアンダークラウドをアップグレードします。

```
$ openstack undercloud upgrade
```

6. アンダークラウドのアップグレードプロセスが完了するまで待ちます。

7. アンダークラウドを再起動して、オペレーティングシステムのカーネルとその他のシステムパッケージを更新します。

```
$ sudo reboot
```

8. ノードが起動するまで待ちます。

アンダークラウドを **OpenStack Platform 13** リリースにアップグレードする手順が完了しました。

3.4. 次のステップ

アンダークラウドのアップグレードが完了しました。これでコンテナイメージのソースを設定することができます。

第4章 コンテナイメージのソースの設定

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

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

4.1. レジストリーメソッド

Red Hat OpenStack Platform は以下のレジストリータイプをサポートしています。

リモートレジストリー

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

ローカルレジストリー

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

Satellite サーバー

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

上記のリストから方法を選択し、レジストリー情報の設定を続けます。

4.2. CONTAINER IMAGE PREPARE コマンドの使用法

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

オーバークラウド用のコンテナイメージ環境ファイルの生成

openstack overcloud container image prepare コマンドの主要な用途の1つに、オーバークラウドが使用するイメージの一覧が記載された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のコマンドで追加します。**openstack overcloud container image prepare** コマンドは、この機能に以下のオプションを使用します。

--output-env-file

作成される環境ファイルの名前を定義します。

以下のスニペットは、このファイルの内容の例を示しています。

```
parameter_defaults:
```

```
DockerAodhApiImage: registry.access.redhat.com/rhosp13/openstack-aodh-api:latest
DockerAodhConfigImage: registry.access.redhat.com/rhosp13/openstack-aodh-api:latest
...
```

インポート方法に対応したコンテナイメージ一覧の生成

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

openstack overcloud container image prepare コマンドでは、この機能に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.access.redhat.com/rhosp13/openstack-aodh-api:latest
- imagename: registry.access.redhat.com/rhosp13/openstack-aodh-evaluator:latest
...
```

コンテナイメージの名前空間の設定

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

--namespace

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

--prefix

イメージ名の前に追加するプレフィックスを定義します。

その結果、director は以下のような形式のイメージ名を生成します。

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

コンテナイメージタグの設定

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

--tag-from-label

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

--tag

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

4.3. 追加のサービス用のコンテナイメージ

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

-e

追加のコンテナイメージを有効化するための環境ファイルを指定します。

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

サービス	環境ファイル
Ceph Storage	<code>environments/ceph-ansible/ceph-ansible.yaml</code>
Collectd	<code>environments/services-docker/collectd.yaml</code>
Congress	<code>environments/services-docker/congress.yaml</code>
Fluentd	<code>environments/services-docker/fluentd-client.yaml</code>
OpenStack Bare Metal (ironic)	<code>environments/services-docker/ironic.yaml</code>
OpenStack Data Processing (sahara)	<code>environments/services-docker/sahara.yaml</code>
OpenStack EC2-API	<code>environments/services-docker/ec2-api.yaml</code>
OpenStack Key Manager (barbican)	<code>environments/services-docker/barbican.yaml</code>
OpenStack Load Balancing-as-a-Service (octavia)	<code>environments/services-docker/octavia.yaml</code>
OpenStack Shared File System Storage (manila)	<code>environments/services-docker/manila.yaml</code>
Sensu	<code>environments/services-docker/sensu-client.yaml</code>

以下の項には、追加するサービスの例を記載します。

Ceph Storage

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

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

--set ceph_namespace

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

--set ceph_image

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

--set ceph_tag

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

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

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

OpenStack Bare Metal (ironic)

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

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

OpenStack Data Processing (sahara)

オーバークラウドで OpenStack 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 \
...
```

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イメージを **registry.access.redhat.com** でホストしています。リモートレジストリーからイメージをプルする方法では、レジストリーはすでに設定済みで、必要なのはプルするイメージの URL と名前空間だけなので、最も簡単です。ただし、オーバークラウドの作成中には、オーバークラウドノードがリモートレジス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。これが問題となる場合には、以下のいずれかの手段を取ることができます。

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

手順

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

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

- 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。
 - Ceph Storage を使用している場合には、Ceph Storage 用のコンテナ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. これで、イメージの場所が記載された **overcloud_images.yaml** 環境ファイルがアンダークラウド上に作成されます。このファイルをデプロイメントで指定します。

4.5. ローカルレジストリーとしてアンダークラウドを使用する方法

アンダークラウド上でローカルレジストリーを設定して、オーバークラウドのコンテナイメージを保管することができます。この方法は、以下の操作を伴います。

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

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

手順

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

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

アンダークラウドの IP アドレスを使用します。これは **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. イメージをローカルレジストリーにアップロードするためのテンプレートと、それらのイメージを参照する環境ファイルを作成します。

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

- 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。
 - Ceph Storage を使用している場合には、Ceph Storage 用のコンテナ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**
3. これで 2 つのファイルが作成されます。
 - リモートソースからのコンテナイメージの情報が含まれている **local_registry_images.yaml**。このファイルを使用して、Red Hat Container Registry (**registry.access.redhat.com**) からアンダークラウドにイメージをプルします。
 - アンダークラウド上の最終的なイメージの場所が記載されている **overcloud_images.yaml**。このファイルはデプロイメントで指定します。両方のファイルが存在することを確認します。
 4. **registry.access.redhat.com** からアンダークラウドにコンテナイメージをプルします。

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

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



注記

コンテナイメージは、およそ 10 GB のディスク領域を使用します。

レジストリーの設定の準備が整いました。

4.6. SATELLITE サーバーをレジストリーとして使用する手順

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

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

手順

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

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

- 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。
- Ceph Storage を使用している場合には、Ceph Storage 用のコンテナ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

このステップの **openstack overcloud container image prepare** コマンドは、**registry.access.redhat.com** 上のレジストリーをターゲットにして、イメージのリストを生成します。ここでは、後半のステップで使用する **openstack overcloud container image prepare** コマンドとは異なる値を指定します。

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

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

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

4. **satellite_images_names** ファイルを、Satellite 6 の **hammer** ツールが含まれるシステムにコピーします。あるいは、『[Hammer CLI ガイド](#)』に記載の手順に従って、**hammer** ツールをアンダークラウドに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イメージを追加します。

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

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

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

8. コンテナイメージを同期します。

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

Satellite サーバーが同期を完了するまで待ちます。



注記

設定によっては、**hammer** から Satellite サーバー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。詳細は、『**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を作成して、イメージを取り入れます。

10. **base** イメージに利用可能なタグを確認します。

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

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

11. アンダークラウドに戻り、Satellite サーバー上のイメージ用に環境ファイルを生成します。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、前のステップで使った **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 のデフォルトのレジストリーポートは 5000 です。例: **--namespace=satellite6.example.com:5000**
- **--prefix**: プレフィックスは Satellite 6 の命名規則に基づきます。これは、コンテンツビューを使用するかどうかによって異なります。
 - コンテンツビューを使用する場合、構成は **[org]-[environment]-[content view]-[product]-** となります (例: **acme-production-myosp13-osp13_containers-**)。
 - コンテンツビューを使用する場合、構成は **[org]-[product]-** となります (例: **acme-osp13_containers-**)。
- **--tag-from-label {version}-{release}**: 各イメージの最新のタグを特定します。
- **-e**: オプションのサービスの環境ファイルを指定します。
- **--set ceph_namespace**、**--set ceph_image**、**--set ceph_tag**: Ceph Storage を使用する場合には、Ceph Storage のコンテナ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 固有のプレフィックスが追加された点に注意してください。このプレフィックスは、**--prefix** オプションと同じ値です。以下に例を示します。

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

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

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

レジストリーの設定の準備が整いました。

4.7. 次のステップ

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

これで、オーバークラウドをアップグレードに向けて準備することができます。

第5章 オーバークラウドのアップグレードの準備

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。お使いのオーバークラウドに、本項のすべてのステップが適用されるわけではありませんが、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。この間には、オーバークラウドで実行中のワークロードはアクティブな状態のままなので、インスタンスはアップグレード中にも稼働し続けることになります。

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

オーバークラウドのアップグレードによる影響を受ける項目

- OpenStack Platform サービス

オーバークラウドのアップグレードによる影響を受けない項目

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

5.2. アップグレードテスト用のコンピュータノードの選択

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、全コンピュータ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タ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台のコンピュータ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タノードから他のコンピュータ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、全コンピュータ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のテスト用のコンピュータノードの例として、**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。

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。自分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 からの証明書を必要とするようにすることができます。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::ContainersLogrotateCron

コンテナログ用の **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** の設定ができるようにします。**environments/securetty.yaml** 環境ファイルで有効化します。

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

OS::TripleO::Services::MySQL service, such as the **Controller** またはスタンドアロンの **Database** ロールなど、**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 ノード上でマイグレーション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のサービスの最新のリストは、『**Advanced Overcloud Customization**』ガイドの「[Service Architecture: Standalone Roles](#)」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 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、他の Pacemaker サービスのコアの依存関係として機能していましたが、高可用性の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が、**director** の Heat テンプレートに直接統合されました。

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のサービスの最新のリストは、『**Advanced Overcloud Customization**』ガイドの「[Service Architecture: Standalone Roles](#)」の項を参照してください。

5.5. 非推奨パラメーター

以下のパラメーターは非推奨となり、ロール固有のパラメーターに置き換えられた点に注意してください。

旧パラメーター	新規パラメーター
controllerExtraConfig	ControllerExtraConfig
OvercloudControlFlavor	OvercloudControllerFlavor
controllerImage	ControllerImage
NovaImage	ComputeImage
NovaComputeExtraConfig	ComputeExtraConfig
NovaComputeServerMetadata	ComputeServerMetadata
NovaComputeSchedulerHints	ComputeSchedulerHints
NovaComputeIPs	ComputeIPs
SwiftStorageServerMetadata	ObjectStorageServerMetadata
SwiftStorageIPs	ObjectStorageIPs
SwiftStorageImage	ObjectStorageImage
OvercloudSwiftStorageFlavor	OvercloudObjectStorageFlavor

カスタムの環境ファイルでこれらのパラメーターを更新します。

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 사용할 수 있습니다. 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
  ...
```

5.6. 非推奨の CLI オプション

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

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

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

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

オプション	説明	Heat テンプレートのパラメーター
--neutron-public-interface	ネットワークノード向けにインターフェースを br-ex にブリッジするインターフェースを定義します。	NeutronPublicInterface
--neutron-network-type	Neutron のテナントネットワーク種別	NeutronNetworkType
--neutron-tunnel-types	neutron テナントネットワークのトンネリング種別。複数の値を指定するには、コンマ区切りの文字列を使用します。	NeutronTunnelTypes
--neutron-tunnel-id-ranges	テナントネットワークを割り当てるに使用できる GRE トンネリングの ID 範囲	NeutronTunnelIdRanges
--neutron-vni-ranges	テナントネットワークを割り当てるに使用できる VXLAN VNI の ID 範囲	NeutronVniRanges
--neutron-network-vlan-ranges	サポートされる Neutron ML2 および Open vSwitch VLAN マッピングの範囲。デフォルトでは、物理ネットワーク「datacentre」上の VLAN を許可するように設定されています。	NeutronNetworkVLANRanges
--neutron-mechanism-drivers	neutron テナントネットワークのメカニズムドライバー。デフォルトでは、「openvswitch」に設定されており、複数の値を指定するにはコンマ区切りの文字列を使用します。	NeutronMechanismDrivers
--neutron-disable-tunneling	VLAN で区切られたネットワークまたは neutron でのフラットネットワークを使用するためにトンネリングを無効化します。	パラメーターのマッピングなし

オプション	説明	Heat テンプレートのパラメーター
--validation-errors-fatal	オーバークラウドの作成プロセスでは、デプロイメントの前に一連のチェックが行われます。このオプションは、デプロイメント前のチェックで何らかの致命的なエラーが発生した場合に終了します。どのようなエラーが発生してもデプロイメントが失敗するので、このオプションを使用することを推奨します。	パラメーターのマッピングなし
--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。

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。

5.7. 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
```

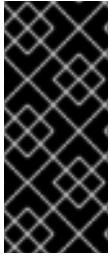
その他の構文例については、デフォルトの **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** ファイルを確認してください。また、ロールの例のスニペットについては、**/usr/share/openstack-tripleo-heat-templates/roles** を確認してください。

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

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

ロール	必要なネットワーク
Telemetry	InternalApi



重要

以前のバージョンでは、***NetName** パラメーター (例: **InternalApiNetName**) によってデフォルトのネットワークの名前が変更されていました。このパラメーターはサポートされなくなりました。カスタムのコンポーザブルネットワークファイルを使用してください。詳しくは、『**Advanced Overcloud Customization**』ガイドの「[Using Composable Networks](#)」を参照してください。

5.8. CEPH STORAGE ノードのアップグレードの準備

コンテナ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。

前提条件

- オーバークラウドに、director で管理されている Ceph Storage クラスタがあること

手順

1. **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. ストレージの環境ファイルで、最新のリソースと設定を使用するようにします。これは、以下のように変更する必要があります。
 - a. **resource_registry** は、**puppet/services** サブディレクトリーの代わりに、コア Heat テンプレートコレクションの **docker/services** サブディレクトリーからコンテナ化されたサービスを使用します。以下に例を示します。

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

上記の行を、以下のように置き換えます。

```
resource_registry:
  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` でデプロイメントのコマンドに追加することができます。

- b. 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** パラメーターの構成に変換します。たとえば、hieradata に以下の内容が記載されているとします。

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

この場合には、**CephAnsibleDisksConfig** は以下ようになります。

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

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。

3. 今後は、デプロイメントのコマンドで `-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するようにしてください。

5.9. 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。

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。

5.10. 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` パラメーターがないため、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ります。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.yaml`) 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、CA 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 Public 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しい情報は、『**director のインストールと使用方法**』ガイドの「[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。

5.11. FAST FORWARD UPGRADE の登録の設定

Fast Forward Upgrade プロセスでは、リポジトリ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 repos --enable=rhel-7-server-openstack-13-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

状況によっては、カスタム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** 環境ファイルを記載しています。

5.12. カスタムの PUPPET パラメーターの確認

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

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

手順

1. 環境ファイルを選択して、**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. コマンドの出力で、選択したファイルのいずれかのロールに **ExtraConfig** パラメーター (例: **ControllerExtraConfig**) が表示された場合には、そのファイルの全パラメーター構造を確認します。
3. **SECTION/parameter** 構文で **value** が続くいずれかの puppet Hierdata がパラメーターに含まれている場合には、実際の Puppet クラスに置き換えられている可能性があります。以下に例を示します。

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

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

```
$ grep dnsmasq_local_resolv
```

その場合には、新規インターフェースに変更します。

5. 構文の変更の実例を以下に示します。

- 例 1:

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

変更後

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

- 例 1:

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

変更後

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

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

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

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

テンプレートでは、設定に **OS::Heat::SoftwareConfig** リソースを使用するようになりました。

```
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]
```

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



警告

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

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

```
$ /usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-
```



```
script.py \  
    --script-dir /usr/share/openstack-tripleo-heat-  
templates/network/scripts \  
    [NIC TEMPLATE] [NIC TEMPLATE] ...
```



重要

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

詳しい情報は、[「Isolating Networks」](#)を参照してください。

5.14. 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](#)」に記載の手順で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 `parepare` 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. 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** ファイルを読み込みます。

```
$ source ~/stackrc
```

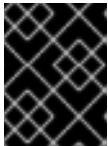
2. Fast Forward Upgrade の `parepare` コマンドを実行します。

```
$ 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。

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



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. 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. 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。

5. 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 にアップグレードされます。
- 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 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。

6.2. 全コントローラーノードのアップグレード

このプロセスでは、全コントローラーノードを OpenStack Platform 13 にアップグレードします。このプロセスは、**openstack overcloud upgrade run** コマンドに **--roles Controller** オプションを指定して、操作をコントローラーノードのみに制限して実行する必要があります。

手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

```
$ openstack overcloud upgrade run --roles Controller --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. コントローラー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。
- 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、まだバージョン 2 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。

6.3. 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 --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。

6.4. 全コンピュートノードのアップグレード

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

手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

```
$ openstack overcloud upgrade run --roles Compute --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. 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、まだバージョン 2 で、アップグレードする必要があります。

6.5. 全 CEPH STORAGE ノードのアップグレード

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

- **openstack overcloud upgrade run** コマンドに **--roles CephStorage** オプションを指定して、操作を Ceph Storage ノードのみに制限して実行します。
- **openstack overcloud ceph-upgrade run** コマンドを実行して、コンテナ化された Red Hat Ceph Storage 3 クラスタにアップグレードします。

手順

1. **stackrc** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. アップグレードのコマンドを実行します。

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

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

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。

6.6. 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 OpenStack Platform 13 の標準のデプロイメントと一致し、今後、通常の **openstack overcloud deploy** の機能を実行できるようになります。

手順

1. **stackrc** ファイルを読み込みます。

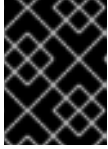
```
$ source ~/stackrc
```

2. 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 /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 <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。

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

**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。

6.7. 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、「[7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ステップの実行

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

7.1. 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手順を以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

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

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

3. アンダークラウドの空き領域を確認します。

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

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

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

5. アンダークラウドのネットワークサービスを確認します。

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

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

6. アンダークラウドの Compute サービスを確認します。

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

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

関連情報

- OpenStack Orchestration (heat) のデータベースで削除済みとマークされている stack のエントリーを完全削除する方法は <https://access.redhat.com/solutions/2215131> のソリューションに記載されています。

7.2. コンテナ化されたオーバークラウドの検証

コンテナ化されたオーバークラウドの機能を確認する手順を以下に示します。

手順

1. アンダークラウドのアクセス情報を読み込みます。

```
$ source ~/stackrc
```

2. ベアメタルノードのステータスを確認します。

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

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

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

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

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

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

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

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

以下の cURL 要求でそれらの情報を使用します。

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

<PASSWORD> と <IP ADDRESS> は、**haproxy.stats** サービスからのそれぞれの情報に置き換えます。その結果表示される一覧には、各ノード上の 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チェックします。

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

7. RabbitMQ クラスターの正常性を確認します。

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

8. Pacemaker リソースの正常性を確認します。

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

以下の点を確認します。

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

9. 各オーバークラウドノードでディスク領域を確認します。

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

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

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

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

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

12. オーバークラウドノードでクロックが同期されていることを確認します。

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

13. オーバークラウドのアクセス情報を読み込みます。

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

14. オーバークラウドのネットワークサービスを確認します。

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

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

15. オーバークラウドの Compute サービスを確認します。

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

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

16. オーバークラウドのボリュームサービスを確認します。

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

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

関連情報

- [「How can I verify my OpenStack environment is deployed with Red Hat recommended configurations?」](#)の記事を参照してください。この記事には、Red Hat OpenStack Platform 環境をチェックして、Red Hat の推奨値に合わせて調整する方法が記載されています。

7.3. 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ますようになります。

前提条件

- アンダークラウドが最新バージョンにアップグレードされていること

手順

1. **stack** ユーザーの **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/*
```

2. アーカイブを展開します。

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

3. director に最新のイメージをインポートします。

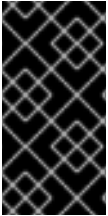
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

4. ノードが新しいイメージを使用するように設定します。

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

5. 新規イメージが存在することを確認します。

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



重要

オーバークラウドノードをデプロイする際には、オーバークラウドイメージのバージョンが、その Heat テンプレートバージョンに対応していることを確認してください。たとえば、OpenStack Platform 13 の Heat テンプレートには、OpenStack Platform 13 のイメージのみを使用してください。

7.4. 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** ファイルを読み込みます。

```
$ source ~/stackrc
```

2. 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** を使用してカスタム設定環境ファイル
- 該当する場合には、**--roles-file** を使用する、カスタムロール (**roles_data**) のファイル

3. デプロイメントが完了するまで待ちます。

7.5. 結果

これで Fast Forward Upgrade のプロセスが完全に終了しました。

付録A アンダークラウドの復元

以下の復元プロセスは、アンダークラウドノードでエラーが発生して、回復不可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. アンダークラウドに **root** ユーザーとしてログインします。
2. **stack** ユーザーを作成します。

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

3. そのユーザーのパスワードを設定します。

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

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

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

5. 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ของผู้ーザー名とパスワードを入力します。

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

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

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

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

```
[root@director ~]# sudo subscription-manager repos --disable=*  
[root@director ~]# sudo 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-13-rpms
```

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

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

10. バックアップ用に一時ディレクトリを作成します。

```
[root@director ~]# mkdir /var/tmp/undercloud_backup
```

11. ファイルシステムのバックアップアーカイブを一時ディレクトリに抽出します。

```
[root@director ~]# sudo tar -xvf /root/undercloud-backup-
[timestamp].tar -C /var/tmp/undercloud_backup --xattrs || true
```

12. **rsync** をインストールします。

```
[root@director ~]# sudo yum -y install rsync
```

13. 以下のディレクトリをバックアップのコンテンツと同期します。

```
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/home/stack/ /home/stack
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/etc/haproxy/ /etc/haproxy/
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/etc/pki/instack-certs/ /etc/pki/instack-
certs/
[root@director ~]# sudo mkdir -p /etc/puppet/hieradata/
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/etc/puppet/hieradata/
/etc/puppet/hieradata/
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/srv/node/ /srv/node/
[root@director ~]# sudo rsync -a -X
/var/tmp/undercloud_backup/var/lib/glance/ /var/lib/glance/
```

14. **openstack-keystone** パッケージをインストールして、その設定データを同期します。

```
[root@director ~]# sudo yum -y install openstack-keystone
[root@director ~]# sudo rsync -a
/var/tmp/undercloud_backup/etc/keystone/ /etc/keystone/
```

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

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


16. アンダークラウドで SSL を使用している場合には、CA 証明書をリフレッシュします。

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

17. 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 ~]# sudo yum install -y mariadb mariadb-server
python-tripleoclient
```

18. データベースを起動します。

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

19. 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;"
```

20. データベースのバックアップを復元します。

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

21. Mariadb を再起動して、バックアップファイルからパーミッションをリフレッシュします。

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

1. 古いユーザーパーミッションの一覧を取得します。

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

2. リストされた各ホストの古いユーザーパーミッションを削除します。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 ~]# mysql -e 'flush privileges'
```

3. **openstack-glance** パッケージをインストールして、そのファイルパーミッションを復元します。

```
[root@director ~]# sudo yum install -y openstack-glance
[root@director ~]# sudo chown -R glance: /var/lib/glance/images
```

4. **openstack-swift** パッケージをインストールして、そのファイルパーミッションを復元します。

```
[root@director ~]# sudo yum install -y openstack-swift
[root@director ~]# sudo chown -R swift: /srv/node
```

5. 新規作成した **stack** ユーザーに切り替えます。

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

6. 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。

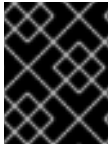
```
[stack@director ~]$ openstack undercloud install
```

7. インストールが完了するまで待ちます。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復元します。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を復元します。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。コンピュータ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。

手順

1. メジャーバージョンのアップグレードの失敗から復元する場合には、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. **yum** の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 台で発生しているはずです (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)。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。

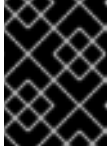
- c. アップグレードから発生した **yum** トランザクションをすべて元に戻します。以下に例を示します。

```
# 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。

- d. 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
```

2. 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接続を停止します。

```
# MYSQLIP=$(hiera 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. ブートストラップコントローラーノードで、Pacemaker による Galera の管理を無効にします。

```
# pcs resource unmanage galera
```

- d. コントローラーノード上の **/etc/my.cnf.d/galera.cnf** から **wsrep_cluster_address** パラメーターをコメントアウトします。

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

- e. すべてのコントローラーノード上の MariaDB データベースを停止します。

```
# 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. root の設定とクラスターのチェックを全コントローラーノード上のバックアップファイルに移動します。

```
# sudo mv /root/.my.cnf /root/.my.cnf.old
# sudo mv /etc/sysconfig/clustercheck
/etc/sysconfig/clustercheck.old
```

- h. ブートストラップコントローラーノードで、Pacemaker が Galera クラスターを管理するように設定します。

```
# pcs resource manage galera
# pcs resource cleanup galera
```

- i. Galera クラスターがきちんと立ち上がるのを待ちます。以下のコマンドを実行して、全ノードがマスターとして設定されているかどうかを確認します。

```
# watch "pcs status | grep -C3 galera"
Master/Slave Set: galera-master [galera]
Masters: [ overcloud-controller-0 overcloud-controller-1
overcloud-controller-2 ]
```

クリーンアップの実行後に全コントローラーがマスターとして表示されない場合には、クリーンアップコマンドを再度実行してください。

```
# pcs resource cleanup galera
```

- j. 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。これは、Galera によって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
```

- k. 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
```

- l. ブートストラップコントローラーノードで、データベースのパスワードを元のパスワードに再設定します。

```
# /usr/bin/mysqladmin -u root password "$(hieramysql::server::root_password)"
```

- m. ブートストラップコントローラーノードで、**pcs status** を実行して Galera リソースを表示します。

```
# pcs status | grep -C3 galera
```

このコマンドでエラーが表示される場合があります。これは、データベースが間違ったユーザー名とパスワードを使用して接続し、データベースのステータスをポーリングするようになったことが理由です。全コントローラーノードで、データベースの設定を復元します。

```
# sudo mv /root/.my.cnf.old /root/.my.cnf
# sudo mv /etc/sysconfig/clustercheck.old
/etcsysconfig/clustercheck
```

- n. 各コントローラーノードのクラスターチェックをローカルでテストします。

```
# /bin/clustercheck
```

- o. ブートストラップコントローラーノードで、Pacemaker のクリーンアップを実行して、Galera の状態を再度プロービングするようにします。

```
# pcs resource cleanup galera
```

- p. 各コントローラーノードでクラスターのチェックをテストします。

```
# curl overcloud-controller-0:9200
# curl overcloud-controller-1:9200
# curl overcloud-controller-2:9200
```

- q. 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復元するようにします。

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

- r. コントローラーノード上の **/etc/my.cnf.d/galera.cnf** から **wsrep_cluster_address** パラメーターをコメント解除します。

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

3. 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/data/
# cd /var/tmp/filesystem_backup/data/
# mv <backup_file>.tar.gz .
# tar -xvzf --xattrs <backup_file>.tar.gz
```



注記

/ ディレクトリーには直接展開しないようにしてください。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。一時ディレクトリーで抽出することを推奨します。

- b. **/usr/libexec/os-apply-config/templates/etc/os-net-config/config.json** ファイルを復元します。

```
$ cp /var/tmp/filesystem_backup/data/usr/libexec/os-apply-config/templates/etc/os-net-config/config.json /usr/libexec/os-apply-config/templates/etc/os-net-config/config.json
```

- c. 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。

4. redis リソースをクリーンアップします。

```
# pcs resource cleanup redis
```

5. 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. 復元した高可用性サービス

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

コントローラーサービス

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

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

コントローラーサービス
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
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

コントローラーサービス
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

Compute サービス
openstack-nova-compute
openvswitch
os-collect-config

付録C NFV の更新用の YAML ファイルのサンプル

これらの YAML ファイルのサンプルは、OVS-DPDK デプロイメント向けの OVS を更新します。

C.1. RED HAT OPENSTACK PLATFORM 10 OVS 2.9 の更新ファイル

C.1.1. post-install.yaml

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json

resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers: {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE', 'UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config: |
        #!/bin/bash
        set -x
        function tuned_service_dependency() {
          tuned_service=/usr/lib/systemd/system/tuned.service
          grep -q "network.target" $tuned_service
          if [ "$?" -eq 0 ]; then
            sed -i '/After=.*s/network.target//g' $tuned_service
          fi
          grep -q "Before=.*network.target" $tuned_service
          if [ ! "$?" -eq 0 ]; then
            grep -q "Before=.*" $tuned_service
            if [ "$?" -eq 0 ]; then
              sed -i 's/^\(Before=.*\)\/\1 network.target
openvswitch.service/g' $tuned_service
            else
              sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
            fi
          fi
        }

        if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
```

```
neutron_ovs_dpdk_agent; then
    tuned_service_dependency
fi
```