Red Hat OpenStack Platform 13

Red Hat OpenStack Platform の最新状態の維持

Red Hat OpenStack Platform のマイナー更新の実施

Last Updated: 2023-04-10

Red Hat OpenStack Platform 13 Red Hat OpenStack Platform の最新状態の維持

Red Hat OpenStack Platform のマイナー更新の実施

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本書では、Red Hat OpenStack Platform 13 (Queens) 環境を更新する手順について説明します。本書では、Red Hat Enterprise Linux 7 をベースにインストールされたコンテナー化された OpenStack Platform デプロイメントを更新することを前提としています。

目次

第1章 はじめに

本書で説明するワークフローは、お使いの Red Hat OpenStack Platform 13 環境が最新のパッケージおよびコンテナーで更新された状態を維持するのに役立ちます。

本ガイドは、以下のバージョンのアップグレードパスを提供します。

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

1.1. ワークフローの概要

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

ステップ	説明
新しいコンテナーイメージの取得	OpenStack Platform 13 のサービス用の最新コンテナーイメージが含まれる新しい環境ファイルを作成します。
アンダークラウドの更新	アンダークラウドを最新の OpenStack Platform 13.z バージョンに更新します。
オーバークラウドの更新	オーバークラウドを最新の OpenStack Platform 13.z バージョンに更新します。
Ceph Storage ノードの更新	すべての Ceph Storage 3 サービスをアップグレードします。
アップグレードの最終段階	コンバージェンスのコマンドを実行して、オーバークラウドスタックをリフレッシュします。

1.2. 更新を妨げる可能性のある既知の問題

マイナーバージョンの更新の正常な完了に影響を及ぼす可能性のある、以下の既知の問題を確認してください。

Red Hat Ceph Storage 3 のマイナー 更新により、OSD が破損する可能性がある

Red Hat Ceph Storage 3 は、EL7 で実行されるコンテナー化デプロイメントについては docker に依存します。BZ#1846830 の ceph-ansible 修正により、Ceph コンテナーを制御する systemd ユニットが更新され、systemd ユニットを実行するには docker サービスが稼働している必要があります。この要件は、安全な更新パスを実装して、制御されていない docker パッケージの更新時におけるサービスの中断やデータ破損も回避するのに不可欠です。
ceph-ansible パッケージを更新するだけでは、ceph-ansible 修正の有効化には十分ではありません。また、デプロイメント Playbook を再実行して、コンテナーの systemd ユニットを更新する必要があります。director 主導型の Ceph Storage デプロイメントの問題を解決する方法は、Red Hat ナレッジベースのソリューション Issue affecting minor updates of Red Had Ceph Storage 3 can cause OSDs corruption を参照してください。

OSP13 update may appear to fail while it's eventually successful

openstack overcloud update run コマンドで使用される python **tripleo-client** は、更新プロセスが完了する前にタイムアウトになる可能性があります。これにより、**openstack overcloud update run** コマンドが失敗を返し、更新プロセスはバックグラウンドで実行を継続しますが、それが完了するまでバックグラウンドで実行を継続します。
この失敗を回避するには、オーバークラウドノードを更新する前に、**tripleo-client/plugin.py** ファイルの **ttl** パラメーターの値を編集して、**tripleo-client** タイムアウト値を増やします。詳細は、Red Hat ナレッジベースのソリューション OSP 13 update process appears to fail while the update process runs in the background and completes successfully を参照してください。

Slight cut in rabbitmq connectivity triggered a data plane loss after a full sync

RHOSP 13 z10(2019 年 12 月 19 日メンテナーンスリリース) よりも前のリリースから環境を更新する場合は、バグ BZ#1955538 で説明されているデータプレーンの接続損失を回避するため、Red Hat ナレッジベースソリューション Stale namespaces on OSP13 can create data plane cut during update を参照してください。

ceph のアップグレード中、すべての OSD (およびその他の ceph サービス) がダウンする

Ceph を使用している場合には、以下の手順を実施する前に、バグ BZ#1910842 を回避するために Red Hat ナレッジベースのソリューション During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted を確認してください。

- すべてのコントローラーノードの更新
- 全 HCI コンピュートノードの更新
- すべての Ceph Storage ノードの更新

z11 アップグレード後の Octavia および LB の問題

更新時に、**/var/lib/config-data/puppet-generated/octavia/etc/octavia/conf.d/common/post-deploy.conf** という名前のファイルがないため、load-balancing サービス (Octavia) コンテナーが再起動を繰り返します。このファイルは、Amphora のデプロイメント後に octavia サービスを設定するために Red Hat OpenStack Platform 13 のライフサイクル中に導入されました。このファイルは現在、更新の **openstack overcloud update converge** ステップで生成されます。この問題を回避するには、更新を続行する必要があります。octavia コンテナーは、**openstack overcloud update converge** コマンドの実行後に通常起動します。現在、Red Hat OpenStack Platform のエンジニアリングチームは、この問題に対する解決策を調査しています。

DBAPIError exception wrapped from (pymysql.err.InternalError) (1054, u"Unknown column 'pool.tls_certificate_id' in 'field list'")

load-balancing サービス (octavia) を使用していて、RHOSP 13 z13 (2020 年 10 月 8 日メンテナーンスリリース) 以前のリリースから更新する場合、バグ BZ#1927169 を回避するために、load-balancing サービスをアップグレードするデータベース移行を正しい順序で実行する必要があります。ブートストラップコントローラーノードを更新しないと、残りのコントロールプレーンを更新することができません。

1. 現在のメンテナーンスリリースを特定するには、以下のコマンドを実行します。

```
$ cat /etc/rhosp-release
```

2. ブートストラップコントローラーノードを特定するには、アンダークラウドノードで以下のコマンドを実行します。その際、**<any_controller_node_IP_address>** は、デプロイメント内のいずれかのコントローラーノードの IP アドレスに置き換えます。

```
$ ssh heat-admin@<any_controller_node_IP_address> sudo hiera -c
/etc/puppet/hiera.yaml octavia_api_short_bootstrap_node_name
```

3. アンダークラウドノードで **openstack overcloud update run** コマンドを実行し、ブートストラップコントローラーノードを更新します。

```
$ openstack overcloud update run --nodes <bootstrap_node_name>
```

13z16 へのマイナー 更新が "Unable to find constraint" というエラーで失敗する

Red Hat OpenStack Platform 13z16 オーバークラウドノードの更新を再開すると、**Unable to find constraint** というエラーが発生する場合があります。このエラーは、更新中に RabbitMQ のバージョンが一致しないために発生します。新しい RabbitMQ バージョンを確実に起動できるようにするには、オーバークラウドに存在する可能性があるpacemaker 禁止をクリアする必要があります。
この問題の詳細は、Red Hat ナレッジベースのソリューション Cannot restart Update of the OSP13z16 controllers を参照してください。

コントローラーで ceph-mon を停止できない。エラー: No such container: ceph-mon controller-2

Red Hat Ceph Storage バージョン 3.3 z5 以前を使用し、docker パッケージを docker-1.13.1-209 に更新すると、RHOSP 13 の更新が失敗します。RHOSP 13 の更新では、docker パッケージが更新される前に ceph-mon コンテナーは停止しません。これにより、孤立した ceph-mon プロセスが発生し、新しい ceph-mon コンテナーの開始がブロックされます。
この問題の詳細は、Red Hat ナレッジベースのソリューション Updating Red Hat OpenStack Platform 13.z12 and older with Ceph Storage may fail during controller update を参照してください。

1.3. トラブルシューティング

- 更新プロセスにかかる時間が予想よりも長い場合は、error: **socket is already closed** でタイムアウトする可能性があります。これは、アンダークラウドの認証トークンが、設定した期間後に期限切れに設定されるために発生する可能性があります。詳細は、Recommendations for Large Deployments を参照してください。

第2章 コンテナーイメージソースの更新

本章では、Red Hat OpenStack Platform 用の新しいオーバークラウドコンテナーイメージにより、レジストリーソースを更新する方法について説明します。

コンテナーイメージのソース更新前の考慮事項

コンテナーイメージソースをあるレジストリータイプから別のレジストリータイプに変更する場合は、次のタスクを完了する前に、現在のコンテナーイメージ環境ファイルの名前空間と接頭辞を更新し、**openstack overcloud deploy** コマンドを実行して名前空間を変更する必要があります。

- 「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」
- 「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」
- 「Satellite サーバーをレジストリーとして使用する手順」

2.1. レジストリーメソッド

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

リモートレジストリー

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

ローカルレジストリー

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

注記

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

Satellite Server

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

上記のリストから方法を選択し、レジストリー情報の設定を続けます。

注記

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

2.2. コンテナーイメージの準備コマンドの使用方法

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

オーバークラウド用のコンテナーイメージ環境ファイルの生成

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

--output-env-file

作成される環境ファイルの名前を定義します。

以下のスニペットは、このファイルの内容の例を示しています。

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

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

インポート方法に対応したコンテナーイメージ一覧の生成

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

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

--output-images-file

作成されるインポート一覧のファイル名を定義します。

このファイルの内容の例を以下に示します。

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

コンテナーイメージの名前空間の設定

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

--namespace

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

--prefix

イメージ名の前に追加する接頭辞を定義します。

その結果、director は以下のような形式のイメージ名を生成します。

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

コンテナーイメージタグの設定

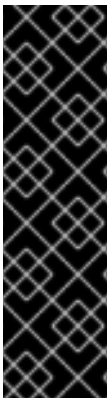
--tag および **--tag-from-label** オプションを併用して、各コンテナーイメージのタグを設定します。

--tag

ソースからの全イメージに特定のタグを設定します。このオプションだけを使用した場合、director はこのタグを使用してすべてのコンテナーイメージをプルします。ただし、このオプションを **--tag-from-label** の値と共に使用する場合は、director はソースイメージとして **--tag** を使用して、ラベルに基づいて特定のバージョンタグを識別します。**--tag** オプションは、デフォルトで **latest** に設定されます。

--tag-from-label

指定したコンテナーイメージラベルの値を使用して、全イメージのバージョン付きタグを検出してプルします。director は **--tag** に設定した値がタグ付けされた各コンテナーイメージを検査し、続いてコンテナーイメージラベルを使用して新しいタグを構築し、レジストリーからプルします。たとえば、**--tag-from-label {version}-{release}** を設定すると、director は **version** および **release** ラベルを使用して新しいタグを作成します。あるコンテナーについて、**version** を **13.0** に設定し、**release** を **34** に設定した場合、タグは **13.0-34** となります。



重要

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

2.3. 追加のサービス用のコンテナーイメージ

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

-e

追加のコンテナーイメージを有効化するための環境ファイルを指定します。

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

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

以下の項には、追加するサービスの例を記載します。

Ceph Storage

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

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

--set ceph_namespace

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

--set ceph_image

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

--set ceph_tag

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

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

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

OpenStack Bare Metal (ironic)

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

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

OpenStack Data Processing (sahara)

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

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

OpenStack Neutron SR-IOV

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

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

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

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

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

OpenStack Shared File System (manila)

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

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

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

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

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

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

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

手順

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

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

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

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

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

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

注記

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

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

レジストリーの設定が完了しました。

2.5. ローカルレジストリーとしてアンダークラウドを使用する方法

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

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

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

手順

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

```
<REGISTRY_IP_ADDRESS>:8787
```

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

2. **registry.redhat.io** にログインします。

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

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

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

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

4. 次の 2 つのファイルが作成されていることを確認します。

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

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

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

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

注記

コンテナーイメージは、およそ 10 GB のディスク領域を使用します。

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

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

注記

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

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

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

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

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

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

レジストリーの設定が完了しました。

2.6. SATELLITE サーバーをレジストリーとして使用する手順

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

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

手順

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

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

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

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

注記

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

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

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

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

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

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

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

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

このカスタム製品に、イメージを保管します。

6. 製品にベースコンテナーイメージを追加します。

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

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

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

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

8. コンテナーイメージを同期します。

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

Satellite Server が同期を完了するまで待ちます。

注記

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

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

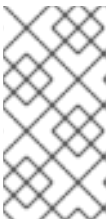
10. **base** イメージに使用可能なタグを確認します。

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

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

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

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



注記

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

このコマンドを実行する際には、以下の情報を含めてください。

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

注記

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

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

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

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

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

レジストリーの設定が完了しました。

2.7. 次のステップ

コンテナーイメージのソースの一覧が記載された新しい **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、更新に向けてオーバークラウドを準備することができます。

第3章 アンダークラウドのアップグレード

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

3.1. アンダークラウドのマイナー更新の実行

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

手順

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

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

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

```
$ openstack undercloud upgrade
```

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

```
$ sudo reboot
```

6. ノードがブートするまで待ちます。

3.2. オーバークラウドイメージの更新

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

前提条件

- アンダークラウドが最新バージョンに更新されていること

手順

1. **stack** ユーザーのホーム下の **images** ディレクトリー (**/home/stack/images**) から既存のイメージを削除します。

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

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

```
$ source ~/stackrc
```

4. director に最新のイメージをインポートします。

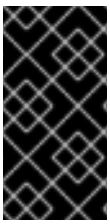
```
$ openstack overcloud image upload --update-existing --image-path /home/stack/images/
```

5. ノードが新しいイメージを使用するように設定します。

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

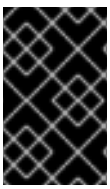
6. 新規イメージが存在することを確認します。

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



重要

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



重要

新しい **overcloud-full** イメージは、古い **overcloud-full** イメージを置き換えます。古いイメージに変更を加えた場合、特に今後新規ノードをデプロイする場合には、新しいイメージで変更を繰り返す必要があります。

3.3. アンダークラウドアップグレード後の注意事項

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

3.4. 次のステップ

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

第4章 オーバークラウドの更新

以下の手順では、オーバークラウドを更新します。

前提条件

- アンダークラウドが最新バージョンに更新されていること

4.1. オーバークラウド更新のスピードアップ

オーバークラウドの更新プロセスを迅速化するには、**DockerPuppetProcessCount** heat パラメーターを設定し、削除されたデータベースエントリーをアーカイブし、更新を実施する前にオーバークラウドノードに必要なパッケージをダウンロードします。

大規模な OpenStack デプロイメントの更新プロセスの迅速化に関する詳しい情報は、Red Hat ナレッジベースのアーティクル Openstack Director Node Performance Tuning for large deployments を参照してください。

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. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

3. 設定ファイルの生成に **container-puppet** が使用する同時プロセスの数を増やすには、**DockerPuppetProcessCount** パラメーターを設定する必要があります。
   a. **templates** ディレクトリーに **updates-environment.yaml** という名前の環境ファイルを作成します。

```
$ touch ~/templates/updates-environment.yaml
```

   b. ファイルを編集し、以下の内容を追加します。

```
parameter_defaults:
  DockerPuppetProcessCount: 8
```

   c. **-e** オプションを使用して、**openstack overcloud update prepare**、**openstack overcloud ceph-upgrade run**、および **openstack overcloud update converge** コマンドを実行するときにこの環境ファイルを含めます。
4. コントローラーノードで、削除したデータベースエントリーをアーカイブします:
   a. オーバークラウドから、コントローラーノードのすべてのインスタンスを一覧表示します。

```
$ source ~/overcloudrc
$ openstack server list
```

   b. **nova_api_cron** コンテナーを実行しているコントローラーノードにログオンします。

```
ssh heat-admin@<controller_ip>
```

- **<controller name or IP>** をコントローラーノードの IP アドレスに置き換えます。

c. 削除されたデータベースエントリーをアーカイブします。

```
$ sudo docker exec -u 42436 -ti nova_api_cron bash
$ nova-manage db archive_deleted_rows --max_rows 1000
$ exit
```

5. すべてのオーバークラウドノードで更新に必要なパッケージをすべてダウンロードするには、以下の手順を実施します。

a. オーバークラウドの静的なインベントリーファイルを作成します。

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

b. 以下の Ansible Playbook を作成します。

```
$ cat > ~/yum-download-only.yaml <<'EOF'
- hosts: all
  gather_facts: false
  tasks:
    - name: Pre-download all packages on all overcloud nodes
      shell:
        yum upgrade -y --downloadonly
      become: true
EOF
```

c. Ansible Playbook **yum-download-only.yaml** を実行します。

```
$ ansible-playbook \
-i ~/inventory.yaml \
-f 20 ~/yum-download-only.yaml \
--limit Controller,Compute,CephStorage
```

4.2. カスタムロールの考慮

デプロイメントにカスタムロールが含まれている場合は、ロールファイルで以下の値を確認します。

- カスタムロールファイルを **/usr/share/openstack-tripleo-heat-templates/roles** ディレクトリーの最新ファイルと比較します。ご利用の環境に関連するロールの **RoleParametersDefault** セクションから、カスタムロールファイルの同等のロールに新規パラメーターを追加します。

- Data Plane Development Kit (DPDK) を使用し、13.4 以前からアップグレードする場合は、OVS-DPDK サービスが含まれるロールに以下の必須パラメーターも含まれていることを確認してください。

```
RoleParametersDefault:
  VhostuserSocketGroup: "hugetlbfs"
  TunedProfileName: "cpu-paritioning"
  NovaLibvirtRxQueueSize: 1024
  NovaLibvirtTxQueueSize: 1024
```

4.3. オーバークラウドの更新準備タスクの実施

更新するには、**openstack overcloud update prepare** コマンドを実行する必要があります。このコマンドにより、次のタスクが実行されます。

- オーバークラウドのプランを OpenStack Platform 13 に更新する
- 更新に向けてノードを準備する

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 更新準備コマンドを実行します。

```
$ openstack overcloud update prepare \
  --templates \
  -r <ROLES DATA FILE> \
  -n <NETWORK DATA FILE> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/updates-environment.yaml \
  -e <ENVIRONMENT FILE> \
  -e <ENVIRONMENT FILE> \
  --stack <STACK_NAME>
  ...
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- カスタム設定環境ファイル (**-e**)
- 新しいコンテナーイメージの場所が記載された環境ファイル (**-e**)。更新のコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** の使用が推奨されるようになっているので、この警告は無視して問題ありません。
- 専用のカスタムロールを使用する場合は、カスタムロール (**roles_data**) ファイルを追加します (**-r**)。
- カスタムネットワークを使用する場合は、コンポーザブルネットワーク (**network_data**) ファイルを追加します (**-n**)。
- 高可用性クラスターをデプロイする場合は、更新の準備コマンドに **--ntp-server** オプションを追加するか、または環境ファイルに **NtpServer** パラメーターおよび値を追加します。
- オーバークラウドスタックの名前がデフォルトの名前 **overcloud** とは異なる場合は、更新の準備コマンドに **--stack** オプションを追加し、**<STACK_NAME>** を実際のスタック名に置き換えます。

3. 更新の準備が完了するまで待ちます。

4.4. CEPH STORAGE クラスターの更新

このプロセスは、Ceph Storage クラスターを更新します。このプロセスでは、**openstack overcloud ceph-upgrade run** コマンドを実行して、Red Hat Ceph Storage 3 クラスターへの更新を実施します。

注記

以下の Ansible と **ceph-ansible** の組み合わせがサポートされます。

- **ansible-2.6** と **ceph-ansible-3.2** の組み合わせ
- **ansible-2.4** と **ceph-ansible-3.1** の組み合わせ

お使いの環境に **ceph- ansible-3.1 を含む ansible-2.** 6 がある場合は、 **ceph-ansible** を最新バージョンに更新します。

```
# subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
# subscription-manager repos --enable=rhel-7-server-ansible-2.6-rpms
# yum update ceph-ansible
```

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

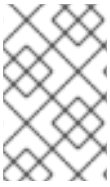
2. Ceph Storage の更新コマンドを実行します。以下に例を示します。

```
$ openstack overcloud ceph-upgrade run \
  --templates \
  -e <ENVIRONMENT FILE> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/updates-environment.yaml
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- カスタム設定環境ファイル (**-e**)
- コンテナーイメージの場所が記載された環境ファイル (**-e**)。更新のコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるため、この警告は無視して問題ありません。
- 該当する場合は、カスタムロール (**roles_data**) のファイル (**--roles-file**)
- 該当する場合は、コンポーザブルネットワーク (**network_data**) のファイル (**--networks-file**)

3. Ceph Storage ノードの更新が完了するまで待ちます。



注記

手順中に Heat スタックがタイムアウトになった場合は、Red Hat ナレッジベースの記 During sequential update of Ceph nodes **openstack overcloud ceph-upgrade run** appears to time out を参照してください。

4.5. すべてのコントローラーノードの更新

コントローラーノードを最新の Red Hat OpenStack Platform (RHOSP) 13 バージョンに更新するには、**openstack overcloud update run** コマンドに **--nodes Controller** オプションを追加します。**--nodes Controller** オプションは、更新操作をコントローラーノードのみに制限します。

Warning

Ceph を使用している場合には、コントローラーノードを更新する前に、バグ BZ#1910842 を回避するために Red Hat ナレッジベースのソリューション During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted を確認してください。

前提条件

- load-balancing サービス (octavia) を使用していて、RHOSP 13 z13 (2020 年 10 月 8 日メンテナーンスリリース) 以前のリリースから更新する場合、バグ BZ#1927169 を回避するために、load-balancing サービスをアップグレードするデータベース移行を正しい順序で実行する必要があります。ブートストラップコントローラーノードを更新しないと、残りのコントロールプレーンを更新することができません。

  1. 現在のメンテナーンスリリースを特定するには、以下のコマンドを実行します。

     ```
     $ cat /etc/rhosp-release
     ```

  2. ブートストラップコントローラーノードを特定するには、アンダークラウドノードで以下のコマンドを実行します。その際、**<any_controller_node_IP_address>** は、デプロイメント内のいずれかのコントローラーノードの IP アドレスに置き換えます。

     ```
     $ ssh heat-admin@<any_controller_node_IP_address> sudo hiera -c
     /etc/puppet/hiera.yaml octavia_api_short_bootstrap_node_name
     ```

  3. アンダークラウドノードで **openstack overcloud update run** コマンドを実行し、ブートストラップコントローラーノードを更新します。

     ```
     $ openstack overcloud update run --nodes <bootstrap_node_name>
     ```

手順

1. **stackrc** ファイルを取得します。

   ```
   $ source ~/stackrc
   ```

2. 更新コマンドを実行します。

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

3. コントローラーノードの更新が完了するまで待ちます。

4.6. 全コンピュートノードの更新

以下の手順では、全コンピュートノードを最新バージョンの OpenStack Platform 13 に更新します。このプロセスでは、**--nodes Compute** オプションを指定して **openstack overcloud update run** コマンドを実行し、操作をコンピュートノードだけに制限します。

並列処理に関する考慮事項

多数のコンピュートノードをアップグレードする場合は、パフォーマンスを向上させるために、**--nodes Compute** オプションを指定して **openstack overcloud upgrade run** コマンドを実行し、20 ノードのバッチを並行して処理することができます。たとえば、デプロイメントに 80 のコンピュートノードがある場合、次のコマンドを実行して、コンピュートノードを並行して更新できます。

```
$ openstack overcloud update run --nodes 'Compute[0:19]' > update-compute-0-19.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[20:39]' > update-compute-20-39.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[40:59]' > update-compute-40-59.log 2>&1 &
$ openstack overcloud update run --nodes 'Compute[60:79]' > update-compute-60-79.log 2>&1 &
```

'Compute[0:19]'、**'Compute[20:39]'**、**'Compute[40:59]'**、および **'Compute[60:79]'** のノード領域分割方法はランダムで、各バッチでノードが更新される順序を制御することはできません。

特定のコンピュートノードを更新するには、バッチで更新するノードをコンマ区切りリストで指定します。

```
$ openstack overcloud update run --nodes <Compute0>,<Compute1>,<Compute2>,<Compute3>
```

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 更新コマンドを実行します。

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

3. コンピュートノードの更新が完了するまで待ちます。

4.7. 全 HCI コンピュートノードの更新

以下の手順では、ハイパーコンバージドインフラストラクチャー (HCI) コンピュートノードを更新します。このプロセスでは、**--nodes ComputeHCI** オプションを指定して **openstack overcloud update run** コマンドを実行し、操作を HCI ノードのみに制限します。

Warning

Ceph を使用している場合には、本セクションの手順を実施する前に、バグ BZ#1910842 を回避するために Red Hat ナレッジベースのソリューション During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted を確認してください。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. 更新コマンドを実行します。

```
$ openstack overcloud update run --nodes ComputeHCI
```

3. ノードの更新が完了するまで待ちます。

4.8. すべての CEPH STORAGE ノードの更新

以下の手順では、Ceph Storage ノードを更新します。このプロセスでは、**--nodes CephStorage** オプションを指定して **openstack overcloud update run** コマンドを実行し、操作を Ceph Storage ノードのみに制限します。

Warning

Ceph を使用している場合には、本セクションの手順を実施する前に、バグ BZ#1910842 を回避するために Red Hat ナレッジベースのソリューション During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted を確認してください。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. グループノードを更新します。
グループ内のすべてのノードを更新するには、以下のコマンドを実行します。

```
$ openstack overcloud update run --nodes <GROUP_NAME>
```

グループ内の単一ノードを更新するには、以下のコマンドを実行します。

```
$ openstack overcloud update run --nodes <GROUP_NAME> [NODE_INDEX]
```

注記

ノードを個別に更新する場合は、必ずすべてのノードを更新してください。

グループ内の最初のノードのインデックスはゼロ (0) です。たとえば、**CephStorage** という名前のグループの最初のノードを更新するには、次のコマンドを実行します。

openstack overcloud update run --nodes CephStorage[0]

3. ノードの更新が完了するまで待ちます。

4.9. 更新の最終処理

更新には、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 Red Hat 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** ファイルを取得します。

```
$ source ~/stackrc
```

2. 更新の最終処理のコマンドを実行します。

```
$ openstack overcloud update converge \
  --templates \
  --stack <STACK_NAME> \
  -e /home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/updates-environment.yaml \
  -e <ENVIRONMENT FILE>
  ...
```

以下のオプションの中で、お使いの環境に適切なオプションを追加します。

- カスタム設定環境ファイル (**-e**)
- 新しいコンテナーイメージの場所が記載された環境ファイル (**-e**)。更新のコマンドで **--container-registry-file** の使用に関する警告が表示される場合があることに注意してください。このオプションは非推奨になり、コンテナーイメージの環境ファイルには **-e** を使用することが推奨されるようになっているので、この警告は無視して問題ありません。
- 該当する場合は、カスタムロール (**roles_data**) のファイル (**--roles-file**)
- 該当する場合は、コンポーザブルネットワーク (**network_data**) のファイル (**--networks-file**)
- オーバークラウドスタックの名前がデフォルトの名前 **overcloud** とは異なる場合は、更新の準備コマンドに **--stack** オプションを追加し、**<STACK_NAME>** を実際のオーバークラウドスタック名に置き換えます。

3. 更新の最終処理が完了するまで待ちます。

第5章 オーバークラウドのリブート

Red Hat OpenStack バージョンのマイナー更新後に、オーバークラウドをリブートします。リブートにより、関連付けられたカーネル、システムレベル、およびコンテナーコンポーネントの更新と共にノードがリフレッシュされます。これらの更新により、パフォーマンスとセキュリティー上のメリットが得られます。

ダウンタイムを計画して、以下のリブート手順を実行します。

5.1. コントローラーノードおよびコンポーザブルノードのリブート

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

手順

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

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

3. ノードをリブートします。

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

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

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

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

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

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

5.2. CEPH STORAGE (OSD) クラスターのリブート

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

手順

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

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

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

```
$ sudo reboot
```

4. ノードがブートするまで待ちます。
5. Ceph MON またはコントローラーノードにログインし、クラスターのステータスを確認します。

```
$ sudo ceph -s
```

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

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

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

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

```
$ sudo ceph status
```

5.3. コンピュートノードのリブート

コンピュートノードをリブートするには、以下のワークフローを実施します。

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

手順

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

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

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

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

4. コンピュートノード上の全インスタンスを一覧表示します。

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

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

6. コンピュートノードにログインして、リブートします。

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

7. ノードがブートするまで待ちます。

8. コンピュートノードを有効にします。

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

9. コンピュートノードが有効化されていることを確認します。

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

5.4. コンピュート HCI ノードのリブート

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

手順

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

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

2. アンダークラウドに **stack** ユーザーとしてログインします。

3. 全コンピュートノードとその UUID を一覧表示します。

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

リブートするコンピュートノードの UUID を特定します。

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

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

5. コンピュートノード上の全インスタンスを一覧表示します。

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

6. 以下のコマンドの1つを使用して、インスタンスを移行します。

   a. 選択した特定のホストにインスタンスを移行する。

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

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

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

   c. 一度にすべてのインスタンスのライブマイグレーションを行う。

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

   注記

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

7. 移行が完了するまで待ちます。

8. 移行が正常に完了したことを確認します。

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

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

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

```
$ sudo ceph -s
```

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

11. コンピュート HCI ノードをリブートします。

```
$ sudo reboot
```

12. ノードがブートするまで待ちます。

13. コンピュートノードを再度有効化します。

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

14. コンピュートノードが有効化されていることを確認します。

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

15. ノードからログアウトして、次のノードをリブートし、ステータスを確認します。全 Ceph Storage ノードがリブートされるまで、このプロセスを繰り返します。

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

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

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

```
$ sudo ceph status
```

第6章 更新後操作の実施

Red Hat OpenStack のマイナーバージョンを更新したら、更新後の関連操作を実施して、環境が完全にサポートされ、これ以降の操作を行う準備が整っている状態にする必要があります。

6.1. OCTAVIA のデプロイメントで考慮すべき事項

デプロイメントで Octavia サービスが使用される場合、実行中の負荷分散 amphora インスタンスを新しいイメージで更新する必要があります。

amphora イメージを更新するには、ロードバランサーをフェイルオーバーし、続いてロードバランサーが再びアクティブな状態になるのを待つ必要があります。ロードバランサーがアクティブな状態に戻ると、新しいイメージを実行します。

詳細は、**Using Octavia for Load Balancing-as-a-Service**の Updating running Load-balancing service instances を参照してください。