Red Hat OpenStack Platform 13

コンテナー化されたサービスへの移行

コンテナー化された OpenStack Platform サービスの操作に関する基本ガイド

Last Updated: 2023-04-10

Red Hat OpenStack Platform 13 コンテナー化されたサービスへの移行

コンテナー化された OpenStack Platform サービスの操作に関する基本ガイド

OpenStack Team
rhos-docs@redhat.com

法律上の通知

概要

本ガイドは、コンテナーで実行される OpenStack Platform サービスの操作に慣れるのに役立つ基本情報をユーザーに提供します。

目次

第1章 はじめに

以前のバージョンの Red Hat OpenStack Platform は、Systemd の管理するサービスを使用していました。しかし、より新しいバージョンの OpenStack Platform は、コンテナーを使用してサービスを実行するようになりました。コンテナー化された OpenStack Platform サービスがどのように動作するか、理解が十分ではない管理者もいます。本ガイドの目的は、OpenStack Platform のコンテナーイメージおよびコンテナー化されたサービスを理解するのに役立つ情報を提供することです。ここでは、以下の点について説明します。

- コンテナーイメージを取得および変更する方法
- コンテナー化されたサービスをオーバークラウドで管理する方法
- コンテナーと Systemd サービスの相違点の理解

本書の主目的は、Systemd ベースの環境からコンテナーベースの環境に移行するために、コンテナー化された OpenStack Platform サービスに関する十分な知識を習得するのに役立つ情報を提供することです。

1.1. コンテナー化されたサービスおよび KOLLA

Red Hat OpenStack Platform (RHOSP) の各主要サービスは、コンテナー内で実行されます。このことにより、それぞれのサービスが、ホストから独立した専用の分離名前空間内に維持されます。これには次の効果があります。

- デプロイ中、RHOSP は Red Hat カスタマーポータルからコンテナーイメージをプルして実行する。
- **podman** コマンドは、サービスの起動や停止などの管理機能を実行する。
- コンテナーをアップグレードするには、新しいコンテナーイメージをプルし、既存のコンテナーを新しいバージョンのコンテナーに置き換える必要がある。

Red Hat OpenStack Platform は、**Kolla** ツールセットによりビルド/管理されるコンテナーセットを使用します。

第2章 コンテナーイメージの取得および変更

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

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

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**

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

以下の項には、追加するサービスの例を記載します。

Ceph Storage

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

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

--set ceph_namespace

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

--set ceph_image

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

--set ceph_tag

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

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

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

OpenStack Bare Metal (ironic)

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

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

OpenStack Data Processing (sahara)

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

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

OpenStack Neutron SR-IOV

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

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

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

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

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

OpenStack Shared File System (manila)

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

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

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

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

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
  ```

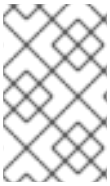
  これにより、オーバークラウドは Satellite の命名規則の Ceph コンテナーイメージを使用することができます。

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. コンテナーイメージの変更

Red Hat は、Red Hat Container Catalog (**registry.redhat.io**) で、事前にビルドされたコンテナーイメージを提供しています。これらのイメージを修正して、さらにレイヤーを追加することができます。これは、認定済みのサードパーティードライバーの RPM をコンテナーに追加する場合に役立ちます。



注記

修正された OpenStack Platform コンテナーイメージが継続的にサポートされるには、修正後のイメージが Red Hat コンテナーのサポートポリシー を順守することを確認してください。

最新の **openstack-keystone** イメージをカスタマイズする方法を、以下の例で説明します。ただし、これらの手順は、他のイメージにも適用することができます。

手順

1. 編集するイメージをプルします。たとえば、**openstack-keystone** イメージの場合には、以下のコマンドを実行します。

```
$ sudo docker pull registry.redhat.io/rhosp13/openstack-keystone:latest
```

2. 元のイメージで、デフォルトのユーザーを確認します。たとえば、**openstack-keystone** イメージの場合には、以下のコマンドを実行します。

```
$ sudo docker run -it registry.redhat.io/rhosp13/openstack-keystone:latest whoami
root
```

注記

openstack-keystone イメージは、**root** をデフォルトユーザーとして使用します。その他のイメージは、特定のユーザーを使用します。たとえば、**openstack-glance-api** はデフォルトユーザーに **glance** を使用します。

3. **Dockerfile** を作成して、既存のコンテナーイメージ上に追加のレイヤーを構築します。Container Catalog から最新の OpenStack Identity (keystone) イメージをプルして、カスタムの RPM ファイルをイメージにインストールする例を以下に示します。

```
FROM registry.redhat.io/rhosp13/openstack-keystone
MAINTAINER Acme
LABEL name="rhosp13/openstack-keystone-acme" vendor="Acme" version="2.1"
release="1"

# switch to root and install a custom RPM, etc.
USER root
COPY custom.rpm /tmp
RUN rpm -ivh /tmp/custom.rpm

# switch the container back to the default user
USER root
```

4. 新規イメージをビルドして、タグ付けします。たとえば、**/home/stack/keystone** ディレクトリーに保管されたローカルの **Dockerfile** でビルドして、アンダークラウドのローカルレジストリーにタグ付けするには、以下のコマンドを実行します。

```
$ docker build /home/stack/keystone -t "192.168.24.1:8787/rhosp13/openstack-keystone-
acme:rev1"
```

5. 編集が終わったイメージをアンダークラウドのローカルレジストリーにプッシュします。

```
$ docker push 192.168.24.1:8787/rhosp13/openstack-keystone-acme:rev1
```

6. オーバークラウドコンテナーイメージの環境ファイル (通常は **overcloud_images.yaml**) を編集して、カスタムのコンテナーイメージを使用するための適切なパラメーターを変更します。

警告

Container Catalog は、コンテナーイメージに完全なソフトウェアスタックを組み込んでパブリッシュします。更新およびセキュリティー問題の修正を含むコンテナーイメージを Container Catalog がリリースする際には、既存のカスタムコンテナーには、それらの更新は **含まれない** ので、カタログからのイメージを使用して再ビルドする必要があります。

第3章 コンテナーベースのオーバークラウドのデプロイおよび更新

本章では、コンテナーベースのオーバークラウドを作成し、最新の状態に維持する方法について説明します。

3.1. オーバークラウドのデプロイ

最小設定のオーバークラウドをデプロイする方法を、以下の手順で説明します。デプロイの結果、2 つのノードを持つ基本的なオーバークラウド (1 つのコントローラーノードおよび 1 つのコンピュートノード) が得られます。

手順

1. **stackrc** ファイルを取得します。

```
$ source ~/stackrc
```

2. **deploy** コマンドを実行し、オーバークラウドイメージの場所を定義したファイル (通常は **overcloud_images.yaml**) を含めます。

```
(undercloud) $ openstack overcloud deploy --templates \
  -e /home/stack/templates/overcloud_images.yaml \
  --ntp-server pool.ntp.org
```

3. オーバークラウドがデプロイメントを完了するまで待ちます。

3.2. オーバークラウドの更新

コンテナー化されたオーバークラウドの更新に関する情報は、**Red Hat OpenStack Platform の最新状態の維持**を参照してください。

第4章 コンテナー化されたサービスの操作

本章では、コンテナーを管理するコマンドの例および OpenStack Platform コンテナーに関するトラブルシューティング方法について説明します。

4.1. コンテナー化されたサービスの管理

オーバークラウドでは、OpenStack Platform サービスの大半をコンテナー内で実行します。特定の状況では、1 つのホスト上で個別のサービスを制御する必要がある場合があります。本項には、オーバークラウドノード上で、コンテナー化されたサービスを管理するために実行することのできる一般的な **docker** コマンドについて記載します。**docker** を使用したコンテナー管理に関する包括的な情報は、**Getting Started with Containers**の Working with Docker formatted containers を参照してください。

注記

これらのコマンドを実行する前には、オーバークラウドノードにログイン済みであることを確認し、これらのコマンドをアンダークラウドで実行しないようにしてください。

コンテナーとイメージの一覧表示

実行中のコンテナーを一覧表示するには、以下のコマンドを実行します。

```
$ sudo docker ps
```

停止中またはエラーの発生したコンテナーも一覧表示するには、コマンドに **--all** オプションを追加します。

```
$ sudo docker ps --all
```

コンテナーイメージを一覧表示するには、以下のコマンドを実行します。

```
$ sudo docker images
```

コンテナーの属性の確認

コンテナーまたはコンテナーイメージのプロパティーを確認するには、**docker inspect** コマンドを使用します。たとえば、**keystone** コンテナーを確認するには、以下のコマンドを実行します。

```
$ sudo docker inspect keystone
```

基本的なコンテナー操作の管理

コンテナー化されたサービスを再起動するには、**docker restart** コマンドを使用します。たとえば、**keystone** コンテナーを再起動するには、以下のコマンドを実行します。

```
$ sudo docker restart keystone
```

コンテナー化されたサービスを停止するには、**docker stop** コマンドを使用します。たとえば、**keystone** のコンテナーを停止するには、以下のコマンドを実行します。

```
$ sudo docker stop keystone
```

停止されているコンテナー化されたサービスを起動するには、**docker start** コマンドを使用します。たとえば、**keystone** のコンテナーを起動するには、以下のコマンドを実行します。

```
$ sudo docker start keystone
```

注記

コンテナー内のサービス設定ファイルに加えた変更は、コンテナーの再起動後には元に戻ります。これは、コンテナーがノードのローカルファイルシステム上の **/var/lib/config-data/puppet-generated/** にあるファイルに基づいてサービス設定を再生成するためです。たとえば、**keystone** コンテナー内の **/etc/keystone/keystone.conf** を編集してコンテナーを再起動すると、そのコンテナーはノードのローカルシステム上にある **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** を使用して設定を再生成します。再起動前にコンテナー内で加えられた変更は、この設定によって上書きされます。

コンテナーのモニター

コンテナー化されたサービスのログを確認するには、**docker logs** コマンドを使用します。たとえば、**keystone** のログを確認するには、以下のコマンドを実行します。

```
$ sudo docker logs keystone
```

コンテナーへのアクセス

コンテナー化されたサービスのシェルに入るには、**docker exec** コマンドを使用して **/bin/bash** を起動します。たとえば、**keystone** コンテナーのシェルに入るには、以下のコマンドを実行します。

```
$ sudo docker exec -it keystone /bin/bash
```

keystone コンテナーのシェルに root ユーザーとして入るには、以下のコマンドを実行します。

```
$ sudo docker exec --user 0 -it <NAME OR ID> /bin/bash
```

コンテナーから出るには、以下のコマンドを実行します。

```
# exit
```

4.2. コンテナー化されたサービスに関するトラブルシューティング

オーバークラウドのデプロイメント中またはデプロイメント後にコンテナー化されたサービスでエラーが発生した場合には、以下の推奨事項に従って、エラーの根本的な原因を特定してください。

注記

これらのコマンドを実行する前には、オーバークラウドノードにログイン済みであることを確認し、これらのコマンドをアンダークラウドで実行しないようにしてください。

コンテナーログの確認

各コンテナーは、主要プロセスからの標準出力を保持します。この出力はログとして機能し、コンテナー実行時に実際に何が発生したのかを特定するのに役立ちます。たとえば、**keystone** コンテナーのログを確認するには、以下のコマンドを使用します。

```
$ sudo docker logs keystone
```

大半の場合は、このログにコンテナーのエラーの原因が記載されています。

コンテナーの検査

状況によっては、コンテナーに関する情報を検証する必要がある場合があります。たとえば、以下のコマンドを使用して **keystone** コンテナーのデータを確認します。

```
$ sudo docker inspect keystone
```

これにより、ローレベルの設定データが含まれた JSON オブジェクトが提供されます。その出力を **jq** コマンドにパイプで渡して、特定のデータを解析することが可能です。たとえば、**keystone** コンテナーのマウントを確認するには、以下のコマンドを実行します。

```
$ sudo docker inspect keystone | jq .[0].Mounts
```

--format オプションを使用して、データを単一行に解析することもできます。これは、コンテナーデータのセットに対してコマンドを実行する場合に役立ちます。たとえば、**keystone** コンテナーを実行するのに使用するオプションを再生成するには、以下のように **inspect** コマンドに **--format** オプションを指定して実行します。

```
$ sudo docker inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range .Mounts}} -v
{{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}' keystone
```

注記

--format オプションは、Go 構文を使用してクエリーを作成します。

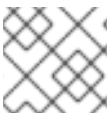
これらのオプションを **docker run** コマンドと共に使用して、トラブルシューティング目的のコンテナーを再度作成します。

```
$ OPTIONS=$( sudo docker inspect --format='{{range .Config.Env}} -e "{{.}}" {{end}} {{range
.Mounts}} -v {{.Source}}:{{.Destination}}{{if .Mode}}:{{.Mode}}{{end}}{{end}} -ti {{.Config.Image}}'
keystone )
$ sudo docker run --rm $OPTIONS /bin/bash
```

コンテナー内でのコマンドの実行

状況によっては、特定の Bash コマンドでコンテナー内の情報を取得する必要がある場合があります。このような場合には、以下の **docker** コマンドを使用して、稼働中のコンテナー内でコマンドを実行します。たとえば、**keystone** コンテナーで次のコマンドを実行します。

```
$ sudo docker exec -ti keystone <COMMAND>
```



注記

-ti オプションを指定すると、コマンドは対話式の擬似ターミナルで実行されます。

<COMMAND> は必要なコマンドに置き換えます。たとえば、各コンテナーには、サービスの接続を確認するためのヘルスチェックスクリプトがあります。**keystone** にヘルスチェックスクリプトを実行するには、以下のコマンドを実行します。

```
$ sudo docker exec -ti keystone /openstack/healthcheck
```

コンテナーのシェルにアクセスするには、コマンドとして **/bin/bash** を使用して **docker exec** を実行します。

```
$ sudo docker exec -ti keystone /bin/bash
```

コンテナーのエクスポート

コンテナーに障害が発生した場合には、ファイルの内容を詳細に調べる必要があります。この場合は、コンテナーの全ファイルシステムを **tar** アーカイブとしてエクスポートすることができます。たとえば、**keystone** コンテナーのファイルシステムをエクスポートするには、以下のコマンドを実行します。

```
$ sudo docker export keystone -o keystone.tar
```

このコマンドにより **keystone.tar** アーカイブが作成されます。これを抽出して、調べることができます。

第5章 SYSTEMD サービスとコンテナー化されたサービスの比較

本章では、コンテナー化されたサービスと Systemd サービスの相違点を示す参考資料を提供します。

5.1. SYSTEMD サービスのコマンドとコンテナー化されたサービスのコマンドの比較

Systemd ベースのコマンドと対応する Docker ベースの等価コマンドの関係を、以下の表に示します。この表は、実施するサービスの操作の種別を把握するのに役立ちます。

機能	Systemd ベース	Docker ベース
全サービスの一覧を表示する	**systemctl list-units -t service**	**docker ps --all**
アクティブなサービスの一覧を表示する	**systemctl list-units -t service --state active**	**docker ps**
サービスのステータスを確認する	**systemctl status openstack-nova-api**	**docker ps --filter "name=nova_api$" --all**
サービスを停止する	**systemctl stop openstack-nova-api**	**docker stop nova_api**
サービスを起動する	**systemctl start openstack-nova-api**	**docker start nova_api**
サービスを再起動する	**systemctl restart openstack-nova-api**	**docker restart nova_api**
サービスの設定を表示する	**systemctl show openstack-nova-api** **systemctl cat openstack-nova-api**	**docker inspect nova_api**
サービスのログを表示する	**journalctl -u openstack-nova-api**	**docker logs nova_api**

5.2. SYSTEMD サービスとコンテナー化されたサービスの比較

Systemd ベースの OpenStack サービスと対応するコンテナーベースの等価サービスを、以下の表に示します。

OpenStack サービス	Systemd サービス	Docker コンテナー

OpenStack サービス	Systemd サービス	Docker コンテナー
aodh	**openstack-aodh-evaluator** **openstack-aodh-listener** **openstack-aodh-notifier** **httpd (openstack-aodh-api)**	**aodh_listener** **aodh_api** **aodh_notifier** **aodh_evaluator**
ceilometer	**openstack-ceilometer-central** **openstack-ceilometer-collector** **openstack-ceilometer-notification** **httpd (openstack-ceilometer-api)**	**ceilometer_agent_notification** **ceilometer_agent_central**
cinder	**openstack-cinder-api** **openstack-cinder-scheduler** **openstack-cinder-volume**	**cinder_scheduler** **cinder_api** **openstack-cinder-volume-docker-0**
glance	**openstack-glance-api** **openstack-glance-registry**	**glance_api**
gnocchi	**openstack-gnocchi-metricd** **openstack-gnocchi-statsd** **httpd (openstack-gnocchi-api)**	**gnocchi_statsd** **gnocchi_api** **gnocchi_metricd**
heat	**openstack-heat-api-cfn** **openstack-heat-api-cloudwatch** **openstack-heat-api** **openstack-heat-engine**	**heat_api_cfn** **heat_engine** **heat_api**
horizon	**httpd (openstack-dashboard)**	**horizon**
keystone	**httpd (openstack-keystone)**	**keystone**

OpenStack サービス	Systemd サービス	Docker コンテナー
neutron	**neutron-dhcp-agent** **neutron-l3-agent** **neutron-metadata-agent** **neutron-openvswitch-agent** **neutron-server**	**neutron_ovs_agent** **neutron_l3_agent** **neutron_metadata_agent** **neutron_dhcp** **neutron_api**
nova	**openstack-nova-api** **openstack-nova-conductor** **openstack-nova-consoleauth** **openstack-nova-novncproxy** **openstack-nova-scheduler**	**nova_metadata** **nova_api** **nova_conductor** **nova_vnc_proxy** **nova_consoleauth** **nova_api_cron** **nova_scheduler** **nova_placement**
panko		**panko_api**

OpenStack サービス	Systemd サービス	Docker コンテナー
swift	**openstack-swift-account-auditor** **openstack-swift-account-reaper** **openstack-swift-account-replicator** **openstack-swift-account** **openstack-swift-container-auditor** **openstack-swift-container-replicator** **openstack-swift-container-updater** **openstack-swift-container** **openstack-swift-object-auditor** **openstack-swift-object-expirer** **openstack-swift-object-replicator** **openstack-swift-object-updater** **openstack-swift-object** **openstack-swift-proxy**	**swift_proxy** **swift_account_server** **swift_container_auditor** **swift_object_expirer** **swift_object_updater** **swift_container_replicator** **swift_account_auditor** **swift_object_replicator** **swift_container_server** **swift_rsync** **swift_account_reaper** **swift_account_replicator** **swift_object_auditor** **swift_object_server** **swift_container_update**

5.3. SYSTEMD のログの場所とコンテナーベースのログの場所の比較

Systemd ベースの OpenStack ログと対応するコンテナーベースの等価ログを、以下の表に示します。すべてのコンテナーベースのログは物理ホストにマウントされたディレクトリーに保管されるので、物理ホストからログにアクセスすることができます。

OpenStack サービス	Systemd サービスのログ	Docker コンテナーのログ
aodh	**/var/log/aodh/**	**/var/log/containers/aodh/** **/var/log/containers/httpd/aodh-api/**

OpenStack サービス	Systemd サービスのログ	Docker コンテナーのログ
ceilometer	**/var/log/ceilometer/**	**/var/log/containers/ceilometer/**
cinder	**/var/log/cinder/**	**/var/log/containers/cinder/** **/var/log/containers/httpd/cinder-api/**
glance	**/var/log/glance/**	**/var/log/containers/glance/**
gnocchi	**/var/log/gnocchi/**	**/var/log/containers/gnocchi/** **/var/log/containers/httpd/gnocchi-api/**
heat	**/var/log/heat/**	**/var/log/containers/heat/** **/var/log/containers/httpd/heat-api/** **/var/log/containers/httpd/heat-api-cfn/**
horizon	**/var/log/horizon/**	**/var/log/containers/horizon/** **/var/log/containers/httpd/horizon/**
keystone	**/var/log/keystone/**	**/var/log/containers/keystone** **/var/log/containers/httpd/keystone/**
databases	**/var/log/mariadb/** **/var/log/mongodb/** **/var/log/mysqld.log**	**/var/log/containers/mysql/**
neutron	**/var/log/neutron/**	**/var/log/containers/neutron/** **/var/log/containers/httpd/neutron-api/**
nova	**/var/log/nova/**	**/var/log/containers/nova/** **/var/log/containers/httpd/nova-api/** **/var/log/containers/httpd/nova-placement/**

OpenStack サービス	Systemd サービスのログ	Docker コンテナーのログ
panko		**/var/log/containers/panko/** **/var/log/containers/httpd/panko-api/**
rabbitmq	**/var/log/rabbitmq/**	**/var/log/containers/rabbitmq/**
redis	**/var/log/redis/**	**/var/log/containers/redis/**
swift	**/var/log/swift/**	**/var/log/containers/swift/**

5.4. SYSTEMD の設定とコンテナーベースの設定の比較

Systemd ベースの OpenStack 設定と対応するコンテナーベースの等価設定を、以下の表に示します。すべてのコンテナーベースの設定の場所は物理ホストで利用可能で、コンテナーにマウントされます。また、それぞれの該当コンテナー内の設定にマージされます (**kolla** により)。

OpenStack サービス	Systemd サービスの設定	Docker コンテナーの設定
aodh	**/etc/aodh/**	**/var/lib/config-data/puppet-generated/aodh/**
ceilometer	**/etc/ceilometer/**	**/var/lib/config-data/puppet-generated/ceilometer/etc/ceilometer/**
cinder	**/etc/cinder/**	**/var/lib/config-data/puppet-generated/cinder/etc/cinder/**
glance	**/etc/glance/**	**/var/lib/config-data/puppet-generated/glance_api/etc/glance/**
gnocchi	**/etc/gnocchi/**	**/var/lib/config-data/puppet-generated/gnocchi/etc/gnocchi/**
haproxy	**/etc/haproxy/**	**/var/lib/config-data/puppet-generated/haproxy/etc/haproxy/**

OpenStack サービス	Systemd サービスの設定	Docker コンテナーの設定
heat	**/etc/heat/**	**/var/lib/config-data/puppet-generated/heat/etc/heat/** **/var/lib/config-data/puppet-generated/heat_api/etc/heat/** **/var/lib/config-data/puppet-generated/heat_api_cfn/etc/heat/**
horizon	**/etc/openstack-dashboard/**	**/var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/**
keystone	**/etc/keystone/**	**/var/lib/config-data/puppet-generated/keystone/etc/keystone/**
databases	**/etc/my.cnf.d/** **/etc/my.cnf**	**/var/lib/config-data/puppet-generated/mysql/etc/my.cnf.d/**
neutron	**/etc/neutron/**	**/var/lib/config-data/puppet-generated/neutron/etc/neutron/**
nova	**/etc/nova/**	**/var/lib/config-data/puppet-generated/nova/etc/nova/** **/var/lib/config-data/puppet-generated/nova_placement/etc/nova/**
panko		**/var/lib/config-data/puppet-generated/panko/etc/panko**
rabbitmq	**/etc/rabbitmq/**	**/var/lib/config-data/puppet-generated/rabbitmq/etc/rabbitmq/**
redis	**/etc/redis/** **/etc/redis.conf**	**/var/lib/config-data/puppet-generated/redis/etc/redis/** **/var/lib/config-data/puppet-generated/redis/etc/redis.conf**

OpenStack サービス	Systemd サービスの設定	Docker コンテナーの設定
swift	**/etc/swift/**	**/var/lib/config-data/puppet-generated/swift/etc/swift/** **/var/lib/config-data/puppet-generated/swift_ringbuilder/etc/swift/**