Red Hat OpenStack Platform 13

Keeping Red Hat OpenStack Platform Updated

Performing minor updates of Red Hat OpenStack Platform

Last Updated: 2023-04-06

Red Hat OpenStack Platform 13 Keeping Red Hat OpenStack Platform Updated

Performing minor updates of Red Hat OpenStack Platform

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document provides the procedure to update your Red Hat OpenStack Platform 13 (Queens) environment. This document assumes you will update a containerized OpenStack Platform deployment installed on Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. INTRODUCTION

This document provides a workflow to help keep your Red Hat OpenStack Platform 13 environment updated with the latest packages and containers.

This guide provides an upgrade path through the following versions:

Old Overcloud Version	New Overcloud Version
Red Hat OpenStack Platform 13	Red Hat OpenStack Platform 13.z

1.1. HIGH LEVEL WORKFLOW

The following table provides an outline of the steps required for the upgrade process:

Step	Description
Obtaining new container images	Create a new environment file containing the latest container images for OpenStack Platform 13 services.
Updating the undercloud	Update the undercloud to the latest OpenStack Platform 13.z version.
Updating the overcloud	Update the overcloud to the latest OpenStack Platform 13.z version.
Updating the Ceph Storage nodes	Upgrade all Ceph Storage 3 services.
Finalize the upgrade	Run the convergence command to refresh your overcloud stack.

1.2. KNOWN ISSUES THAT MIGHT BLOCK AN UPDATE

Review the following known issues that might affect a successful minor version update.

Minor updates of Red Hat Ceph Storage 3 can cause OSD corruption

Red Hat Ceph Storage 3 relies on docker for containerized deployments that run on EL7. The ceph-ansible fix for BZ#1846830 updates the systemd units that control Ceph containers, making the systemd units require the docker service to be up and running for execution. This requirement is essential to implement a safe update path and avoid service disruption or even data corruption on uncontrolled updates of the docker package.

Updating the ceph-ansible package is not sufficient for the ceph-ansible fix to be effective. You must also update the systemd units of the containers by rerunning the deployment playbook. For information about resolving the issue in your director-driven Ceph Storage deployment, refer to the Red Hat Knowledgebase solution Issue affecting minor updates of Red Had Ceph Storage 3 can cause OSDs corruption.

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

The python **tripleo-client** that is used in the **openstack overcloud update run** command might

time out before the update process can complete. This results in the **openstack overcloud update run** command returning a failure, while the update process continues to run in the background until it completes.

To avoid this failure, edit the value of the **ttl** parameter in the **tripleo-client/plugin.py** file to increase the **tripleo-client** timeout value before you update your overcloud nodes. For more information, see the Red Hat Knowledgebase solution OSP 13 update process appears to fail while the update process runs in the background and completes successfully.

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

If you want to update your environment from a release earlier than RHOSP 13 z10 (19 December 2019 maintenance release), to avoid data plane connectivity loss that is described in bug BZ#1955538, see the Red Hat Knowledgebase solution Stale namespaces on OSP13 can create data plane cut during update.

During ceph upgrade all the OSDs (and other ceph services) went down

If you are using Ceph, see the Red Hat Knowledgebase solution During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted to avoid bug BZ#1910842 before you complete the following procedures:

- Updating all Controller nodes
- Updating all HCI Compute nodes
- Updating all Ceph Storage nodes

Octavia and LB issues after z11 upgrade

During an update, the load-balancing service (Octavia) containers would continuously restart due to a missing file named **/var/lib/config-data/puppet-generated/octavia/etc/octavia/conf.d/common/post-deploy.conf**. This file was introduced during the Red Hat OpenStack Platform 13 lifecycle to configure octavia services after the Amphora deployment. This file is currently generated during the **openstack overcloud update converge** step of an update. To work around this issue, you must continue with the update. The octavia containers start normally after you run the **openstack overcloud update converge** command. The Red Hat OpenStack Platform engineering team is currently investigating a resolution to this issue.

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

If you use the load-balancing service (octavia) and want to update from a release earlier than RHOSP 13 z13 (8 October 2020 maintenance release), to avoid bug BZ#1927169, you must run the database migrations that upgrade the load-balancing service in the correct order. You must update the bootstrap Controller node, before you can update the rest of the control plane.

1. To identify your current maintenance release, run the following command:

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

2. On the undercloud node, to identify the bootstrap Controller node, run the following command and replace **<any_controller_node_IP_address>** with the IP address of any of the Controller nodes in your deployment:

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

3. On the undercloud node, run the **openstack overcloud update run** command to update the bootstrap Controller node:

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

Minor update to 13z16 failed with "Unable to find constraint"

When you restart an update of a Red Hat OpenStack Platform 13z16 overcloud node, you might experience an error **Unable to find constraint**. This error occurs because of a discrepancy in RabbitMQ version during the update. To ensure that the new RabbitMQ version can start, you must clear any pacemaker bans that might exist in the overcloud.
For more information about this issue, see the Red Hat Knowledgebase solution Cannot restart Update of the OSP13z16 controllers.

Cannot stop ceph-mon on controllers: No such container: ceph-mon controller-2

If you use Red Hat Ceph Storage version 3.3 z5 or earlier and update the docker package to docker-1.13.1-209, the RHOSP 13 update fails. The RHOSP 13 update does not stop the ceph-mon container before the docker package updates. This results in an orphan ceph-mon process, which blocks the new ceph-mon container from starting.
For more information about this issue, see the Red Hat Knowledgebase solution Updating Red Hat OpenStack Platform 13.z12 and earlier with Ceph Storage might fail during controller update.

1.3. TROUBLESHOOTING

- If the update process takes longer than expected, then it might time out with the error: **socket is already closed**. This can arise because the undercloud's authentication token is set to expire after a set duration. For more information, see Recommendations for Large Deployments.

CHAPTER 2. UPDATING YOUR CONTAINER IMAGE SOURCE

This chapter provides information on how to update your registry source with new overcloud container images for Red Hat OpenStack Platform.

Considerations before updating the container images source

If you want to change your container image source from one registry type to another, you must update the namespace and prefix in your current container image environment files and the run the **openstack overcloud deploy** command to change the namespace before you complete any the following tasks:

- Section 2.4, “Using the Red Hat registry as a remote registry source”
- Section 2.5, “Using the undercloud as a local registry”
- Section 2.6, “Using a Satellite server as a registry”

2.1. REGISTRY METHODS

Red Hat OpenStack Platform supports the following registry types:

Remote Registry

The overcloud pulls container images directly from **registry.redhat.io**. This method is the easiest for generating the initial configuration. However, each overcloud node pulls each image directly from the Red Hat Container Catalog, which can cause network congestion and slower deployment. In addition, all overcloud nodes require internet access to the Red Hat Container Catalog.

Local Registry

The undercloud uses the **docker-distribution** service to act as a registry. This allows the director to synchronize the images from **registry.redhat.io** and push them to the **docker-distribution** registry. When creating the overcloud, the overcloud pulls the container images from the undercloud’s **docker-distribution** registry. This method allows you to store a registry internally, which can speed up the deployment and decrease network congestion. However, the undercloud only acts as a basic registry and provides limited life cycle management for container images.

NOTE

The **docker-distribution** service acts separately from **docker**. **docker** is used to pull and push images to the **docker-distribution** registry and does not serve the images to the overcloud. The overcloud pulls the images from the **docker-distribution** registry.

Satellite Server

Manage the complete application life cycle of your container images and publish them through a Red Hat Satellite 6 server. The overcloud pulls the images from the Satellite server. This method provides an enterprise grade solution to store, manage, and deploy Red Hat OpenStack Platform containers.

Select a method from the list and continue configuring your registry details.

NOTE

When building for a multi-architecture cloud, the local registry option is not supported.

2.2. CONTAINER IMAGE PREPARATION COMMAND USAGE

This section provides an overview on how to use the **openstack overcloud container image prepare** command, including conceptual information on the command's various options.

Generating a Container Image Environment File for the Overcloud

One of the main uses of the **openstack overcloud container image prepare** command is to create an environment file that contains a list of images the overcloud uses. You include this file with your overcloud deployment commands, such as **openstack overcloud deploy**. The **openstack overcloud container image prepare** command uses the following options for this function:

--output-env-file

Defines the resulting environment file name.

The following snippet is an example of this file's contents:

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

The environment file also contains the **DockerInsecureRegistryAddress** parameter set to the IP address and port of the undercloud registry. This parameter configures overcloud nodes to access images from the undercloud registry without SSL/TLS certification.

Generating a Container Image List for Import Methods

If you aim to import the OpenStack Platform container images to a different registry source, you can generate a list of images. The syntax of list is primarily used to import container images to the container registry on the undercloud, but you can modify the format of this list to suit other import methods, such as Red Hat Satellite 6.

The **openstack overcloud container image prepare** command uses the following options for this function:

--output-images-file

Defines the resulting file name for the import list.

The following is an example of this file's contents:

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

Setting the Namespace for Container Images

Both the **--output-env-file** and **--output-images-file** options require a namespace to generate the resulting image locations. The **openstack overcloud container image prepare** command uses the following options to set the source location of the container images to pull:

--namespace

Defines the namespace for the container images. This is usually a hostname or IP address with a directory.

--prefix

Defines the prefix to add before the image names.

As a result, the director generates the image names using the following format:

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

Setting Container Image Tags

Use the **--tag** and **--tag-from-label** options together to set the tag for each container images.

--tag

Sets the specific tag for all images from the source. If you only use this option, director pulls all container images using this tag. However, if you use this option in combination with **--tag-from-label**, director uses the **--tag** as a source image to identify a specific version tag based on labels. The **--tag** option is set to **latest** by default.

--tag-from-label

Use the value of specified container image labels to discover and pull the versioned tag for every image. Director inspects each container image tagged with the value that you set for **--tag**, then uses the container image labels to construct a new tag, which director pulls from the registry. For example, if you set **--tag-from-label {version}-{release}**, director uses the **version** and **release** labels to construct a new tag. For one container, **version** might be set to **13.0** and **release** might be set to **34**, which results in the tag **13.0-34**.

IMPORTANT

The Red Hat Container Registry uses a specific version format to tag all Red Hat OpenStack Platform container images. This version format is **{version}-{release}**, which each container image stores as labels in the container metadata. This version format helps facilitate updates from one **{release}** to the next. For this reason, you must always use the **--tag-from-label {version}-{release}** when running the **openstack overcloud container image prepare** command. Do not only use **--tag** on its own to to pull container images. For example, using **--tag latest** by itself causes problems when performing updates because director requires a change in tag to update a container image.

2.3. CONTAINER IMAGES FOR ADDITIONAL SERVICES

The director only prepares container images for core OpenStack Platform Services. Some additional features use services that require additional container images. You enable these services with environment files. The **openstack overcloud container image prepare** command uses the following option to include environment files and their respective container images:

-e

Include environment files to enable additional container images.

The following table provides a sample list of additional services that use container images and their respective environment file locations within the **/usr/share/openstack-tripleo-heat-templates** directory.

Service	Environment File
Ceph Storage	**environments/ceph-ansible/ceph-ansible.yaml**
Collectd	**environments/services-docker/collectd.yaml**

Service	Environment File
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** NOTE: See OpenStack Shared File System (manila)for more information.
Open Virtual Network (OVN)	**environments/services-docker/neutron-ovn-dvr-ha.yaml**
Sensu	**environments/services-docker/sensu-client.yaml**

The next few sections provide examples of including additional services.

Ceph Storage

If deploying a Red Hat Ceph Storage cluster with your overcloud, you need to include the **/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml** environment file. This file enables the composable containerized services in your overcloud and the director needs to know these services are enabled to prepare their images.

In addition to this environment file, you also need to define the Ceph Storage container location, which is different from the OpenStack Platform services. Use the **--set** option to set the following parameters specific to Ceph Storage:

--set ceph_namespace

Defines the namespace for the Ceph Storage container image. This functions similar to the **--namespace** option.

--set ceph_image

Defines the name of the Ceph Storage container image. Usually,this is **rhceph-3-rhel7**.

--set ceph_tag

Defines the tag to use for the Ceph Storage container image. This functions similar to the **--tag** option. When **--tag-from-label** is specified, the versioned tag is discovered starting from this tag.

The following snippet is an example that includes Ceph Storage in your container image files:

```
$ 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)

If deploying OpenStack Bare Metal (ironic) in your overcloud, you need to include the **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/ironic.yaml** environment file so the director can prepare the images. The following snippet is an example on how to include this environment file:

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

OpenStack Data Processing (sahara)

If deploying OpenStack Data Processing (sahara) in your overcloud, you need to include the **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/sahara.yaml** environment file so the director can prepare the images. The following snippet is an example on how to include this environment file:

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

OpenStack Neutron SR-IOV

If deploying OpenStack Neutron SR-IOV in your overcloud, include the **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/neutron-sriov.yaml** environment file so the director can prepare the images. The default Controller and Compute roles do not support the SR-IOV service, so you must also use the **-r** option to include a custom roles file that contains SR-IOV services. The following snippet is an example on how to include this environment file:

```
$ 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)

If deploying OpenStack Load Balancing-as-a-Service in your overcloud, include the **/usr/share/openstack-tripleo-heat-templates/environments/services-docker/octavia.yaml** environment file so the director can prepare the images. The following snippet is an example on how to include this environment file:

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

OpenStack Shared File System (manila)

Using the format **manila-{backend-name}-config.yaml**, you can choose a supported back end to deploy the Shared File System with that back end. Shared File System service containers can be prepared by including any of the following environment files:

```
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
```

For more information about customizing and deploying environment files, see the following resources:

- Deploying the updated environment in *CephFS via NFS Back End Guide for the Shared File System Service*
- Deploy the Shared File System Service with NetApp Back Ends in *NetApp Back End Guide for the Shared File System Service*
- Deploy the Shared File System Service with a CephFS Back End in *CephFS Back End Guide for the Shared File System Service*

2.4. USING THE RED HAT REGISTRY AS A REMOTE REGISTRY SOURCE

Red Hat hosts the overcloud container images on **registry.redhat.io**. Pulling the images from a remote registry is the simplest method because the registry is already configured and all you require is the URL and namespace of the image that you want to pull. However, during overcloud creation, the overcloud nodes all pull images from the remote repository, which can congest your external connection. As a result, this method is not recommended for production environments. For production environments, use one of the following methods instead:

- Setup a local registry
- Host the images on Red Hat Satellite 6

Procedure

1. To pull the images directly from **registry.redhat.io** in your overcloud deployment, an environment file is required to specify the image parameters. Run the following command to generate the container image environment file:

```
(undercloud) $ sudo openstack overcloud container image prepare \
  --namespace=registry.redhat.io/rhosp13 \
  --prefix=openstack- \
```

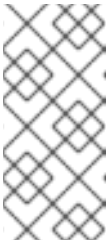
```
  --tag-from-label {version}-{release} \
  --output-env-file=/home/stack/templates/overcloud_images.yaml
```

- Use the **-e** option to include any environment files for optional services.
- Use the **-r** option to include a custom roles file.
- If using Ceph Storage, include the additional parameters to define the Ceph Storage container image location: **--set ceph_namespace**, **--set ceph_image**, **--set ceph_tag**.

2. Modify the **overcloud_images.yaml** file and include the following parameters to authenticate with **registry.redhat.io** during deployment:

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

- Replace **<USERNAME>** and **<PASSWORD>** with your credentials for **registry.redhat.io**. The **overcloud_images.yaml** file contains the image locations on the undercloud. Include this file with your deployment.



NOTE

Before you run the **openstack overcloud deploy** command, you must log in to the remote registry:

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

The registry configuration is ready.

2.5. USING THE UNDERCLOUD AS A LOCAL REGISTRY

You can configure a local registry on the undercloud to store overcloud container images.

You can use director to pull each image from the **registry.redhat.io** and push each image to the **docker-distribution** registry that runs on the undercloud. When you use director to create the overcloud, during the overcloud creation process, the nodes pull the relevant images from the undercloud **docker-distribution** registry.

This keeps network traffic for container images within your internal network, which does not congest your external network connection and can speed the deployment process.

Procedure

1. Find the address of the local undercloud registry. The address uses the following pattern:

```
<REGISTRY_IP_ADDRESS>:8787
```

Use the IP address of your undercloud, which you previously set with the **local_ip** parameter in your **undercloud.conf** file. For the commands below, the address is assumed to be **192.168.24.1:8787**.

2. Log in to **registry.redhat.io**:

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

3. Create a template to upload the images to the local registry, and the environment file to refer to those images:

```
(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
```

   - Use the **-e** option to include any environment files for optional services.
   - Use the **-r** option to include a custom roles file.
   - If using Ceph Storage, include the additional parameters to define the Ceph Storage container image location: **--set ceph_namespace**, **--set ceph_image**, **--set ceph_tag**.

4. Verify that the following two files have been created:

   - **local_registry_images.yaml**, which contains container image information from the remote source. Use this file to pull the images from the Red Hat Container Registry (**registry.redhat.io**) to the undercloud.
   - **overcloud_images.yaml**, which contains the eventual image locations on the undercloud. You include this file with your deployment.

5. Pull the container images from the remote registry and push them to the undercloud registry:

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

   Pulling the required images might take some time depending on the speed of your network and your undercloud disk.

NOTE

The container images consume approximately 10 GB of disk space.

6. The images are now stored on the undercloud's **docker-distribution** registry. To view the list of images on the undercloud's **docker-distribution** registry, run the following command:

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

NOTE

The **_catalog** resource by itself displays only 100 images. To display more images, use the **?n=<interger>** query string with the **_catalog** resource to display a larger number of images:

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

To view a list of tags for a specific image, use the **skopeo** command:

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

To verify a tagged image, use the **skopeo** command:

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

The registry configuration is ready.

2.6. USING A SATELLITE SERVER AS A REGISTRY

Red Hat Satellite 6 offers registry synchronization capabilities. This provides a method to pull multiple images into a Satellite server and manage them as part of an application life cycle. The Satellite also acts as a registry for other container-enabled systems to use. For more details information on managing container images, see "Managing Container Images" in the *Red Hat Satellite 6 Content Management Guide*.

The examples in this procedure use the **hammer** command line tool for Red Hat Satellite 6 and an example organization called **ACME**. Substitute this organization for your own Satellite 6 organization.

Procedure

1. Create a template to pull images to the local registry:

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

- Use the **-e** option to include any environment files for optional services.
- Use the **-r** option to include a custom roles file.
- If using Ceph Storage, include the additional parameters to define the Ceph Storage container image location: **--set ceph_namespace**, **--set ceph_image**, **--set ceph_tag**.

NOTE

This version of the **openstack overcloud container image prepare** command targets the registry on the **registry.redhat.io** to generate an image list. It uses different values than the **openstack overcloud container image prepare** command used in a later step.

2. This creates a file called **satellite_images** with your container image information. You will use this file to synchronize container images to your Satellite 6 server.

3. Remove the YAML-specific information from the **satellite_images** file and convert it into a flat file containing only the list of images. The following **sed** commands accomplish this:

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

   This provides a list of images that you pull into the Satellite server.

4. Copy the **satellite_images_names** file to a system that contains the Satellite 6 **hammer** tool. Alternatively, use the instructions in the *Hammer CLI Guide* to install the **hammer** tool to the undercloud.

5. Run the following **hammer** command to create a new product (**OSP13 Containers**) to your Satellite organization:

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

   This custom product will contain our images.

6. Add the base container image to the product:

```
$ 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. Add the overcloud container images from the **satellite_images** file.

```
$ 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. Synchronize the container images:

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

Wait for the Satellite server to complete synchronization.

NOTE

Depending on your configuration, **hammer** might ask for your Satellite server username and password. You can configure **hammer** to automatically login using a configuration file. See the "Authentication" section in the *Hammer CLI Guide*.

9. If your Satellite 6 server uses content views, create a new content view version to incorporate the images.

10. Check the tags available for the **base** image:

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

This displays tags for the OpenStack Platform container images.

11. Return to the undercloud and generate an environment file for the images on your Satellite server. The following is an example command for generating the environment file:

```
(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
```

NOTE

This version of the **openstack overcloud container image prepare** command targets the Satellite server. It uses different values than the **openstack overcloud container image prepare** command used in a previous step.

When running this command, include the following data:

- **--namespace** - The URL and port of the registry on the Satellite server. The registry port on Red Hat Satellite is 5000. For example, **--namespace=satellite6.example.com:5000**.

NOTE

If you are using Red Hat Satellite version 6.10, you do not need to specify a port. The default port of **443** is used. For more information, see "How can we adapt RHOSP13 deployment to Red Hat Satellite 6.10?".

- **--prefix=** - The prefix is based on a Satellite 6 convention for labels, which uses lower case characters and substitutes spaces for underscores. The prefix differs depending on whether you use content views:

- If you use content views, the structure is **[org]-[environment]-[content view]-[product]-**. For example: **acme-production-myosp13-osp13_containers-**.
- If you do not use content views, the structure is **[org]-[product]-**. For example: **acme-osp13_containers-**.

- **--tag-from-label {version}-{release}** - Identifies the latest tag for each image.
- **-e** - Include any environment files for optional services.
- **-r** - Include a custom roles file.
- **--set ceph_namespace**, **--set ceph_image**, **--set ceph_tag** - If using Ceph Storage, include the additional parameters to define the Ceph Storage container image location. Note that **ceph_image** now includes a Satellite-specific prefix. This prefix is the same value as the **--prefix** option. For example:

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

This ensures the overcloud uses the Ceph container image using the Satellite naming convention.

12. The **overcloud_images.yaml** file contains the image locations on the Satellite server. Include this file with your deployment.

The registry configuration is ready.

2.7. NEXT STEPS

You now have a new **overcloud_images.yaml** environment file that contains a list of your container image sources. Include this file with all future upgrade and deployment operations.

You can now prepare the overcloud for the update.

CHAPTER 3. UPGRADING THE UNDERCLOUD

This process upgrades the undercloud and its overcloud images to **Red Hat OpenStack Platform 13**.

3.1. PERFORMING A MINOR UPDATE OF AN UNDERCLOUD

The director provides commands to update the packages on the undercloud node. This allows you to perform a minor update within the current version of your OpenStack Platform environment.

Procedure

1. Log into the director as the **stack** user.
2. Update the **python-tripleoclient** package and its dependencies to ensure you have the latest scripts for the minor version update:

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

3. The director uses the **openstack undercloud upgrade** command to update the Undercloud environment. Run the command:

```
$ openstack undercloud upgrade
```

4. Wait until the undercloud upgrade process completes.
5. Reboot the undercloud to update the operating system's kernel and other system packages:

```
$ sudo reboot
```

6. Wait until the node boots.

3.2. UPDATING THE OVERCLOUD IMAGES

You need to replace your current overcloud images with new versions. The new images ensure the director can introspect and provision your nodes using the latest version of OpenStack Platform software.

Prerequisites

- You have updated the undercloud to the latest version.

Procedure

1. Remove any existing images from the **images** directory on the **stack** user's home (**/home/stack/images**):

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

2. Extract the archives:

```
$ 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. On the undercloud node, source the undercloud credentials:

```
$ source ~/stackrc
```

4. Import the latest images into the director:

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

5. Configure your nodes to use the new images:

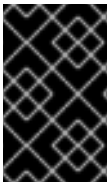
```
$ openstack overcloud node configure $(openstack baremetal node list -c UUID -f value)
```

6. Verify the existence of the new images:

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

IMPORTANT

When deploying overcloud nodes, ensure the overcloud image version corresponds to the respective heat template version. For example, only use the OpenStack Platform 13 images with the OpenStack Platform 13 heat templates.



IMPORTANT

The new **overcloud-full** image replaces the old **overcloud-full** image. If you made changes to the old image, you must repeat the changes in the new image, especially if you want to deploy new nodes in the future.

3.3. UNDERCLOUD POST-UPGRADE NOTES

- If using a local set of core templates in your **stack** users home directory, ensure you update the templates using the recommended workflow in "Using Customized Core Heat Templates" . You must update the local copy before upgrading the overcloud.

3.4. NEXT STEPS

The undercloud upgrade is complete. You can now prepare the overcloud for the upgrade.

CHAPTER 4. UPDATING THE OVERCLOUD

This process updates the overcloud.

Prerequisites

- You have updated the undercloud to the latest version.

4.1. SPEEDING UP THE OVERCLOUD UPDATE

To speed up the overcloud update process, you can configure the **DockerPuppetProcessCount** heat parameter, archive deleted database entries, and download the required packages on the overcloud nodes before you perform an update.

For more information about speeding up the update process for large OpenStack deployments, see the Red Hat Knowledgebase article Openstack Director Node Performance Tuning for large deployments .

Procedure

1. Log in to the undercloud as the **stack** user.
2. Source the **stackrc** file:

```
$ source ~/stackrc
```

3. To increase the number of concurrent processes that **container-puppet** uses to generate configuration files, you must configure the **DockerPuppetProcessCount** parameter.
   a. Create an environment file called **updates-environment.yaml** in your **templates** directory:

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

   b. Edit the file and add the following content:

```
parameter_defaults:
  DockerPuppetProcessCount: 8
```

   c. Use the **-e** option to include this environment file when you run the **openstack overcloud update prepare**, **openstack overcloud ceph-upgrade run**, and **openstack overcloud update converge** commands.
4. On a Controller node, archive your deleted database entries:
   a. From the overcloud, list all instances of the Controller nodes:

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

   b. Log on to a Controller node that is running the **nova_api_cron** container:

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

   - Replace **<controller name or IP>** with the IP address of a Controller node.

c. Archive deleted database entries:

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

5. To download all packages that are required for an update on all the overcloud nodes, complete the following steps:

a. Create a static inventory file of your overcloud:

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

b. Create the following Ansible playbook:

```
$ 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. Run the **yum-download-only.yaml** Ansible playbook:

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

4.2. CONSIDERATION FOR CUSTOM ROLES

Check the following values in your roles file if your deployment includes custom roles:

- Compare your custom roles file with the latest files in the **/usr/share/openstack-tripleo-heat-templates/roles** directory. Add any new parameters from the **RoleParametersDefault** sections for relevant roles for your environment to the equivalent roles in your custom roles file.

- If you use Data Plane Development Kit (DPDK) and are upgrading from 13.4 or lower, ensure that the roles that contain OVS-DPDK services also contain the following mandatory parameters:

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

4.3. RUNNING THE OVERCLOUD UPDATE PREPARATION

The update requires running **openstack overcloud update prepare** command, which performs the following tasks:

- Updates the overcloud plan to OpenStack Platform 13
- Prepares the nodes for the update

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the update preparation command:

```
$ 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>
  ...
```

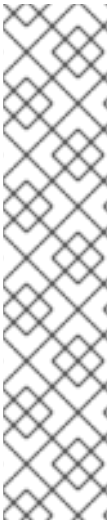
Include the following options relevant to your environment:

- Custom configuration environment files (**-e**).
- The environment file with your new container image locations (**-e**). Note that the update command might display a warning about using the **--container-registry-file**. You can ignore this warning as this option is deprecated in favor of using **-e** for the container image environment file.
- If you use your own custom roles, include your custom roles (**roles_data**) file (**-r**).
- If you use custom networks, include your composable network (**network_data**) file (**-n**).
- If you deploy a high availability cluster, include the **--ntp-server** option in the update preparation command, or include the **NtpServer** parameter and value in your environment file.
- If the name of your overcloud stack is different to the default name **overcloud**, include the **--stack** option in the update preparation command and replace **<STACK_NAME>** with the name of your stack.

3. Wait until the update preparation completes.

4.4. UPDATING THE CEPH STORAGE CLUSTER

This process updates the Ceph Storage cluster. The process involves running the **openstack overcloud ceph-upgrade run** command to perform an update to a Red Hat Ceph Storage 3 cluster.



NOTE

The following combinations of Ansible with **ceph-ansible** are supported:

- **ansible-2.6** with **ceph-ansible-3.2**
- **ansible-2.4** with **ceph-ansible-3.1**

If your environment has **ansible-2.6** with **ceph-ansible-3.1**, update **ceph-ansible** to the newest version:

```
# 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
```

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the Ceph Storage update command. For example:

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

Include the following options relevant to your environment:

- Custom configuration environment files (**-e**)
- The environment file with your container image locations (**-e**). Note that the update command might display a warning about using the **--container-registry-file**. You can ignore this warning, as this option is deprecated in favor of using **-e** for the container image environment file.
- If applicable, your custom roles (**roles_data**) file (**--roles-file**)
- If applicable, your composable network (**network_data**) file (**--networks-file**)

3. Wait until the Ceph Storage node update completes.

NOTE

If the Heat stack times out during the procedure, see the Red Hat Knowledgebase article During sequential update of Ceph nodes **openstack overcloud ceph-upgrade run** appears to time out.

4.5. UPDATING ALL CONTROLLER NODES

To update the Controller nodes to the latest Red Hat OpenStack Platform (RHOSP) 13 version, include the **--nodes Controller** option in the **openstack overcloud update run** command. The **--nodes Controller** option restricts the update operations to the Controller nodes only.

Warning

If you use Ceph, see the Red Hat Knowledgebase solution During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted before you update the Controller nodes in order to avoid bug BZ#1910842.

Prerequisites

- If you use the load-balancing service (octavia) and want to update from a release earlier than RHOSP 13 z13 (8 October 2020 maintenance release), to avoid bug BZ#1927169, you must run the database migrations that upgrade the load-balancing service in the correct order. You must update the bootstrap Controller node, before you can update the rest of the control plane.

  1. To identify your current maintenance release, run the following command:

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

  2. On the undercloud node, to identify the bootstrap Controller node, run the following command and replace **<any_controller_node_IP_address>** with the IP address of any of the Controller nodes in your deployment:

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

  3. On the undercloud node, run the **openstack overcloud update run** command to update the bootstrap Controller node:

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

Procedure

1. Source the **stackrc** file:

   ```
   $ source ~/stackrc
   ```

2. Run the update command:

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

3. Wait until the Controller node update completes.

4.6. UPDATING ALL COMPUTE NODES

This process updates all Compute nodes to the latest OpenStack Platform 13 version. The process involves running the **openstack overcloud update run** command and including the **--nodes Compute** option to restrict operations to the Compute nodes only.

Parallelization considerations

When you update a large number of Compute nodes, to improve performance, you can run the **openstack overcloud update run** command with the **--nodes Compute** option in parallel on

batches of 20 nodes. For example, if you have 80 Compute nodes in your deployment, you can run the following commands to update the Compute nodes in parallel:

```
$ 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 &
```

The **'Compute[0:19]'**, **'Compute[20:39]'**, **'Compute[40:59]'**, and **'Compute[60:79]'** way of partitioning the nodes space is random and you don't have control over the order in which the nodes are updated in each batch.

To update specific Compute nodes, list the nodes that you want to update in a batch separated by a comma:

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

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the update command:

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

3. Wait until the Compute node update completes.

4.7. UPDATING ALL HCI COMPUTE NODES

This process updates the Hyperconverged Infrastructure (HCI) Compute nodes. The process involves running the **openstack overcloud update run** command and including the **--nodes ComputeHCI** option to restrict operations to the HCI nodes only.

Warning

If you are using Ceph, see the Red Hat Knowledgebase solution During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted before you follow this section in order to avoid the following bug BZ#1910842.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the update command:

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

3. Wait until the node update completes.

4.8. UPDATING ALL CEPH STORAGE NODES

This process updates the Ceph Storage nodes. The process involves running the **openstack overcloud update run** command and including the **--nodes CephStorage** option to restrict operations to the Ceph Storage nodes only.

Warning

If you are using Ceph, see the Red Hat Knowledgebase solution During minor update of OSP13/RHCS3 to latest packages Ceph services go offline and need to be manually restarted before you follow this section in order to avoid the following bug BZ#1910842.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Update group nodes.
   To update all nodes in a group:

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

   To update a single node in a group:

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

NOTE

Ensure that you update all nodes if you choose to update nodes individually.

The index of the first node in a group is zero (0). For example, to update the first node in a group named **CephStorage**, the command is:

openstack overcloud update run --nodes CephStorage[0]

3. Wait until the node update completes.

4.9. FINALIZING THE UPDATE

The update requires a final step to update the overcloud stack. This ensures that the stack resource structure aligns with a regular deployment of Red Hat OpenStack Platform 13 and you can perform standard **openstack overcloud deploy** functions in the future.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the update finalization command:

```
$ 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>
...
```

Include the following options relevant to your environment:

- Custom configuration environment files (**-e**)
- The environment file with your new container image locations (**-e**). Note that the update command might display a warning about using the **--container-registry-file**. You can ignore this warning as this option is deprecated in favor of using **-e** for the container image environment file.
- If applicable, your custom roles (**roles_data**) file (**--roles-file**)
- If applicable, your composable network (**network_data**) file (**--networks-file**)
- If the name of your overcloud stack is different than the default name **overcloud**, you must include the **--stack** option in the update preparation command and replace **<STACK_NAME>** with the name of your overcloud stack.

3. Wait until the update finalization completes.

CHAPTER 5. REBOOTING THE OVERCLOUD

After a minor Red Hat OpenStack version update, reboot your overcloud. The reboot refreshes the nodes with any associated kernel, system-level, and container component updates. These updates may provide performance and security benefits.

Plan downtime to perform the following reboot procedures.

5.1. REBOOTING CONTROLLER AND COMPOSABLE NODES

The following procedure reboots controller nodes and standalone nodes based on composable roles. This excludes Compute nodes and Ceph Storage nodes.

Procedure

1. Log in to the node that you want to reboot.
2. Optional: If the node uses Pacemaker resources, stop the cluster:

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

3. Reboot the node:

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

4. Wait until the node boots.
5. Check the services. For example:
   a. If the node uses Pacemaker services, check that the node has rejoined the cluster:

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

   b. If the node uses Systemd services, check that all services are enabled:

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

   c. Repeat these steps for all Controller and composable nodes.

5.2. REBOOTING A CEPH STORAGE (OSD) CLUSTER

The following procedure reboots a cluster of Ceph Storage (OSD) nodes.

Procedure

1. Log in to a Ceph MON or Controller node and disable Ceph Storage cluster rebalancing temporarily:

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

2. Select the first Ceph Storage node to reboot and log into it.

3. Reboot the node:

```
$ sudo reboot
```

4. Wait until the node boots.

5. Log in to a Ceph MON or Controller node and check the cluster status:

```
$ sudo ceph -s
```

   Check that the **pgmap** reports all **pgs** as normal (**active+clean**).

6. Log out of the Ceph MON or Controller node, reboot the next Ceph Storage node, and check its status. Repeat this process until you have rebooted all Ceph storage nodes.

7. When complete, log into a Ceph MON or Controller node and enable cluster rebalancing again:

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

8. Perform a final status check to verify the cluster reports **HEALTH_OK**:

```
$ sudo ceph status
```

5.3. REBOOTING COMPUTE NODES

Rebooting a Compute node involves the following workflow:

- Select a Compute node to reboot and disable it so that it does not provision new instances.
- Migrate the instances to another Compute node to minimise instance downtime.
- Reboot the empty Compute node and enable it.

Procedure

1. Log in to the undercloud as the **stack** user.

2. To identify the Compute node that you intend to reboot, list all Compute nodes:

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

3. From the overcloud, select a Compute Node and disable it:

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

4. List all instances on the Compute node:

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

5. Migrate your instances. For more information on migration strategies, see Migrating virtual machines between Compute nodes.

6. Log into the Compute Node and reboot it:

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

7. Wait until the node boots.

8. Enable the Compute node:

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

9. Verify that the Compute node is enabled:

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

5.4. REBOOTING COMPUTE HCI NODES

The following procedure reboots Compute hyperconverged infrastructure (HCI) nodes.

Procedure

1. Log in to a Ceph MON or Controller node and disable Ceph Storage cluster rebalancing temporarily:

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

2. Log in to the undercloud as the **stack** user.

3. List all Compute nodes and their UUIDs:

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

Identify the UUID of the Compute node you aim to reboot.

4. From the undercloud, select a Compute node and disable it:

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

5. List all instances on the Compute node:

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

6. Use one of the following commands to migrate your instances:

   a. Migrate the instance to a specific host of your choice:

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

b. Let **nova-scheduler** automatically select the target host:

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

c. Live migrate all instances at once:

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

NOTE

The **nova** command might cause some deprecation warnings, which are safe to ignore.

7. Wait until the migration completes.
8. Confirm that the migration was successful:

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

9. Continue migrating instances until none remain on the chosen Compute node.
10. Log in to a Ceph MON or a Controller node and check the cluster status:

```
$ sudo ceph -s
```

Check that the **pgmap** reports all **pgs** as normal (**active+clean**).

11. Reboot the Compute HCI node:

```
$ sudo reboot
```

12. Wait until the node boots.
13. Enable the Compute node again:

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

14. Verify that the Compute node is enabled:

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

15. Log out of the node, reboot the next node, and check its status. Repeat this process until you have rebooted all Ceph storage nodes.
16. When complete, log in to a Ceph MON or Controller node and enable cluster rebalancing again:

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

17. Perform a final status check to verify the cluster reports **HEALTH_OK**:

```
$ sudo ceph status
```

CHAPTER 6. PERFORMING POST-UPDATE TASKS

After a minor Red Hat OpenStack version update, you must perform post-update related tasks to ensure that your environment is fully supported and ready for future operations.

6.1. CONSIDERATIONS FOR OCTAVIA DEPLOYMENTS

If your deployment uses Octavia services, you must update the running load balancing amphora instance with a new image.

To update an amphora image, you must fail over the load balancer and then wait for the load balancer to regain an active state. When the load balancer is active again, it runs the new image.

For more information, see Updating running Load-balancing service instances in the *Using Octavia for Load Balancing-as-a-Service* guide.