Red Hat OpenStack Platform 13

Upgrading Red Hat OpenStack Platform

Upgrading a Red Hat OpenStack Platform environment

Red Hat OpenStack Platform 13 Upgrading Red Hat OpenStack Platform

Upgrading a Red Hat OpenStack Platform environment

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document lays out the different methods through which users can upgrade from Red Hat OpenStack Platform 12 (Pike) to 13 (Queens). These methods assume that you will be upgrading to and from an OpenStack deployment installed on Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. INTRODUCTION

This document provides a workflow to help upgrade your Red Hat OpenStack Platform environment to the latest major version and keep it updated with minor releases of that version.

This guide provides an upgrade path through the following versions:

Old Overcloud Version	New Overcloud Version
Red Hat OpenStack Platform 12	Red Hat OpenStack Platform 13

1.1. HIGH LEVEL WORKFLOW

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

Step	Description
Preparing your environment	Perform a backup of the database and configuration of the undercloud and overcloud Controller nodes. Update to the latest minor release. Validate the environment.
Upgrading the undercloud	Upgrade the undercloud from OpenStack Platform 12 to OpenStack Platform 13.
Obtaining container images	Create an environment file containing the locations of container images for OpenStack Platform 13 services.
Preparing the overcloud	Perform relevant steps to transition your overcloud configuration files to OpenStack Platform 13.
Upgrading your Controller nodes	Upgrade all Controller nodes simultaneously to OpenStack Platform 13.
Upgrading your Compute nodes	Test the upgrade on selected Compute nodes. If the test succeeds, upgrade all Compute nodes.
Upgrading your Ceph Storage nodes	Upgrade all Ceph Storage nodes. This includes an upgrade to containerized version of Red Hat Ceph Storage 3.
Finalize the upgrade	Run the convergence command to refresh your overcloud stack.

1.2. REPOSITORIES

Both the undercloud and overcloud require access to Red Hat repositories either through the Red Hat Content Delivery Network or through Red Hat Satellite 6. If using a Red Hat Satellite Server,

synchronize the required repositories to your OpenStack Platform environment. Use the following list of CDN channel names as a guide:

Table 1.1. OpenStack Platform Repositories

Name	Repository	Description of Requirement
Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rpms**	Base operating system repository for x86_64 systems.
Red Hat Enterprise Linux 7 Server – Extras (RPMs)	**rhel-7-server-extras-rpms**	Contains Red Hat OpenStack Platform dependencies.
Red Hat Enterprise Linux 7 Server – RH Common (RPMs)	**rhel-7-server-rh-common-rpms**	Contains tools for deploying and configuring Red Hat OpenStack Platform.
Red Hat Satellite Tools 6.3 (for RHEL 7 Server) (RPMs) x86_64	**rhel-7-server-satellite-tools-6.3-rpms**	Tools for managing hosts with Red Hat Satellite Server 6. Note that using later versions of the Satellite Tools repository might cause the undercloud installation to fail.
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	**rhel-ha-for-rhel-7-server-rpms**	High availability tools for Red Hat Enterprise Linux. Used for Controller node high availability.
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-rpms**	Core Red Hat OpenStack Platform repository. Also contains packages for Red Hat OpenStack Platform director.
Red Hat Ceph Storage OSD 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-osd-rpms**	(For Ceph Storage Nodes) Repository for Ceph Storage Object Storage daemon. Installed on Ceph Storage nodes.
Red Hat Ceph Storage MON 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-mon-rpms**	(For Ceph Storage Nodes) Repository for Ceph Storage Monitor daemon. Installed on Controller nodes in OpenStack environments using Ceph Storage nodes.

Name	Repository	Description of Requirement
Red Hat Ceph Storage Tools 3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-3-tools-rpms**	Provides tools for nodes to communicate with the Ceph Storage cluster. Enable this repository for all nodes when you deploy an overcloud with a Ceph Storage cluster or when you integrate your overcloud with an existing Ceph Storage cluster.
Red Hat OpenStack 13 Director Deployment Tools for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-deployment-tools-rpms**	(For Ceph Storage Nodes) Provides a set of deployment tools that are compatible with the current version of Red Hat OpenStack Platform director. Installed on Ceph nodes without an active Red Hat OpenStack Platform subscription.
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	**rhel-7-server-nfv-rpms**	Repository for Real Time KVM (RT-KVM) for NFV. Contains packages to enable the real time kernel. This repository should be enabled for all Compute nodes targeted for RT-KVM. NOTE: You need a separate subscription to a **Red Hat OpenStack Platform for Real Time** SKU before you can access this repository.
Red Hat OpenStack Platform 13 Extended Life Cycle Support for RHEL 7 (RPMs)	**rhel-7-server-openstack-13-els-rpms**	Contains updates for Extended Live Cycle Support which began on June 26, 2021. You need the "Entitlement for OpenStack 13 Platform Extended Life Cycle Support" (MCT3637) for this repository to be available.

NOTE

To configure repositories for your Red Hat OpenStack Platform environment in an offline network, see "Configuring Red Hat OpenStack Platform Director in an Offline Environment" on the Red Hat Customer Portal.

1.3. BEFORE STARTING THE UPGRADE

- Apply any firmware updates to your hardware before performing the upgrade.

- During updates, if the Open vSwitch (OVS) major version changes (for example, 2.9 to 2.11), director renames any user-customized configuration files with an .rpmsave extension, and installs the default OVS configuration.
  If you want to retain your earlier OVS customizations, you must manually reapply your modifications contained in the renamed files (for example, logrotate configurations in /etc/logrotate.d/openvswitch). This two-step update method avoids data plane interruptions that would be triggered by an automatic RPM package update.

CHAPTER 2. PREPARING FOR AN OPENSTACK PLATFORM UPGRADE

This process prepares your OpenStack Platform environment for a full update. This involves the following process:

- Update the undercloud packages and run the upgrade command

- Reboot the undercloud in case a newer kernel or newer system packages are installed

- Update the overcloud using the overcloud upgrade command

- Reboot the overcloud nodes in case a newer kernel or newer system packages are installed

- Perform a validation check on both the undercloud and overcloud

These procedures ensure your OpenStack Platform environment is in the best possible state before proceeding with the upgrade.

2.1. PERFORMING A MINOR UPDATE OF THE CURRENT UNDERCLOUD AND OVERCLOUD

Before upgrading to OpenStack Platform 13, you must update your existing environment to the latest minor version, which is version 12 in this case. To update your existing OpenStack Platform 12 environment, see "Keeping OpenStack Platform Updated".

2.2. 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.

2.3. 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
   ```

2.4. 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
   ```

2.5. VALIDATING THE UNDERCLOUD

The following is a set of steps to check the functionality of your undercloud.

Procedure

1. Source the undercloud access details:

   ```
   $ source ~/stackrc
   ```

2. Check for failed Systemd services:

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

3. Check the undercloud free space:

> (undercloud) $ df -h

Use the "Undercloud Requirements" as a basis to determine if you have adequate free space.

4. If you have NTP installed on the undercloud, check that clocks are synchronized:

> (undercloud) $ sudo ntpstat

5. Check the undercloud network services:

> (undercloud) $ openstack network agent list

All agents should be **Alive** and their state should be **UP**.

6. Check the undercloud compute services:

> (undercloud) $ openstack compute service list

All agents' status should be **enabled** and their state should be **up**

Related Information

- The following solution article shows how to remove deleted stack entries in your OpenStack Orchestration (heat) database: https://access.redhat.com/solutions/2215131

2.6. VALIDATING A CONTAINERIZED OVERCLOUD

The following is a set of steps to check the functionality of your containerized overcloud.

Procedure

1. Source the undercloud access details:

> $ source ~/stackrc

2. Check the status of your bare metal nodes:

> (undercloud) $ openstack baremetal node list

All nodes should have a valid power state (**on**) and maintenance mode should be **false**.

3. Check for failed Systemd services:

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

4. Check for failed containerized services:

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

```
echo "=== $NODE ===" ; ssh heat-admin@$NODE "sudo docker ps -f 'status=dead' -f
'status=restarting'" ; done
```

5. Check the HAProxy connection to all services. Obtain the Control Plane VIP address and authentication details for the **haproxy.stats** service:

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

Use these details in the following cURL request:

```
(undercloud) $ curl -s -u admin:<PASSWORD> "http://<IP ADDRESS>:1993/;csv" | egrep -vi
"(frontend|backend)" | cut -d, -f 1,2,18,37,57 | column -s, -t
```

Replace **<PASSWORD>** and **<IP ADDRESS>** details with the actual details from the **haproxy.stats** service. The resulting list shows the OpenStack Platform services on each node and their connection status.

> **NOTE**
>
> In case the nodes run Redis services, only one node displays an **ON** status for that service. This is because Redis is an active–passive service, which runs only on one node at a time.

6. Check overcloud database replication health:

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

7. Check RabbitMQ cluster health:

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

8. Check Pacemaker resource health:

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

Look for:

- All cluster nodes **online**.

- No resources **stopped** on any cluster nodes.

- No **failed** pacemaker actions.

9. Check the disk space on each overcloud node:

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

10. Check overcloud Ceph Storage cluster health. The following command runs the **ceph** tool on a Controller node to check the cluster:

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

11. Check Ceph Storage OSD for free space. The following command runs the **ceph** tool on a Controller node to check the free space:

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

12. Check that clocks are synchronized on overcloud nodes

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

13. Source the overcloud access details:

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

14. Check the overcloud network services:

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

All agents should be **Alive** and their state should be **UP**.

15. Check the overcloud compute services:

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

All agents' status should be **enabled** and their state should be **up**

16. Check the overcloud volume services:

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

All agents' status should be **enabled** and their state should be **up**.

Related Information

- Review the article "How can I verify my OpenStack environment is deployed with Red Hat recommended configurations?". This article provides some information on how to check your Red Hat OpenStack Platform environment and tune the configuration to Red Hat's recommendations.

CHAPTER 3. UPGRADING THE UNDERCLOUD

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

3.1. UPGRADING THE UNDERCLOUD TO OPENSTACK PLATFORM 13

This procedure upgrades the undercloud toolset and the core Heat template collection to the **OpenStack Platform 13** release.

Procedure

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

2. Disable the current OpenStack Platform repository:

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

3. Set the RHEL version to RHEL 7.9:

   ```
   $ sudo subscription-manager release --set=7.9
   ```

4. Enable the new OpenStack Platform repository:

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

5. Re-enable updates to the overcloud base images:

   ```
   $ sudo yum-config-manager --setopt=exclude= --save
   ```

6. Run **yum** to upgrade the director's main packages:

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

7. Run the following command to upgrade the undercloud:

   ```
   $ openstack undercloud upgrade
   ```

8. Wait until the undercloud upgrade process completes.

9. Reboot the undercloud to update the operating system's kernel and other system packages:

   ```
   $ sudo reboot
   ```

10. Wait until the node boots.

You have upgraded the undercloud to the **OpenStack Platform 13** release.

3.2. UPGRADING 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 upgraded the undercloud to the latest version.

Procedure

1. Source the undercloud access details:

   ```
   $ source ~/stackrc
   ```

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

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

3. 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 ~
   ```

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. COMPARING PREVIOUS TEMPLATE VERSIONS

The upgrade process installs a new set of core Heat templates that correspond to the latest overcloud version. Red Hat OpenStack Platform's repository retains the previous version of the core template collection in the **openstack-tripleo-heat-templates-compat** package. This procedure shows how to

compare these versions so you can identify changes that might affect your overcloud upgrade.

Procedure

1. Install the **openstack-tripleo-heat-templates-compat** package:

   ```
   $ sudo yum install openstack-tripleo-heat-templates-compat
   ```

   This installs the previous templates in the **compat** directory of your Heat template collection (**/usr/share/openstack-tripleo-heat-templates/compat**) and also creates a link to **compat** named after the previous version (**pike**). These templates are backwards compatible with the upgraded director, which means you can use the latest version of the director to install an overcloud of the previous version.

2. Create a temporary copy of the core Heat templates:

   ```
   $ cp -a /usr/share/openstack-tripleo-heat-templates /tmp/osp13
   ```

3. Move the previous version into its own directory:

   ```
   $ mv /tmp/osp13/compat /tmp/osp12
   ```

4. Perform a **diff** on the contents of both directories:

   ```
   $ diff -urN /tmp/osp12 /tmp/osp13
   ```

   This shows the core template changes from one version to the next. These changes provide an idea of what should occur during the overcloud upgrade.

3.4. 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.

3.5. NEXT STEPS

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

CHAPTER 4. CONFIGURING A CONTAINER IMAGE SOURCE

A containerized overcloud requires access to a registry with the required container images. This chapter provides information on how to prepare the registry and your overcloud configuration to use container images for Red Hat OpenStack Platform.

This guide provides several use cases to configure your overcloud to use a registry. Before attempting one of these use cases, it is recommended to familiarize yourself with how to use the image preparation command. See Section 4.2, "Container image preparation command usage" for more information.

4.1. 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.

4.2. 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.

4.3. 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**
Congress	**environments/services-docker/congress.yaml**
Fluentd	**environments/services-docker/fluentd.yaml**
OpenStack Bare Metal (ironic)	**environments/services-docker/ironic.yaml**

Service	Environment File
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*

4.4. 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.

4.5. 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.

4.6. 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.

4.7. NEXT STEPS

You now have an **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 upgrade.

CHAPTER 5. PREPARING FOR THE OVERCLOUD UPGRADE

This process prepares the overcloud for the upgrade process.

Prerequisites

- You have upgraded the undercloud to the latest version.

5.1. PREPARING OVERCLOUD REGISTRATION DETAILS

You need to provide the overcloud with the latest subscription details to ensure the overcloud consumes the latest packages during the upgrade process.

Prerequisites

- A subscription containing the latest OpenStack Platform repositories.

- If using activation keys for registration, create a new activation key including the new OpenStack Platform repositories.

Procedure

1. Edit the environment file containing your registration details. For example:

   ```
   $ vi ~/templates/rhel-registration/environment-rhel-registration.yaml
   ```

2. Edit the following parameter values:

   rhel_reg_repos

   Update to include the new repositories for Red Hat OpenStack Platform 13.

   rhel_reg_activation_key

   Update the activation key to access the Red Hat OpenStack Platform 13 repositories.

   rhel_reg_sat_repo

   If using a newer version of Red Hat Satellite 6, update the repository containing Satellite 6's management tools.

3. Save the environment file.

Related Information

- For more information about registration parameters, see "Registering the Overcloud with an Environment File" in the *Advanced Overcloud Customizations* guide.

5.2. DEPRECATED PARAMETERS

Note that the following parameters are deprecated and have been replaced.

Old Parameter	New Parameter
KeystoneNotificationDriver	**NotificationDriver**

Old Parameter	New Parameter
controllerExtraConfig	ControllerExtraConfig
OvercloudControlFlavor	OvercloudControllerFlavor
controllerImage	ControllerImage
NovaImage	ComputeImage
NovaComputeExtraConfig	ComputeExtraConfig
NovaComputeServerMetadata	ComputeServerMetadata
NovaComputeSchedulerHints	ComputeSchedulerHints **NOTE** If you are using a custom Compute role, in order to use the role-specific **ComputeSchedulerHints**, you need to add the following configuration to your environment to ensure the deprecated **NovaComputeSchedulerHints** parameter is set but undefined: parameter_defaults: NovaComputeSchedulerHints: {} You must add this configuration to use any role-specific **_ROLE_SchedulerHints** parameters when using a custom role.
NovaComputeIPs	ComputeIPs
SwiftStorageServerMetadata	ObjectStorageServerMetadata
SwiftStorageIPs	ObjectStorageIPs
SwiftStorageImage	ObjectStorageImage
OvercloudSwiftStorageFlavor	OvercloudObjectStorageFlavor
NeutronDpdkCoreList	OvsPmdCoreList
NeutronDpdkMemoryChannels	OvsDpdkMemoryChannels
NeutronDpdkSocketMemory	OvsDpdkSocketMemory

Old Parameter	New Parameter
NeutronDpdkDriverType	OvsDpdkDriverType
HostCpusList	OvsDpdkCoreList

For the values of the new parameters, use double quotation marks without nested single quotation marks, as shown in the following examples:

Old Parameter With Value	New Parameter With Value
NeutronDpdkCoreList: "'2,3'"	OvsPmdCoreList: "2,3"
HostCpusList: "'0,1'"	OvsDpdkCoreList: "0,1"

Update these parameters in your custom environment files. The following parameters have been deprecated with no current equivalent.

NeutronL3HA

L3 high availability is enabled in all cases except for configurations with distributed virtual routing (**NeutronEnableDVR**).

CeilometerWorkers

Ceilometer is deprecated in favor of newer components (Gnocchi, Aodh, Panko).

CinderNetappEseriesHostType

All E-series support has been deprecated.

ControllerEnableSwiftStorage

Manipulation of the **ControllerServices** parameter should be used instead.

OpenDaylightPort

Use the EndpointMap to define a default port for OpenDaylight.

OpenDaylightConnectionProtocol

The value of this parameter is now determined based on whether or not you are deploying the Overcloud with TLS.

Run the following **egrep** command in your **/home/stack** directory to identify any environment files that contain deprecated parameters:

```
$ egrep -r -w
'KeystoneNotificationDriver|controllerExtraConfig|OvercloudControlFlavor|controllerImage|NovaImage|N
ovaComputeExtraConfig|NovaComputeServerMetadata|NovaComputeSchedulerHints|NovaComputeIP
s|SwiftStorageServerMetadata|SwiftStorageIPs|SwiftStorageImage|OvercloudSwiftStorageFlavor|Neutr
onDpdkCoreList|NeutronDpdkMemoryChannels|NeutronDpdkSocketMemory|NeutronDpdkDriverType|
HostCpusList|NeutronDpdkCoreList|HostCpusList|NeutronL3HA|CeilometerWorkers|CinderNetappEser
iesHostType|ControllerEnableSwiftStorage|OpenDaylightPort|OpenDaylightConnectionProtocol' *
```

If your OpenStack Platform environment still requires these deprecated parameters, the default **roles_data** file allows their use. However, if you are using a custom **roles_data** file and your overcloud still requires these deprecated parameters, you can allow access to them by editing the **roles_data** file

and adding the following to each role:

Controller Role

```
- name: Controller
  uses_deprecated_params: True
  deprecated_param_extraconfig: 'controllerExtraConfig'
  deprecated_param_flavor: 'OvercloudControlFlavor'
  deprecated_param_image: 'controllerImage'
  ...
```

Compute Role

```
- name: Compute
  uses_deprecated_params: True
  deprecated_param_image: 'NovaImage'
  deprecated_param_extraconfig: 'NovaComputeExtraConfig'
  deprecated_param_metadata: 'NovaComputeServerMetadata'
  deprecated_param_scheduler_hints: 'NovaComputeSchedulerHints'
  deprecated_param_ips: 'NovaComputeIPs'
  deprecated_server_resource_name: 'NovaCompute'
  disable_upgrade_deployment: True
  ...
```

Object Storage Role

```
- name: ObjectStorage
  uses_deprecated_params: True
  deprecated_param_metadata: 'SwiftStorageServerMetadata'
  deprecated_param_ips: 'SwiftStorageIPs'
  deprecated_param_image: 'SwiftStorageImage'
  deprecated_param_flavor: 'OvercloudSwiftStorageFlavor'
  disable_upgrade_deployment: True
  ...
```

5.3. DEPRECATED CLI OPTIONS

Some command line options are outdated or deprecated in favor of using Heat template parameters, which you include in the **parameter_defaults** section on an environment file. The following table maps deprecated options to their Heat template equivalents.

Table 5.1. Mapping deprecated CLI options to Heat template parameters

Option	Description	Heat Template Parameter
--control-scale	The number of Controller nodes to scale out	**ControllerCount**
--compute-scale	The number of Compute nodes to scale out	**ComputeCount**

Option	Description	Heat Template Parameter
--ceph-storage-scale	The number of Ceph Storage nodes to scale out	**CephStorageCount**
--block-storage-scale	The number of Cinder nodes to scale out	**BlockStorageCount**
--swift-storage-scale	The number of Swift nodes to scale out	**ObjectStorageCount**
--control-flavor	The flavor to use for Controller nodes	**OvercloudControllerFlavor**
--compute-flavor	The flavor to use for Compute nodes	**OvercloudComputeFlavor**
--ceph-storage-flavor	The flavor to use for Ceph Storage nodes	**OvercloudCephStorageFlavor**
--block-storage-flavor	The flavor to use for Cinder nodes	**OvercloudBlockStorageFlavor**
--swift-storage-flavor	The flavor to use for Swift storage nodes	**OvercloudSwiftStorageFlavor**
--neutron-flat-networks	Defines the flat networks to configure in neutron plugins. Defaults to "datacentre" to permit external network creation	**NeutronFlatNetworks**
--neutron-physical-bridge	An Open vSwitch bridge to create on each hypervisor. This defaults to "br-ex". Typically, this should not need to be changed	**HypervisorNeutronPhysicalBridge**
--neutron-bridge-mappings	The logical to physical bridge mappings to use. Defaults to mapping the external bridge on hosts (br-ex) to a physical name (datacentre). You would use this for the default floating network	**NeutronBridgeMappings**
--neutron-public-interface	Defines the interface to bridge onto br-ex for network nodes	**NeutronPublicInterface**
--neutron-network-type	The tenant network type for Neutron	**NeutronNetworkType**

Option	Description	Heat Template Parameter
--neutron-tunnel-types	The tunnel types for the Neutron tenant network. To specify multiple values, use a comma separated string	**NeutronTunnelTypes**
--neutron-tunnel-id-ranges	Ranges of GRE tunnel IDs to make available for tenant network allocation	**NeutronTunnelIdRanges**
--neutron-vni-ranges	Ranges of VXLAN VNI IDs to make available for tenant network allocation	**NeutronVniRanges**
--neutron-network-vlan-ranges	The Neutron ML2 and Open vSwitch VLAN mapping range to support. Defaults to permitting any VLAN on the 'datacentre' physical network	**NeutronNetworkVLANRanges**
--neutron-mechanism-drivers	The mechanism drivers for the neutron tenant network. Defaults to "openvswitch". To specify multiple values, use a comma-separated string	**NeutronMechanismDrivers**
--neutron-disable-tunneling	Disables tunneling in case you aim to use a VLAN segmented network or flat network with Neutron	No parameter mapping.
--validation-errors-fatal	The overcloud creation process performs a set of pre-deployment checks. This option exits if any fatal errors occur from the pre-deployment checks. It is advisable to use this option as any errors can cause your deployment to fail.	No parameter mapping
--ntp-server	Sets the NTP server to use to synchronize time	**NtpServer**

These parameters have been removed from Red Hat OpenStack Platform. It is recommended to convert your CLI options to Heat parameters and add them to an environment file.

5.4. COMPOSABLE NETWORKS

This version of Red Hat OpenStack Platform introduces a new feature for composable networks. If using a custom **roles_data** file, edit the file to add the composable networks to each role. For example, for Controller nodes:

```
- name: Controller
  networks:
    - External
    - InternalApi
    - Storage
    - StorageMgmt
    - Tenant
```

Check the default **/usr/share/openstack-tripleo-heat-templates/roles_data.yaml** file for further examples of syntax. Also check the example role snippets in **/usr/share/openstack-tripleo-heat-templates/roles**.

The following table provides a mapping of composable networks to custom standalone roles:

Role	Networks Required
Ceph Storage Monitor	**Storage**, **StorageMgmt**
Ceph Storage OSD	**Storage**, **StorageMgmt**
Ceph Storage RadosGW	**Storage**, **StorageMgmt**
Cinder API	**InternalApi**
Compute	**InternalApi**, **Tenant**, **Storage**
Controller	**External**, **InternalApi**, **Storage**, **StorageMgmt**, **Tenant**
Database	**InternalApi**
Glance	**InternalApi**
Heat	**InternalApi**
Horizon	**InternalApi**
Ironic	None required. Uses the Provisioning/Control Plane network for API.
Keystone	**InternalApi**
Load Balancer	**External**, **InternalApi**, **Storage**, **StorageMgmt**, **Tenant**
Manila	**InternalApi**
Message Bus	**InternalApi**

Role	Networks Required
Networker	**InternalApi**, **Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**External**, **InternalApi**, **Tenant**
Redis	**InternalApi**
Sahara	**InternalApi**
Swift API	**Storage**
Swift Storage	**StorageMgmt**
Telemetry	**InternalApi**

> **IMPORTANT**
>
> In previous versions, the ***NetName** parameters (e.g. **InternalApiNetName**) changed the names of the default networks. This is no longer supported. Use a custom composable network file. For more information, see "Using Composable Networks" in the *Advanced Overcloud Customization* guide.

5.5. INCREASING THE RESTART DELAY FOR LARGE CEPH CLUSTERS

During the upgrade, each Ceph monitor and OSD is stopped sequentially. The migration does not continue until the same service that was stopped is successfully restarted. Ansible waits 15 seconds (the delay) and checks 5 times for the service to start (the retries). If the service does not restart, the migration stops so the operator can intervene.

Depending on the size of the Ceph cluster, you may need to increase the retry or delay values. The exact names of these parameters and their defaults are as follows:

```
health_mon_check_retries: 5
health_mon_check_delay: 15
health_osd_check_retries: 5
health_osd_check_delay: 15
```

You can update the default values for these parameters. For example, to make the cluster check 30 times and wait 40 seconds between each check for the Ceph OSDs, and check 20 times and wait 10 seconds between each check for the Ceph MONs, pass the following parameters in a **yaml** file with **-e** using the **openstack overcloud deploy** command:

```
parameter_defaults:
  CephAnsibleExtraConfig:
    health_osd_check_delay: 40
```

```
health_osd_check_retries: 30
health_mon_check_retries: 10
health_mon_check_delay: 20
```

5.6. PREPARING TO UPGRADE CEPH

OpenStack Platform 13 introduced Red Hat Ceph Storage 3, which requires the CephMgr service. The default templates in OpenStack Platform 13 provide roles that contain the CephMgr service. However, if you used the composable roles feature to customize roles, and the overcloud deployed Ceph, then you must update the role that includes the CephMon service to also include the CephMgr service.

See Comparing Previous Template Versions for an example of how to compare the templates.

> **NOTE**
>
> If you deployed a hyperconverged overcloud by customizing your roles, you must complete these instructions.

Procedure

The **openstack overcloud deploy** command can include a roles file, such as **roles_file.yaml**, in the overcloud deployment.

If any **roles** file includes **OS::TripleO::Services::CephMon**, add the CephMgr service to the roles file:

```
OS::TripleO::Services::CephMon
OS::TripleO::Services::CephMgr
```

> **NOTE**
>
> You must add **OS::TripleO::Services::CephMgr** before the Ceph upgrade described in Upgrading all Ceph Storage nodes .

5.7. CREATING NODE-SPECIFIC CEPH LAYOUT

To create a node-specific Ceph layout using **ceph-ansible**, do the following:

```
parameter_defaults:
  NodeDataLookup: |
    {"6E310C04-6186-45DB-B643-54332E76FAD1": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "1E222ACE-56B9-4F14-9DB8-929734C7CAAF": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]},
     "C6841D59-9E3E-4875-BC43-FB5F49227CE7": {"devices": ["/dev/vdb"], "dedicated_devices":
["/dev/vdc"]}
    }
```

5.8. CHECKING CUSTOM PUPPET PARAMETERS

If you use the **ExtraConfig** interfaces for customizations of Puppet parameters, Puppet might report duplicate declaration errors during the upgrade. This is due to changes in the interfaces provided by the puppet modules themselves.

This procedure shows how to check for any custom **ExtraConfig** hieradata parameters in your environment files.

Procedure

1. Select an environment file and the check if it has an **ExtraConfig** parameter:

   ```
   $ grep ExtraConfig ~/templates/custom-config.yaml
   ```

2. If the results show an **ExtraConfig** parameter for any role (e.g. **ControllerExtraConfig**) in the chosen file, check the full parameter structure in that file.

3. If the parameter contains any puppet Hierdata with a **SECTION/parameter** syntax followed by a **value**, it might have been been replaced with a parameter with an actual Puppet class. For example:

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

4. Check the director's Puppet modules to see if the parameter now exists within a Puppet class. For example:

   ```
   $ grep dnsmasq_local_resolv
   ```

   If so, change to the new interface.

5. The following are examples to demonstrate the change in syntax:

   - Example 1:

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

     Changes to:

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

   - Example 2:

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

     Changes to:

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

5.9. CONVERTING NETWORK INTERFACE TEMPLATES TO THE NEW STRUCTURE

Previously the network interface structure used a **OS::Heat::StructuredConfig** resource to configure interfaces:

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

The templates now use a **OS::Heat::SoftwareConfig** resource for configuration:

```
resources:
  OsNetConfigImpl:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config:
        str_replace:
          template:
            get_file: /usr/share/openstack-tripleo-heat-templates/network/scripts/run-os-net-config.sh
          params:
            $network_config:
              network_config:
                [NETWORK INTERFACE CONFIGURATION HERE]
```

This configuration takes the interface configuration stored in the **$network_config** variable and injects it as a part of the **run-os-net-config.sh** script.

> **WARNING**
>
> It is mandatory to update your network interface template to use this new structure and check your network interface templates still conforms to the syntax. Not doing so can cause failure during the fast forward upgrade process.

The director's Heat template collection contains a script to help convert your templates to this new format. This script is located in **/usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py**. For an example of usage:

```
$ /usr/share/openstack-tripleo-heat-templates/tools/yaml-nic-config-2-script.py \
   --script-dir /usr/share/openstack-tripleo-heat-templates/network/scripts \
   [NIC TEMPLATE] [NIC TEMPLATE] ...
```

> **IMPORTANT**
>
> Ensure your templates does not contain any commented lines when using this script. This can cause errors when parsing the old template structure.

For more information, see "Network isolation".

> **NOTE**
>
> If you enabled High Availability for Compute Instances (Instance HA) in Red Hat OpenStack Platform 12 or earlier and you want to perform an upgrade to version 13 or later, you must manually disable Instance Ha first. For instructions, see Disabling Instance HA from previous versions.

5.10. PREPARING BLOCK STORAGE SERVICE TO RECEIVE CUSTOM CONFIGURATION FILES

When upgrading to the containerized environment, use the **CinderVolumeOptVolumes** parameter to add docker volume mounts. This enables custom configuration files on the host to be made available to the cinder-volume service when it's running in a container.

For example:

```
parameter_defaults:
  CinderVolumeOptVolumes:
    /etc/cinder/nfs_shares1:/etc/cinder/nfs_shares1
    /etc/cinder/nfs_shares2:/etc/cinder/nfs_shares2
```

5.11. PREPARING FOR PRE-PROVISIONED NODES UPGRADE

Pre-provisioned nodes are nodes created outside of the director's management. An overcloud using pre-provisioned nodes requires some additional steps prior to upgrading.

Prerequisites

- The overcloud uses pre-provisioned nodes.

Procedure

1. Run the following commands to save a list of node IP addresses in the **OVERCLOUD_HOSTS** environment variable:

   ```
   $ source ~/stackrc
   $ export OVERCLOUD_HOSTS=$(openstack server list -f value -c Networks | cut -d "=" -f 2 |
   tr '\n' ' ')
   ```

2. Run the following script:

```
$ /usr/share/openstack-tripleo-heat-templates/deployed-server/scripts/enable-ssh-admin.sh
```

3. Proceed with the upgrade.

   - When using the **openstack overcloud upgrade run** command with pre-provisioned nodes, include the **--ssh-user tripleo-admin** parameter.

   - When upgrading Compute or Object Storage nodes, use the following:

     a. Use the **-U** option with the **upgrade-non-controller.sh** script and specify the **stack** user. This is because the default user for pre-provisioned nodes is **stack** and not **heat-admin**.

     b. Use the node's IP address with the **--upgrade** option. This is because the nodes are not managed with the director's Compute (nova) and Bare Metal (ironic) services and do not have a node name.
        For example:

        ```
        $ upgrade-non-controller.sh -U stack --upgrade 192.168.24.100
        ```

Related Information

- For more information on pre-provisioned nodes, see "Configuring a Basic Overcloud using Pre-Provisioned Nodes" in the *Director Installation and Usage* guide.

5.12. NEXT STEPS

The overcloud preparation stage is complete. You can now perform an upgrade of the overcloud to 13 using the steps in Chapter 6, *Upgrading the Overcloud*.

CHAPTER 6. UPGRADING THE OVERCLOUD

This process upgrades the overcloud.

Prerequisites

- You have upgraded the undercloud to the latest version.

- You have prepared your custom environment files to accommodate the changes in the upgrade.

6.1. RUNNING THE OVERCLOUD UPGRADE PREPARATION

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

- Updates the overcloud plan to OpenStack Platform 13

- Prepares the nodes for the upgrade

Procedure

1. Source the **stackrc** file:

   ```
   $ source ~/stackrc
   ```

2. Run the upgrade preparation command:

   ```
   $ openstack overcloud upgrade prepare \
       --templates \
       -e /home/stack/templates/overcloud_images.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 upgrade 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 using **--roles-file**.

   - If applicable, your composable network (**network_data**) file using **--networks-file**.

3. Wait until the upgrade preparation completes.

6.2. UPGRADING CONTROLLER AND CUSTOM ROLE NODES

Use the following process to upgrade all the Controller nodes, split Controller services, and other custom nodes to OpenStack Platform 13. The process involves running the **openstack overcloud upgrade run** command and including the **--nodes** option to restrict operations to only the selected nodes:

```
$ openstack overcloud upgrade run --nodes [ROLE]
```

Substitute **[ROLE]** for the name of a role or a comma-separated list of roles.

If your overcloud uses monolithic Controller nodes, run this command against the **Controller** role.

If your overcloud uses split Controller services, use the following guide to upgrade the node role in the following order:

- All roles that use Pacemaker. For example: **ControllerOpenStack**, **Database**, **Messaging**, and **Telemetry**.

- **Networker** nodes
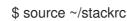
- Any other custom roles

Do not upgrade the following nodes yet:

- Compute nodes of any type such as DPDK based or Hyper-Converged Infratructure (HCI) Compute nodes

- **CephStorage** nodes

You will upgrade these nodes at a later stage.

Procedure

1. Source the **stackrc** file:

   ```
   $ source ~/stackrc
   ```

2. If you use monolithic Controller nodes, run the upgrade command against the **Controller** role:

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

   - If you use a custom stack name, pass the name with the **--stack** option.

3. If you use Controller services split across multiple roles:

   a. Run the upgrade command for roles with Pacemaker services:

      ```
      $ openstack overcloud upgrade run --nodes ControllerOpenStack
      $ openstack overcloud upgrade run --nodes Database
      $ openstack overcloud upgrade run --nodes Messaging
      $ openstack overcloud upgrade run --nodes Telemetry
      ```

      - If you use a custom stack name, pass the name with the **--stack** option.

   b. Run the upgrade command for the **Networker** role:

      ```
      $ openstack overcloud upgrade run --nodes Networker
      ```

      - If you use a custom stack name, pass the name with the **--stack** option.

c. Run the upgrade command for any remaining custom roles, except for **Compute** or **CephStorage** roles:

```
$ openstack overcloud upgrade run --nodes ObjectStorage
```

- If you use a custom stack name, pass the name with the **--stack** option.

6.3. UPGRADING ALL COMPUTE NODES

Important

- If you are using a hyperconverged deployment, see Section 6.5, "Upgrading hyperconverged nodes" for how to upgrade.

- If you are using a mixed hyperconverged deployment, see Section 6.6, "Upgrading mixed hyperconverged nodes" for how to upgrade.

This process upgrades all remaining Compute nodes to OpenStack Platform 13. The process involves running the **openstack overcloud upgrade run** command and including the **--nodes Compute** option to restrict operations to the Compute nodes only.

Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade command:

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

- If you are using a custom stack name, pass the name with the **--stack** option.

- If you are using custom Compute roles, ensure that you include the role names with the **--nodes** option.

3. Wait until the Compute node upgrade completes.

6.4. UPGRADING ALL CEPH STORAGE NODES

Important

- If you are using a hyperconverged deployment, see Section 6.5, "Upgrading hyperconverged nodes" for how to upgrade.

- If you are using a mixed hyperconverged deployment, see Section 6.6, "Upgrading mixed hyperconverged nodes" for how to upgrade.

This process upgrades the Ceph Storage nodes. The process involves:

- Running the **openstack overcloud upgrade run** command and including the **--nodes CephStorage** option to restrict operations to the Ceph Storage nodes only.

- Running the **openstack overcloud ceph-upgrade run** command to perform an upgrade to a containerized Red Hat Ceph Storage 3 cluster.

Procedure

1. Source the **stackrc** file:

   ```
   $ source ~/stackrc
   ```

2. Run the upgrade command:

   ```
   $ openstack overcloud upgrade run --nodes CephStorage
   ```

   - If using a custom stack name, pass the name with the **--stack** option.

3. Wait until the node upgrade completes.

4. Run the Ceph Storage upgrade command. For example:

   ```
   $ openstack overcloud ceph-upgrade run \
       --templates \
       -e /home/stack/templates/overcloud_images.yaml \
       -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
   ansible.yaml \
       -e /home/stack/templates/ceph-customization.yaml \
       -e <ENVIRONMENT FILE>
   ```

   Include the following options relevant to your environment:

   - Custom configuration environment files (**-e**). For example:

     - The environment file with your container image locations (**overcloud_images.yaml**). Note that the upgrade 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.

     - The relevant environment files for your Ceph Storage nodes.

     - Any additional environment files relevant to your environment.

   - If using a custom stack name, pass the name with the **--stack** option.

   - If applicable, your custom roles (**roles_data**) file using **--roles-file**.

   - If applicable, your composable network (**network_data**) file using **--networks-file**.

5. Wait until the Ceph Storage node upgrade completes.

6.5. UPGRADING HYPERCONVERGED NODES

If you are using only hyperconverged nodes from the ComputeHCI role, and are not using dedicated compute nodes or dedicated Ceph nodes, complete the following procedure to upgrade your nodes:

Procedure

1. Source the stackrc file:

   ```
   $ source ~/stackrc
   ```

2. Run the upgrade command:

   ```
   $ openstack overcloud upgrade run --roles ComputeHCI
   ```

   If you are using a custom stack name, pass the name to the upgrade command with the **--stack** option.

3. Run the Ceph Storage upgrade command. For example:

   ```
   $ openstack overcloud ceph-upgrade run \
       --templates \
       -e /home/stack/templates/overcloud_images.yaml \
       -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml \
       -e /home/stack/templates/ceph-customization.yaml \
       -e <ENVIRONMENT FILE>
   ```
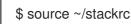
   Include the following options relevant to your environment:

   - Custom configuration environment files (**-e**). For example:

     - The environment file with your container image locations (**overcloud_images.yaml**). Note that the upgrade 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.

     - The relevant environment files for your Ceph Storage nodes.

   - If using a custom stack name, pass the name with the **--stack** option.

   - If applicable, your custom roles (**roles_data**) file using **--roles-file**.

   - If applicable, your composable network (**network_data**) file using **--networks-file**.

4. Wait until the Ceph Storage node upgrade completes.

6.6. UPGRADING MIXED HYPERCONVERGED NODES

If you are using dedicated compute nodes or dedicated ceph nodes in addition to hyperconverged nodes like the ComputeHCI role, complete the following procedure to upgrade your nodes:

Procedure

1. Source the stackrc file:

   ```
   $ source ~/stackrc
   ```

2. Run the upgrade command for the Compute node:

```
$ openstack overcloud upgrade run --roles Compute
If using a custom stack name, pass the name with the --stack option.
```

3. Wait until the node upgrade completes.

4. Run the upgrade command for the ComputeHCI node:

```
$ openstack overcloud upgrade run --roles ComputeHCI
If using a custom stack name, pass the name with the --stack option.
```

5. Wait until the node upgrade completes.

6. Run the upgrade command for the Ceph Storage node:

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

7. Wait until the Ceph Storage node upgrade completes.

8. Run the Ceph Storage upgrade command. For example:

```
$ openstack overcloud ceph-upgrade run \
    --templates \
    -e /home/stack/templates/overcloud_images.yaml \
    -e /usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-
ansible.yaml \
    -e /home/stack/templates/ceph-customization.yaml \
    -e <ENVIRONMENT FILE>
```

Include the following options relevant to your environment:

- Custom configuration environment files (**-e**). For example:

  - The environment file with your container image locations (**overcloud_images.yaml**). Note that the upgrade 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.

  - The relevant environment files for your Ceph Storage nodes.

  - Any additional environment files relevant to your environment.

- If using a custom stack name, pass the name with the **--stack** option.

- If applicable, your custom roles (**roles_data**) file using **--roles-file**.

- If applicable, your composable network (**network_data**) file using **--networks-file**.

9. Wait until the Ceph Storage node upgrade completes.

6.7. FINALIZING THE UPGRADE

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

Procedure

1. Source the **stackrc** file:

   ```
   $ source ~/stackrc
   ```

2. Run the upgrade finalization command:

   ```
   $ openstack overcloud upgrade converge \
       --templates \
       -e <ENVIRONMENT FILE>
   ```

   Include the following options relevant to your environment:

   - Custom configuration environment files (**-e**).

   - If using a custom stack name, pass the name with the **--stack** option.

   - If applicable, your custom roles (**roles_data**) file using **--roles-file**.

   - If applicable, your composable network (**network_data**) file using **--networks-file**.

3. Wait until the upgrade finalization completes.

CHAPTER 7. EXECUTING POST UPGRADE STEPS

This process implements final steps after completing the main upgrade process.

Prerequisites

- You have completed the overcloud upgrade to the latest major release.

7.1. GENERAL CONSIDERATIONS AFTER AN OVERCLOUD UPGRADE

The following items are general considerations after an overcloud upgrade:

- If necessary, review the resulting configuration files on the overcloud nodes. The upgraded packages might have installed **.rpmnew** files appropriate to the upgraded version of each service.

- The Compute nodes might report a failure with **neutron-openvswitch-agent**. If this occurs, log into each Compute node and restart the service. For example:

  ```
  $ sudo docker restart neutron_ovs_agent
  ```

- In some circumstances, the **corosync** service might fail to start on IPv6 environments after rebooting Controller nodes. This is due to Corosync starting before the Controller node configures the static IPv6 addresses. In these situations, restart Corosync manually on the Controller nodes:

  ```
  $ sudo systemctl restart corosync
  ```