



# **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

Copyright © 2018 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

## 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

<b>CHAPTER 1. INTRODUCTION</b>	<b>3</b>
1.1. HIGH LEVEL WORKFLOW	3
1.2. REPOSITORIES	3
<b>CHAPTER 2. PREPARING FOR AN OPENSTACK PLATFORM UPGRADE</b>	<b>6</b>
2.1. BACKING UP THE UNDERCLOUD	6
2.2. BACKING UP CONTAINERIZED OVERCLOUD CONTROL PLANE SERVICES	7
2.3. PERFORMING A MINOR UPDATE OF AN UNDERCLOUD	9
2.4. PERFORMING A MINOR UPDATE OF A CONTAINERIZED OVERCLOUD	10
2.5. REBOOTING CONTROLLER AND COMPOSABLE NODES	11
2.6. REBOOTING A CEPH STORAGE (OSD) CLUSTER	12
2.7. REBOOTING COMPUTE NODES	13
2.8. VALIDATING THE UNDERCLOUD	14
2.9. VALIDATING A CONTAINERIZED OVERCLOUD	15
<b>CHAPTER 3. UPGRADING THE UNDERCLOUD</b>	<b>18</b>
3.1. UPGRADING THE UNDERCLOUD TO OPENSTACK PLATFORM 13	18
3.2. UPGRADING THE OVERCLOUD IMAGES	18
3.3. COMPARING PREVIOUS TEMPLATE VERSIONS	19
3.4. NEXT STEPS	20
<b>CHAPTER 4. CONFIGURING A CONTAINER IMAGE SOURCE</b>	<b>21</b>
4.1. REGISTRY METHODS	21
4.2. CONTAINER IMAGE PREPARATION COMMAND USAGE	21
4.3. CONTAINER IMAGES FOR ADDITIONAL SERVICES	23
4.4. USING THE RED HAT REGISTRY AS A REMOTE REGISTRY SOURCE	25
4.5. USING THE UNDERCLOUD AS A LOCAL REGISTRY	26
4.6. USING A SATELLITE SERVER AS A REGISTRY	27
4.7. NEXT STEPS	30
<b>CHAPTER 5. PREPARING FOR THE OVERCLOUD UPGRADE</b>	<b>31</b>
5.1. PREPARING OVERCLOUD REGISTRATION DETAILS	31
5.2. DEPRECATED PARAMETERS	31
5.3. DEPRECATED CLI OPTIONS	33
5.4. COMPOSABLE NETWORKS	35
5.5. CHECKING CUSTOM PUPPET PARAMETERS	37
5.6. CONVERTING NETWORK INTERFACE TEMPLATES TO THE NEW STRUCTURE	38
5.7. NEXT STEPS	39
<b>CHAPTER 6. UPGRADING THE OVERCLOUD</b>	<b>40</b>
6.1. RUNNING THE OVERCLOUD UPGRADE PREPARATION	40
6.2. UPGRADING ALL CONTROLLER NODES	40
6.3. UPGRADING ALL COMPUTE NODES	41
6.4. UPGRADING ALL CEPH STORAGE NODES	41
6.5. FINALIZING THE UPGRADE	42
<b>CHAPTER 7. EXECUTING POST UPGRADE STEPS</b>	<b>44</b>
7.1. GENERAL CONSIDERATIONS AFTER AN OVERCLOUD UPGRADE	44



# 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)	<b>rhel-7-server-rpms</b>	Base operating system repository for x86_64 systems.
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	<b>rhel-7-server-extras-rpms</b>	Contains Red Hat OpenStack Platform dependencies.
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	<b>rhel-7-server-rh-common-rpms</b>	Contains tools for deploying and configuring Red Hat OpenStack Platform.
Red Hat Satellite Tools for RHEL 7 Server RPMs x86_64	<b>rhel-7-server-satellite-tools-6.3-rpms</b>	Tools for managing hosts with Red Hat Satellite 6.
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	<b>rhel-ha-for-rhel-7-server-rpms</b>	High availability tools for Red Hat Enterprise Linux. Used for Controller node high availability.
Red Hat OpenStack Platform 13 for RHEL 7 (RPMs)	<b>rhel-7-server-openstack-13-rpms</b>	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)	<b>rhel-7-server-rhceph-3-osd-rpms</b>	(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)	<b>rhel-7-server-rhceph-3-mon-rpms</b>	(For Ceph Storage Nodes) Repository for Ceph Storage Monitor daemon. Installed on Controller nodes in OpenStack environments using Ceph Storage nodes.
Red Hat Ceph Storage Tools 3 for Red Hat Enterprise Linux 7 Server (RPMs)	<b>rhel-7-server-rhceph-3-tools-rpms</b>	Provides tools for nodes to communicate with the Ceph Storage cluster. This repository should be enabled for all nodes when deploying an overcloud with a Ceph Storage cluster.



Name	Repository	Description of Requirement
Enterprise Linux for Real Time for NFV (RHEL 7 Server) (RPMs)	<b>rhel-7-server-nfv-rpms</b>	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 will need a separate subscription to a <b>Red Hat OpenStack Platform for Real Time</b> SKU before you can access this repository.



## 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.

## CHAPTER 2. PREPARING FOR AN OPENSTACK PLATFORM UPGRADE

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

- Backup both the undercloud and overcloud
- 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. BACKING UP THE UNDERCLOUD

A full undercloud backup includes the following databases and files:

- All MariaDB databases on the undercloud node
- MariaDB configuration file on the undercloud (so that you can accurately restore databases)
- The configuration data: **/etc**
- Log data: **/var/log**
- Image data: **/var/lib/glance**
- Certificate generation data if using SSL: **/var/lib/certmonger**
- Any container image data: **/var/lib/docker** and **/var/lib/registry**
- All swift data: **/srv/node**
- All data in the stack user home directory: **/home/stack**



#### NOTE

Confirm that you have sufficient disk space available on the undercloud before performing the backup process. Expect the archive file to be at least 3.5 GB, if not larger.

#### Procedure

1. Log into the undercloud as the **root** user.
2. Create a **backup** directory, and change the user ownership of the directory to the **stack** user:

```
[root@director ~]# mkdir /backup
```

```
[root@director ~]# chown stack: /backup
```

3. From the **backup** directory, back up the database:

```
[root@director ~]# cd /backup
[root@director ~]# mysqldump --opt --all-databases >
/root/undercloud-all-databases.sql
```

4. Archive the database backup and the configuration files:

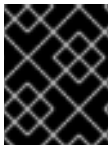
```
[root@director ~]# tar --xattrs --ignore-failed-read -cf \
  undercloud-backup-`date +%F`.tar \
  /root/undercloud-all-databases.sql \
  /etc \
  /var/log \
  /var/lib/glance \
  /var/lib/certmonger \
  /var/lib/docker \
  /var/lib/registry \
  /srv/node \
  /root \
  /home/stack
```

- The **--ignore-failed-read** option skips any directory that does not apply to your undercloud.
- The **--xattrs** option includes extended attributed, which are required to store metadata for Object Storage (swift).

This creates a file named **undercloud-backup-`<date>`.tar.gz**, where **<date>** is the system date. Copy this **tar** file to a secure location.

## 2.2. BACKING UP CONTAINERIZED OVERCLOUD CONTROL PLANE SERVICES

The following procedure creates a backup of the containerized overcloud databases and configuration. A backup of the overcloud database and services ensures you have a snapshot of a working environment. Having this snapshot helps in case you need to restore the overcloud to its original state in case of an operational failure.



### IMPORTANT

This procedure only includes crucial control plane services. It does not include backups of Compute node workloads, data on Ceph Storage nodes, nor any additional services.

### Procedure

1. Perform the database backup:
  - a. Log into a Controller node. You can access the overcloud from the undercloud:

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

- b. Change to the **root** user:

```
$ sudo -i
```

- c. Create a temporary directory to store the backups:

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

- d. Obtain the database password and store it in the **MYSQldbPASS** environment variable. The password is stored in the **mysql::server::root\_password** variable within the **/etc/puppet/hieradata/service\_configs.json** file. Use the following command to store the password:

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

- e. Backup the database:

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

This dumps a database backup called **/var/tmp/mysql\_backup/openstack\_databases-`<date>`.sql** where **`<date>`** is the system date and time. Copy this database dump to a secure location.

- f. Backup all the users and permissions information:

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

This will dump a database backup called **/var/tmp/mysql\_backup/openstack\_databases\_grants-`<date>`.sql** where **`<date>`** is the system date and time. Copy this database dump to a secure location.

## 2. Backup the OpenStack Telemetry database:

- a. Connect to any controller and get the IP of the MongoDB primary instance:

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

- b. Create the backup:

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

- c. Copy the database dump in **/var/tmp/mongo\_backup/** to a secure location.

## 3. Backup the Redis cluster:

- a. Obtain the Redis endpoint from HAProxy:

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

- b. Obtain the master password for the Redis cluster:

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

- c. Check connectivity to the Redis cluster:

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

- d. Dump the Redis database:

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

This stores the database backup in the default **/var/lib/redis/** directory. Copy this database dump to a secure location.

## 4. Backup the filesystem on each Controller node:

- a. Create a directory for the backup:

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

- b. Run the following
- tar**
- command:

```
# tar --ignore-failed-read --xattrs \
  -zcvf /var/tmp/filesystem_backup/fs_backup-`date +%Y-%m-%d-%H-%M-%S`.tar.gz \
  /var/lib/config-data \
  /var/log/containers \
  /etc/corosync \
  /etc/logrotate.d \
  /etc/openvswitch \
  /var/log/openvswitch \
  /srv/node \
  /home/heat-admin
```

The **--ignore-failed-read** option ignores any missing directories, which is useful if certain services are not used or separated on their own custom roles.

5. Copy the resulting **tar** file to a secure location.

## 2.3. 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.

## 2.4. PERFORMING A MINOR UPDATE OF A CONTAINERIZED OVERCLOUD

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

### Procedure

1. Find the latest tag for the containerized service images:

```
$ openstack overcloud container image tag discover \
  --image registry.access.redhat.com/rhosp12/openstack-base:latest \
  --tag-from-label version-release
```

Make a note of the most recent tag.

2. Create an updated environment file for your container image source. Run using the **openstack overcloud container image prepare** command. For example, to use images from **registry.access.redhat.com**:

```
$ openstack overcloud container image prepare \
  --namespace=registry.access.redhat.com/rhosp12 \
  --prefix=openstack- \
  --tag [TAG] \ 1
  --set ceph_namespace=registry.access.redhat.com/rhceph \
  --set ceph_image=rhceph-2-rhel7 \
  --set ceph_tag=latest \
  --env-file=/home/stack/templates/overcloud_images.yaml \
  -e /home/stack/templates/custom_environment_file.yaml 2
```

1 Replace **[TAG]** with the tag obtained from the previous step.

2

Include all additional environment files with the **-e** parameter. The director checks the custom resources in all included environment files and identifies the container images

For more information about generating this environment file for different source types, see ["Configuring a container image source"](#) in the *Director Installation and Usage* guide.

3. Run the **openstack overcloud update stack** command to update the container image locations in your overcloud:

```
$ openstack overcloud update stack --init-minor-update \
  --container-registry-file
  /home/stack/templates/overcloud_images.yaml
```

The **--init-minor-update** only performs an update of the parameters in the overcloud stack. It does not perform the actual package or container update. Wait until this command completes.

4. Perform a package and container update using the **openstack overcloud update** command. Using the **--nodes** option to upgrade node for each role. For example, the following command updates nodes in the **Controller** role

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

Run this command for each role group in the following order:

- **Controller**
  - **CephStorage**
  - **Compute**
  - **ObjectStorage**
  - Any custom roles such as **Database**, **MessageBus**, **Networker**, and so forth.
5. The update process starts for the chosen role starts. The director uses an Ansible playbook to perform the update and displays the output of each task.
  6. Update the next role group. Repeat until you have updated all nodes.

## 2.5. 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. Select a node to reboot. Log into it and stop the cluster before rebooting:

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

2. Reboot the node:

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

- 
- 3. Wait until the node boots.
- 4. Re-enable the cluster for the node:

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

- 5. Log into the node and check the services. For example:
  - a. If the node uses Pacemaker services, check the node has rejoined the cluster:

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

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

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

## 2.6. REBOOTING A CEPH STORAGE (OSD) CLUSTER

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

### Procedure

- 1. Log into 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 into the 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 node, reboot the next 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
```

## 2.7. REBOOTING COMPUTE NODES

The following procedure reboots Compute nodes. To ensure minimal downtime of instances in your OpenStack Platform environment, this procedure also includes instructions on migrating instances from the chosen Compute node. This 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
- Reboot the empty Compute node and enable it

### Procedure

1. Log into the undercloud as the **stack** user.
2. 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.

3. 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
```

4. List all instances on the Compute node:

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

5. 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.

6. Wait until migration completes.

7. Confirm the migration was successful:

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

8. Continue migrating instances until none remain on the chosen Compute Node.

9. Log into the Compute Node and reboot it:

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

10. Wait until the node boots.

11. Enable the Compute Node again:

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

12. Check whether the Compute node is enabled:

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

## 2.8. 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>

## 2.9. 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)" | awk -F',' '{
print $1" "$2" "$18 }'
```

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

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 into the director as the **stack** user.
2. Disable the current OpenStack Platform repository:

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

3. Enable the new OpenStack Platform repository:

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

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

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

5. Run the following command to upgrade the undercloud:

```
$ openstack undercloud upgrade
```

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

```
$ sudo reboot
```

8. 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. 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. Import the latest images into the director:

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

4. Configure your nodes to use the new images:

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

5. 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.

## 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 (**ocata**). 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/osp12/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. 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.access.redhat.com**. 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.access.redhat.com** 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.access.redhat.com/rhosp13/openstack-aodh-
api:latest
  DockerAodhConfigImage: registry.access.redhat.com/rhosp13/openstack-
aodh-api:latest
  ...
```

## 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.access.redhat.com/rhosp13/openstack-aodh-api:latest
- imagename: registry.access.redhat.com/rhosp13/openstack-aodh-
evaluator:latest
  ...
```

## 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

The **openstack overcloud container image prepare** command uses the **latest** tag for each container image by default. However, you can select a specific tag for an image version using one of the following options:

### **--tag-from-label**

Use the value of the specified container image labels to discover the versioned tag for every image.

### **--tag**

Sets the specific tag for all images. All OpenStack Platform container images use the same tag to provide version synchronicity. When using in combination with **--tag-from-label**, the versioned tag is discovered starting from this tag.

## 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	<b>environments/ceph-ansible/ceph-ansible.yaml</b>
Collectd	<b>environments/services-docker/collectd.yaml</b>
Congress	<b>environments/services-docker/congress.yaml</b>
Fluentd	<b>environments/services-docker/fluentd-client.yaml</b>
OpenStack Bare Metal (ironic)	<b>environments/services-docker/ironic.yaml</b>
OpenStack Data Processing (sahara)	<b>environments/services-docker/sahara.yaml</b>
OpenStack EC2-API	<b>environments/services-docker/ec2-api.yaml</b>
OpenStack Key Manager (barbican)	<b>environments/services-docker/barbican.yaml</b>

Service	Environment File
OpenStack Load Balancing-as-a-Service (octavia)	<b>environments/services-docker/octavia.yaml</b>
OpenStack Shared File System Storage (manila)	<b>environments/services-docker/manila.yaml</b>
Open Virtual Network (OVN)	<b>environments/services-docker/neutron-ovn-dvr-ha.yaml</b>
Sensu	<b>environments/services-docker/sensu-client.yaml</b>

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.access.redhat.com/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 \
...
```

## 4.4. USING THE RED HAT REGISTRY AS A REMOTE REGISTRY SOURCE

Red Hat hosts the overcloud container images on **registry.access.redhat.com**. Pulling the images from a remote registry is the simplest method because the registry is already setup and all you require is the URL and namespace of the image you aim to pull. However, during overcloud creation, the overcloud nodes all pull images from the remote repository, which can congest your external connection. If that is a problem, you can either:

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

### Procedure

1. To pull the images directly from **registry.access.redhat.com** in your overcloud deployment, an environment file is required to specify the image parameters. The following command automatically creates this environment file:

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

- Use the `-e` option to include any environment files for optional services.
  - 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. This creates an **overcloud\_images.yaml** environment file, which contains image locations, on the undercloud. You include this file with your deployment.

## 4.5. USING THE UNDERCLOUD AS A LOCAL REGISTRY

You can configure a local registry on the undercloud to store overcloud container images. This method involves the following:

- The director pulls each image from the **registry.access.redhat.com**.
- The director pushes each images to the **docker-distribution** registry running on the undercloud.
- The director creates the overcloud.
- During the overcloud creation, the nodes pull the relevant images from the undercloud's **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 will use 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. Create a template to upload the the images to the local registry, and the environment file to refer to those images:

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

- Use the **-e** option to include any environment files for optional services.
  - 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**.
3. This creates two files:
    - **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.access.redhat.com**) to the undercloud.
    - **overcloud\_images.yaml**, which contains the eventual image locations on the undercloud. You include this file with your deployment.  
Check that both files exist.

4. Pull the container images from **registry.access.redhat.com** to the undercloud.

```
(undercloud) $ sudo 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.

5. 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 using the following command:

```
(undercloud) $ curl http://192.0.2.5:8787/v2/_catalog | jq
.repositories[]
```

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

```
(undercloud) $ skopeo inspect --tls-verify=false
docker://192.0.2.5:8787/rhosp13/openstack-keystone | jq .RepoTags[]
```

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

```
(undercloud) $ skopeo inspect --tls-verify=false
docker://192.0.2.5:8787/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.
- 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.access.redhat.com** 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.access.redhat.com \  
  --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.access.redhat.com \  
    --docker-upstream-name rhosp13/openstack-base \  
    --name $IMAGENAME
```



```
--product "OSP13 Containers" \
--content-type docker \
--url https://registry.access.redhat.com \
--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 default registry port on Red Hat Satellite is 5000. For example, **--namespace=satellite6.example.com:5000**.

- **--prefix=** - The prefix is based on a Satellite 6 convention. This 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.
- **--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. This creates an **overcloud\_images.yaml** environment file, which contains the image locations on the Satellite server. You 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 with role-specific parameters:

Old Parameter	New Parameter
<b>controllerExtraConfig</b>	<b>ControllerExtraConfig</b>
<b>OvercloudControlFlavor</b>	<b>OvercloudControllerFlavor</b>
<b>controllerImage</b>	<b>ControllerImage</b>
<b>NovaImage</b>	<b>ComputeImage</b>
<b>NovaComputeExtraConfig</b>	<b>ComputeExtraConfig</b>
<b>NovaComputeServerMetadata</b>	<b>ComputeServerMetadata</b>
<b>NovaComputeSchedulerHints</b>	<b>ComputeSchedulerHints</b>
<b>NovaComputeIPs</b>	<b>ComputeIPs</b>
<b>SwiftStorageServerMetadata</b>	<b>ObjectStorageServerMetadata</b>
<b>SwiftStorageIPs</b>	<b>ObjectStorageIPs</b>
<b>SwiftStorageImage</b>	<b>ObjectStorageImage</b>
<b>OvercloudSwiftStorageFlavor</b>	<b>OvercloudObjectStorageFlavor</b>

Update these parameters in your custom environment files.

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
<b>--control-scale</b>	The number of Controller nodes to scale out	<b>ControllerCount</b>
<b>--compute-scale</b>	The number of Compute nodes to scale out	<b>ComputeCount</b>
<b>--ceph-storage-scale</b>	The number of Ceph Storage nodes to scale out	<b>CephStorageCount</b>
<b>--block-storage-scale</b>	The number of Cinder nodes to scale out	<b>BlockStorageCount</b>
<b>--swift-storage-scale</b>	The number of Swift nodes to scale out	<b>ObjectStorageCount</b>
<b>--control-flavor</b>	The flavor to use for Controller nodes	<b>OvercloudControllerFlavor</b>
<b>--compute-flavor</b>	The flavor to use for Compute nodes	<b>OvercloudComputeFlavor</b>
<b>--ceph-storage-flavor</b>	The flavor to use for Ceph Storage nodes	<b>OvercloudCephStorageFlavor</b>

Option	Description	Heat Template Parameter
<b>--block-storage-flavor</b>	The flavor to use for Cinder nodes	<b>OvercloudBlockStorageFlavor</b>
<b>--swift-storage-flavor</b>	The flavor to use for Swift storage nodes	<b>OvercloudSwiftStorageFlavor</b>
<b>--neutron-flat-networks</b>	Defines the flat networks to configure in neutron plugins. Defaults to "datacentre" to permit external network creation	<b>NeutronFlatNetworks</b>
<b>--neutron-physical-bridge</b>	An Open vSwitch bridge to create on each hypervisor. This defaults to "br-ex". Typically, this should not need to be changed	<b>HypervisorNeutronPhysicalBridge</b>
<b>--neutron-bridge-mappings</b>	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	<b>NeutronBridgeMappings</b>
<b>--neutron-public-interface</b>	Defines the interface to bridge onto br-ex for network nodes	<b>NeutronPublicInterface</b>
<b>--neutron-network-type</b>	The tenant network type for Neutron	<b>NeutronNetworkType</b>
<b>--neutron-tunnel-types</b>	The tunnel types for the Neutron tenant network. To specify multiple values, use a comma separated string	<b>NeutronTunnelTypes</b>
<b>--neutron-tunnel-id-ranges</b>	Ranges of GRE tunnel IDs to make available for tenant network allocation	<b>NeutronTunnelIdRanges</b>
<b>--neutron-vni-ranges</b>	Ranges of VXLAN VNI IDs to make available for tenant network allocation	<b>NeutronVniRanges</b>
<b>--neutron-network-vlan-ranges</b>	The Neutron ML2 and Open vSwitch VLAN mapping range to support. Defaults to permitting any VLAN on the 'datacentre' physical network	<b>NeutronNetworkVLANRanges</b>

Option	Description	Heat Template Parameter
<b>--neutron-mechanism-drivers</b>	The mechanism drivers for the neutron tenant network. Defaults to "openvswitch". To specify multiple values, use a comma-separated string	<b>NeutronMechanismDrivers</b>
<b>--neutron-disable-tunneling</b>	Disables tunneling in case you aim to use a VLAN segmented network or flat network with Neutron	No parameter mapping.
<b>--validation-errors-fatal</b>	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
<b>--ntp-server</b>	Sets the NTP server to use to synchronize time	<b>NtpServer</b>

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**.

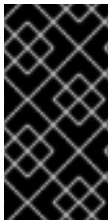
The following table provides a mapping of composable networks to custom standalone roles:

Role	Networks Required
Ceph Storage Monitor	<b>Storage, StorageMgmt</b>

Role	Networks Required
Ceph Storage OSD	<b>Storage, StorageMgmt</b>
Ceph Storage RadosGW	<b>Storage, StorageMgmt</b>
Cinder API	<b>InternalApi</b>
Compute	<b>InternalApi, Tenant, Storage</b>
Controller	<b>External, InternalApi, Storage, StorageMgmt, Tenant</b>
Database	<b>InternalApi</b>
Glance	<b>InternalApi</b>
Heat	<b>InternalApi</b>
Horizon	<b>InternalApi</b>
Ironic	None required. Uses the Provisioning/Control Plane network for API.
Keystone	<b>InternalApi</b>
Load Balancer	<b>External, InternalApi, Storage, StorageMgmt, Tenant</b>
Manila	<b>InternalApi</b>
Message Bus	<b>InternalApi</b>
Networker	<b>InternalApi, Tenant</b>
Neutron API	<b>InternalApi</b>
Nova	<b>InternalApi</b>
OpenDaylight	<b>External, InternalApi, Tenant</b>
Redis	<b>InternalApi</b>
Sahara	<b>InternalApi</b>
Swift API	<b>Storage</b>
Swift Storage	<b>StorageMgmt</b>



Role	Networks Required
Telemetry	<b>InternalApi</b>



## 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. 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 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 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'
```

## 5.6. 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.

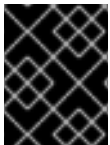


### WARNING

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 ["Isolating Networks"](#).



### 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](#).

## 5.7. 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 **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 ALL CONTROLLER NODES

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

### Procedure

1. Source the **stackrc** file:

```
$ source ~/stackrc
```

2. Run the upgrade command:

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

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

3. Wait until the Controller node upgrade completes.

## 6.3. UPGRADING ALL COMPUTE NODES

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

3. Wait until the Compute node upgrade completes.

## 6.4. UPGRADING ALL CEPH STORAGE NODES

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. 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 systemctl restart neutron-openvswitch-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
```