Red Hat OpenStack Platform 13

Fast Forward Upgrades

Upgrading across long life versions from Red Hat OpenStack Platform 10 to 13

Last Updated: 2018-10-18

Red Hat OpenStack Platform 13 Fast Forward Upgrades

Upgrading across long life versions from Red Hat OpenStack Platform 10 to 13

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This guide provides the fast forward upgrade process. This process upgrades your OpenStack Platform environment from one long life version to the next long life version. In this case, the guide focuses on upgrading from Red Hat OpenStack Platform 10 (Newton) to 13 (Queens).

Table of Contents

CHAPTER 1. INTRODUCTION

This document provides a workflow to help upgrade your Red Hat OpenStack Platform environment to the latest long life version.

1.1. BEFORE YOU BEGIN

Note the following:

- The fast forward upgrade workflow is currently considered under development. In particular, invocations of the **`ffwd-upgrade`** CLI command should be initially limited to development and test environments. If you decide to use fast forward upgrade in production, contact Red Hat's Customer Experience and Engagement team (https://access.redhat.com/support) to obtain support before attempting your production-level fast forward upgrade.
- If you originally deployed your Red Hat OpenStack Platform environment using version 7 or 8, be aware there is an issue involving an older version of the XFS file system that will hinder your upgrade path and deployment with containerized services. For more information about the issue and how to resolve it, see the following article:
  - "XFS ftype=0 prevents upgrading to a version of OpenStack Director with containers".

1.2. FAST FORWARD UPGRADES

Red Hat OpenStack Platform provides a **fast forward upgrade** feature. This feature provides an upgrade path through multiple versions of the overcloud. The goal is to provide users an opportunity to remain on certain OpenStack versions that are considered **long life versions** and upgrade when the next long life version is available.

This guide provides a fast forward upgrade path through the following versions:

Old Version	New Version
Red Hat OpenStack Platform 10	Red Hat OpenStack Platform 13

1.3. HIGH LEVEL WORKFLOW

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

Step	Description
Preparing your environment	Perform a backup of the databases and configuration for the undercloud node and overcloud Controller nodes. Update to the latest minor release and reboot. Validate the environment.
Upgrading the undercloud	Upgrade to each sequential version of the undercloud from OpenStack Platform 10 to OpenStack Platform 13.

Step	Description
Obtaining container images	Create an environment file containing the locations of container images for various OpenStack services.
Preparing the overcloud	Perform relevant steps to transition your overcloud configuration files to OpenStack Platform 13.
Performing the fast forward upgrade	Upgrade the overcloud plan with the latest set of OpenStack Platform director templates. Run package and database upgrades through each sequential version so that the database schema is ready for the upgrade 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 the containerized version of Red Hat Ceph Storage 3.
Finalize the upgrade	Run the convergence command to refresh your overcloud stack.

CHAPTER 2. PREPARING FOR AN OPENSTACK PLATFORM UPGRADE

This process prepares your OpenStack Platform environment. This involves the following steps:

- Backing up both the undercloud and overcloud.
- Updating the undercloud to the latest minor version of OpenStack Platform 10, including the latest Open vSwitch.
- Rebooting the undercloud in case a newer kernel or newer system packages are installed.
- Updating the overcloud to the latest minor version of OpenStack Platform 10, including the latest Open vSwitch.
- Rebooting the overcloud nodes in case a newer kernel or newer system packages are installed.
- Performing validation checks 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.

Related Information

- If you need to restore the undercloud backup, see Appendix A, *Restoring the undercloud*.

2.2. BACKING UP THE OVERCLOUD CONTROL PLANE SERVICES

The following procedure creates a backup of the 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$MYSQLDBPASS -s -N -e "SELECT CONCAT('\"SHOW
GRANTS FOR ''',user,'''@''',host,''';\"') FROM mysql.user where
(length(user) > 0 and user NOT LIKE 'root')" | xargs -n1 mysql -
uroot -p$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/`hostname`-filesystem-`date
'+%Y-%m-%d-%H-%M-%S'`.tar \
    /etc \
    /srv/node \
    /var/log \
    /var/lib/nova \
    --exclude /var/lib/nova/instances \
    /var/lib/glance \
    /var/lib/keystone \
    /var/lib/cinder \
    /var/lib/heat \
    /var/lib/heat-config \
    /var/lib/heat-cfntools \
    /var/lib/rabbitmq \
    /var/lib/neutron \
    /var/lib/haproxy \
    /var/lib/openvswitch \
    /var/lib/redis \
    /usr/libexec/os-apply-config \
    /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.

Related Information

- If you need to restore the overcloud backup, see Appendix B, *Restoring the overcloud*.

2.3. PREPARING UPDATES FOR NFV-ENABLED ENVIRONMENTS

If your environment has network function virtualization (NFV) enabled, you need to follow these steps before updating your undercloud and overcloud.

Procedure

1. Add the content from this sample post-install.yaml file to any existing **post-install.yaml** file.
2. Change the vhost user socket directory in a custom environment file, for example, **network-environment.yaml**:

```
parameter_defaults:
  NeutronVhostuserSocketDir: "/var/lib/vhost_sockets"
```

3. Add the **ovs-dpdk-permissions.yaml** file to your **openstack overcloud deploy** command to configure the qemu group setting as **hugetlbfs** for OVS-DPDK:

```
-e environments/ovs-dpdk-permissions.yaml
```

2.4. UPDATING THE CURRENT UNDERCLOUD PACKAGES FOR OPENSTACK PLATFORM 10.Z

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. This is a minor update within **OpenStack Platform 10**.

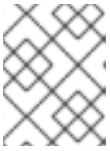
NOTE

This step also updates the undercloud operating system to the latest version of Red Hat Enterprise Linux 7 and Open vSwtich to version 2.9.

Procedure

1. Log into the undercloud as the **stack** user.
2. Stop the main OpenStack Platform services:

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```



NOTE

This causes a short period of downtime for the undercloud. The overcloud is still functional during the undercloud upgrade.

3. 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
```

4. Run the **openstack undercloud upgrade** command:

```
$ openstack undercloud upgrade
```

5. Wait until the command completes its execution.

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

```
$ sudo reboot
```

7. Wait until the node boots.

8. Log into the undercloud as the **stack** user.

In addition to undercloud package updates, it is recommended to keep your overcloud images up to date to keep the image configuration in sync with the latest **openstack-tripleo-heat-template** package. This ensures successful deployment and scaling operations in between the current preparation stage and the actual fast forward upgrade. The next section shows how to update your images in this scenario. If you aim to immediately upgrade your environment after preparing your environment, you can skip the next section.

2.5. UPDATING THE CURRENT OVERCLOUD IMAGES FOR OPENSTACK PLATFORM 10.Z

The undercloud update process might download new image archives from the **rhosp-director-images** and **rhosp-director-images-ipa** packages. This process updates these images on your undercloud within **Red Hat OpenStack Platform 10**.

Prerequisites

- You have updated to the latest minor release of your current undercloud version.

Procedure

1. Check the **yum** log to determine if new image archives are available:

```
$ sudo grep "rhosp-director-images" /var/log/yum.log
```

2. If new archives are available, replace your current images with new images. To install the new images, first 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-
10.0.tar /usr/share/rhosp-director-images/ironic-python-agent-
latest-10.0.tar; do tar -xvf $i; done
```

4. Import the latest images into the director and configure nodes to use the new images

```
$ cd ~
$ openstack overcloud image upload --update-existing --image-path
/home/stack/images/
$ openstack overcloud node configure $(openstack baremetal node list
-c UUID -f csv --quote none | sed "1d" | paste -s -d " ")
```

5. To finalize the image update, verify the existence of the new images:

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

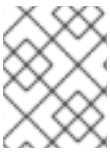
   The director also retains the old images and renames them using the timestamp of when they were updated. If you no longer need these images, delete them.

The director is now updated and using the latest images. You do not need to restart any services after the update.

The undercloud is now using updated OpenStack Platform 10 packages. Next, update the overcloud to the latest minor release.

2.6. UPDATING THE CURRENT OVERCLOUD PACKAGES FOR OPENSTACK PLATFORM 10.Z

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. This is a minor update within **Red Hat OpenStack Platform 10**.



NOTE

This step also updates the overcloud nodes' operating system to the latest version of Red Hat Enterprise Linux 7 and Open vSwtich to version 2.9.

Prerequisites

- You have updated to the latest minor release of your current undercloud version.
- You have performed a backup of the overcloud.

Procedure

1. Update the current plan using your original **`openstack overcloud deploy`** command and including the **`--update-plan-only`** option. For example:

```
$ openstack overcloud deploy --update-plan-only \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-
templates/environments/network-isolation.yaml \
  -e /home/stack/templates/network-environment.yaml \
  -e /home/stack/templates/storage-environment.yaml \
  -e /home/stack/templates/rhel-registration/environment-rhel-
registration.yaml \
  [-e <environment_file>|...]
```

The **--update-plan-only** only updates the Overcloud plan stored in the director. Use the **-e** option to include environment files relevant to your Overcloud and its update path. The order of the environment files is important as the parameters and resources defined in subsequent environment files take precedence. Use the following list as an example of the environment file order:

- Any network isolation files, including the initialization file (**environments/network-isolation.yaml**) from the heat template collection and then your custom NIC configuration file.
- Any external load balancing environment files.
- Any storage environment files.
- Any environment files for Red Hat CDN or Satellite registration.
- Any other custom environment files.

2. Perform a package update on all nodes using the **openstack overcloud update** command:

```
$ openstack overcloud update stack -i overcloud
```

The **-i** runs an interactive mode to update each node sequentially. When the update process completes a node update, the script provides a breakpoint for you to confirm. Without the **-i** option, the update remains paused at the first breakpoint. Therefore, it is mandatory to include the **-i** option.

The script performs the following functions:

a. The script runs on nodes one-by-one:

   i. For Controller nodes, this means a full package update.

   ii. For other nodes, this means an update of Puppet modules only.

b. Puppet runs on all nodes at once:

   i. For Controller nodes, the Puppet run synchronizes the configuration.

   ii. For other nodes, the Puppet run updates the rest of the packages and synchronizes the configuration.

3. The update process starts. During this process, the director reports an **IN_PROGRESS** status and periodically prompts you to clear breakpoints. For example:

```
starting package update on stack overcloud
```

```
IN_PROGRESS
IN_PROGRESS
WAITING
on_breakpoint: [u'overcloud-compute-0', u'overcloud-controller-2',
u'overcloud-controller-1', u'overcloud-controller-0']
Breakpoint reached, continue? Regexp or Enter=proceed (will clear
49913767-e2dd-4772-b648-81e198f5ed00), no=cancel update, C-c=quit
interactive mode:
```

Press Enter to clear the breakpoint from last node on the **`on_breakpoint`** list. This begins the update for that node.

4. The script automatically predefines the update order of nodes:
   - Each Controller node individually
   - Each individual Compute node individually
   - Each Ceph Storage node individually
   - All other nodes individually

   It is recommended to use this order to ensure a successful update, specifically:

   a. Clear the breakpoint of each Controller node individually. Each Controller node requires an individual package update in case the node's services must restart after the update. This reduces disruption to highly available services on other Controller nodes.

   b. After the Controller node update, clear the breakpoints for each Compute node. You can also type a Compute node name to clear a breakpoint on a specific node or use a Python-based regular expression to clear breakpoints on multiple Compute nodes at once.

   c. Clear the breakpoints for each Ceph Storage nodes. You can also type a Ceph Storage node name to clear a breakpoint on a specific node or use a Python-based regular expression to clear breakpoints on multiple Ceph Storage nodes at once.

   d. Clear any remaining breakpoints to update the remaining nodes. You can also type a node name to clear a breakpoint on a specific node or use a Python-based regular expression to clear breakpoints on multiple nodes at once.

   e. Wait until all nodes have completed their update.

5. The update command reports a **`COMPLETE`** status when the update completes:

```
...
IN_PROGRESS
IN_PROGRESS
IN_PROGRESS
COMPLETE
update finished with status COMPLETE
```

6. If you configured fencing for your Controller nodes, the update process might disable it. When the update process completes, reenable fencing with the following command on one of the Controller nodes:

```
$ sudo pcs property set stonith-enabled=true
```

The update process does not reboot any nodes in the Overcloud automatically. Updates to the kernel and other system packages require a reboot. Check the **/var/log/yum.log** file on each node to see if either the **kernel** or **openvswitch** packages have updated their major or minor versions. If they have, reboot each node using the following procedures.

2.7. 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
```

2.8. 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
```

2.9. 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
    ```

2.10. VERIFYING SYSTEM PACKAGES

Before the upgrade, all nodes should be using the latest versions of the following packages:

Package	Version
openvswitch	At least 2.9
qemu-img-rhev	At least 2.10
qemu-kvm-common-rhev	At least 2.10
qemu-kvm-rhev	At least 2.10
qemu-kvm-tools-rhev	At least 2.10

Use the following procedure on each node to check the package versions.

Procedure

1. Log into a node.
2. Run **yum** to check the system packages:

```
$ sudo yum list qemu-img-rhev qemu-kvm-common-rhev qemu-kvm-rhev
qemu-kvm-tools-rhev openvswitch
```

3. Run **ovs-vsctl** to check the version currently running:

```
$ sudo ovs-vsctl --version
```

The undercloud is now uses updated OpenStack Platform 10 packages. Use the next few procedures to check the system is in a working state.

2.11. VALIDATING AN OPENSTACK PLATFORM 10 UNDERCLOUD

The following is a set of steps to check the functionality of your **Red Hat OpenStack Platform 10** undercloud before an upgrade.

Procedure

1. Source the undercloud access details:

```
$ source ~/stackrc
```

2. Check for failed Systemd services:

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

3. Check the undercloud free space:

```
$ df -h
```

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

4. If you have NTP installed on the undercloud, check the clock is synchronized:

```
$ sudo ntpstat
```

5. Check the undercloud network services:

```
$ openstack network agent list
```

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

6. Check the undercloud compute services:

```
$ 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

2.12. VALIDATING AN OPENSTACK PLATFORM 10 OVERCLOUD

The following is a set of steps to check the functionality of your **Red Hat OpenStack Platform 10** overcloud before an upgrade.

Procedure

1. Source the undercloud access details:

```
$ source ~/stackrc
```

2. Check the status of your bare metal nodes:

```
$ 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:

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

```
$ 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 /etc/haproxy/haproxy.cfg'
```

Use these details in the following cURL request:

```
$ 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.

5. Check overcloud database replication health:

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

6. Check RabbitMQ cluster health:

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

7. Check Pacemaker resource health:

```
$ 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.

8. Check the disk space on each overcloud node:

```
$ 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
```

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

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

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

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

11. Check that clocks are synchronized on overcloud nodes

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

12. Source the overcloud access details:

```
$ source ~/overcloudrc
```

13. Check the overcloud network services:

```
$ openstack network agent list
```

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

14. Check the overcloud compute services:

```
$ openstack compute service list
```

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

15. Check the overcloud volume services:

```
$ 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.
- Review the article "Database Size Management for Red Hat Enterprise Linux OpenStack Platform" to check and clean unused database records for OpenStack Platform services on the overcloud.

2.13. FINALIZING UPDATES FOR NFV-ENABLED ENVIRONMENTS

If your environment has network function virtualization (NFV) enabled, you need to follow these steps after updating your undercloud and overcloud.

Procedure

You need to migrate your existing OVS-DPDK instances to ensure that the vhost socket mode changes from **dkdpvhostuser** to **dkdpvhostuserclient** mode in the OVS ports. We recommend that you snapshot existing instances and rebuild a new instance based on that snapshot image. See Manage Instance Snapshots for complete details on instance snapshots.

To snapshot an instance and boot a new instance from the snapshot:

1. Find the server ID for the instance you want to take a snapshot of:

```
# openstack server list
```

2. Shut down the source instance before you take the snapshot to ensure that all data is flushed to disk:

```
# openstack server stop SERVER_ID
```

3. Create the snapshot image of the instance:

```
# openstack image create --id SERVER_ID SNAPSHOT_NAME
```

4. Boot a new instance with this snapshot image:

```
# openstack server create --flavor DPDK_FLAVOR --nic net-
id=DPDK_NET_ID--image SNAPSHOT_NAME INSTANCE_NAME
```

5. Optionally, verify that the new instance status is **ACTIVE**:

```
# openstack server list
```

Repeat this procedure for all instances that you need to snapshot and relaunch.

2.14. NEXT STEPS

With the preparation stage complete, you can now perform an upgrade of the undercloud from 10 to 13 using the steps in Chapter 3, *Upgrading the undercloud*.

CHAPTER 3. UPGRADING THE UNDERCLOUD

This following procedures upgrades the undercloud to **Red Hat OpenStack Platform 13**. You accomplish this by performing an upgrade through each sequential version of the undercloud from OpenStack Platform 10 to OpenStack Platform 13.

3.1. UPGRADING THE UNDERCLOUD TO OPENSTACK PLATFORM 11

This procedure upgrades the undercloud toolset and the core Heat template collection to the **OpenStack Platform 11** release.

Procedure

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

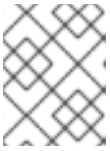
```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
```

3. Enable the new OpenStack Platform repository:

```
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
```

4. Stop the main OpenStack Platform services:

```
$ sudo systemctl stop 'openstack-*' 'neutron-*' httpd
```



NOTE

This causes a short period of downtime for the undercloud. The overcloud is still functional during the undercloud upgrade.

5. The default Provisioning/Control Plane network has changed from **192.0.2.0/24** to **192.168.24.0/24**. If you used default network values in your previous **undercloud.conf** file, your Provisioning/Control Plane network is set to **192.0.2.0/24**. This means you need to set certain parameters in your **undercloud.conf** file to continue using the **192.0.2.0/24** network. These parameters are:
   - **local_ip**
   - **network_gateway**
   - **undercloud_public_vip**
   - **undercloud_admin_vip**
   - **network_cidr**
   - **masquerade_network**
   - **dhcp_start**

- **dhcp_end**

Set the network values in **undercloud.conf** to ensure continued use of the **192.0.2.0/24** CIDR during future upgrades. Ensure your network configuration set correctly before running the **openstack undercloud upgrade** command.

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

```
$ sudo yum update -y instack-undercloud openstack-puppet-modules
openstack-tripleo-common python-tripleoclient
```

7. Run the following command to upgrade the undercloud:

```
$ openstack undercloud upgrade
```

8. Wait until the undercloud upgrade process completes.

You have upgraded the undercloud to the **OpenStack Platform 11** release.

3.2. UPGRADING THE UNDERCLOUD TO OPENSTACK PLATFORM 12

This procedure upgrades the undercloud toolset and the core Heat template collection to the **OpenStack Platform 12** 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-
11-rpms
```

3. Enable the new OpenStack Platform repository:

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

4. Install the **ceph-ansible** package when overcloud is configured with Ceph storage:

```
$ sudo yum install -y ceph-ansible
```

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

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

6. Edit the **/home/stack/undercloud.conf** file and check that the **enabled_drivers** parameter does not contain the **pxe_ssh** driver. This driver is deprecated in favor of the Virtual Baseboard Management Controller (VBMC) and removed from Red Hat OpenStack Platform. For more information about this new driver and migration instructions, see the Appendix "Virtual Baseboard Management Controller (VBMC)" in the *Director Installation and Usage Guide*.
7. Run the following command to upgrade the undercloud:

```
$ openstack undercloud upgrade
```

8. Wait until the undercloud upgrade process completes.

You have upgraded the undercloud to the **OpenStack Platform 12** release.

3.3. 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.

3.4. NEXT STEPS

The undercloud upgrade is complete. You can now configure a source for your container images.

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

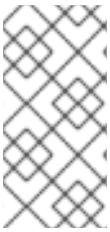
Red Hat OpenStack Platform supports the following registry types:

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

Service	Environment File
OpenStack Load Balancing-as-a-Service (octavia)	**environments/services-docker/octavia.yaml**
OpenStack Shared File System Storage (manila)	**environments/services-docker/manila.yaml**
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.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*.

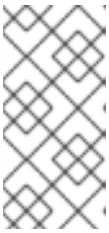
The examples in this procedure use the **hammer** command line tool for Red Hat Satellite 6 and an example organization called **ACME**. Substitute this organization for your own Satellite 6 organization.

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 $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 section prepares the overcloud for the upgrade process. Not all steps in this section will apply to your overcloud. However, it is recommended to step through each one and determine if your overcloud requires any additional configuration before the upgrade process begins.

5.1. PREPARING FOR OVERCLOUD SERVICE DOWNTIME

The overcloud upgrade process disables the main services at key points. This means you cannot use any overcloud services to create new resources during the upgrade duration. Workloads running in the overcloud remain active during this period, which means instances continue to run through the upgrade duration.

It is important to plan a maintenance window to ensure no users can access the overcloud services during the upgrade duration.

Affected by overcloud upgrade

- OpenStack Platform services

Unaffected by overcloud upgrade

- Instances running during the upgrade
- Ceph Storage OSDs (backend storage for instances)
- Linux networking
- Open vSwitch networking
- Undercloud

5.2. SELECTING COMPUTE NODES FOR UPGRADE TESTING

The overcloud upgrade process allows you to either:

- Upgrade all nodes in a role
- Individual nodes separately

To ensure a smooth overcloud upgrade process, it is useful to test the upgrade on a few individual Compute nodes in your environment before upgrading all Compute nodes. This ensures no major issues occur during the upgrade while maintaining minimal downtime to your workloads.

Use the following recommendations to help choose test nodes for the upgrade:

- Select two or three Compute nodes for upgrade testing
- Select nodes without any critical instances running
- If necessary, migrate critical instances from the selected test Compute nodes to other Compute nodes

The instructions in Chapter 6, *Upgrading the overcloud* use **`compute-0`** as an example of a Compute node to test the upgrade process before running the upgrade on all Compute nodes.

The next step updates your **roles_data** file to ensure any new composable services have been added to the relevant roles in your environment. To manually edit your existing **roles_data** file, use the following lists of new composable services for OpenStack Platform 13 roles.

NOTE

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

5.3. NEW COMPOSABLE SERVICES

This version of Red Hat OpenStack Platform contains new composable services. If using a custom **roles_data** file with your own roles, include these new compulsory services in their applicable roles.

All Roles

The following new services apply to all roles.

OS::TripleO::Services::MySQLClient

Configures the MariaDB client on a node, which provides database configuration for other composable services. Add this service to all roles with standalone composable services.

OS::TripleO::Services::CertmongerUser

Allows the overcloud to require certificates from Certmonger. Only used if enabling TLS/SSL communication.

OS::TripleO::Services::Docker

Installs **docker** to manage containerized services.

OS::TripleO::Services::ContainersLogrotateCrond

Installs the **logrotate** service for container logs.

OS::TripleO::Services::Securetty

Allows configuration of **securetty** on nodes. Enabled with the **environments/securetty.yaml** environment file.

OS::TripleO::Services::Tuned

Enables and configures the Linux tuning daemon (**tuned**).

OS::TripleO::Services::AuditD

Adds the **auditd** daemon and configures rules. Disabled by default.

OS::TripleO::Services::Collectd

Adds the **collectd** daemon. Disabled by default.

OS::TripleO::Services::Rhsm

Configures subscriptions using an Ansible-based method. Disabled by default.

OS::TripleO::Services::RsyslogSidecar

Configures a sidecar container for logging. Disabled by default.

Specific Roles

The following new services apply to specific roles:

`OS::TripleO::Services::NovaPlacement`

Configures the OpenStack Compute (nova) Placement API. If using a standalone Nova API role in your current overcloud, add this service to the role. Otherwise, add the service to the Controller role.

`OS::TripleO::Services::PankoApi`

Configures the OpenStack Telemetry Event Storage (panko) service. If using a standalone Telemetry role in your current overcloud, add this service to the role. Otherwise, add the service to the Controller role.

`OS::TripleO::Services::Clustercheck`

Required on any role that also uses the **`OS::TripleO::Services::MySQL`** service, such as the **Controller** or standalone **Database** role.

`OS::TripleO::Services::Iscsid`

Configures the **`iscsid`** service on the **Controller**, **Compute**, and **BlockStorage** roles.

`OS::TripleO::Services::NovaMigrationTarget`

Configures the migration target service on **Compute** nodes.

`OS::TripleO::Services::Ec2Api`

Enables the OpenStack Compute (nova) EC2-API service on **Controller** nodes. Disabled by default.

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

Enables the Ceph Manager service on **Controller** nodes. Enabled as a part of the **`ceph-ansible`** configuration.

`OS::TripleO::Services::CephMds`

Enables the Ceph Metadata Service (MDS) on **Controller** nodes. Disabled by default.

`OS::TripleO::Services::CephRbdMirror`

Enables the RADOS Block Device (RBD) mirroring service. Disabled by default.

In addition, see the "Service Architecture: Standalone Roles" section in the *Advanced Overcloud Customization* guide for updated lists of services for specific custom roles.

In addition to new composable services, take note of any deprecated services since OpenStack Platform 13.

5.4. DEPRECATED COMPOSABLE SERVICES

If using a custom **`roles_data`** file, remove these services from their applicable roles.

`OS::TripleO::Services::Core`

This service acted as a core dependency for other Pacemaker services. This service has been removed to accommodate high availability composable services.

`OS::TripleO::Services::VipHosts`

This service configured the /etc/hosts file with node hostnames and IP addresses. This service is now integrated directly into the director's Heat templates.

`OS::TripleO::Services::FluentdClient`

This service has been replaced with the **`OS::TripleO::Services::Fluentd`** service.

`OS::TripleO::Services::ManilaBackendGeneric`

The Manila generic backend is no longer supported.

If using a custom **`roles_data`** file, remove these services from their respective roles.

In addition, see the "Service Architecture: Standalone Roles" section in the *Advanced Overcloud Customization* guide for updated lists of services for specific custom roles.

5.5. DEPRECATED PARAMETERS

Note that the following parameters are deprecated and have been replaced with role-specific parameters:

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

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
  ...
```

5.6. 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**
--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**

Option	Description	Heat Template Parameter
--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**
--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**

Option	Description	Heat Template Parameter
--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.

Later examples in this guide include an **deprecated_cli_options.yaml** environment file that includes these new parameters.

5.7. 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**
Networker	**InternalApi**, **Tenant**
Neutron API	**InternalApi**
Nova	**InternalApi**
OpenDaylight	**External**, **InternalApi**, **Tenant**
Redis	**InternalApi**

Role	Networks Required
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.

5.8. PREPARING FOR CEPH STORAGE NODE UPGRADES

Due to the upgrade to containerized services, the method for installing and updating Ceph Storage nodes has changed. Ceph Storage configuration now uses a set of playbooks in the **ceph-ansible** package, which you install on the undercloud.

Prerequisites

- Your overcloud has a director-managed Ceph Storage cluster.

Procedure

1. Install the **ceph-ansible** package to the undercloud:

```
[stack@director ~]$ sudo yum install -y ceph-ansible
```

2. Check that you are using the latest resources and configuration in your storage environment file. This requires the following changes:

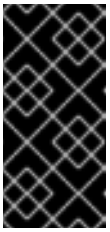
   a. The **resource_registry** uses containerized services from the **docker/services** subdirectory of your core Heat template collection instead of the **puppet/services** subdirectory. For example, replace:

```
resource_registry:
  OS::TripleO::Services::CephMon: /usr/share/openstack-tripleo-
heat-templates/puppet/services/ceph-mon.yaml
  OS::TripleO::Services::CephOSD: /usr/share/openstack-tripleo-
heat-templates/puppet/services/ceph-osd.yaml
  OS::TripleO::Services::CephClient: /usr/share/openstack-
tripleo-heat-templates/puppet/services/ceph-client.yaml
```

   With:

```
resource_registry:
```

```
  OS::TripleO::Services::CephMon: /usr/share/openstack-tripleo-
heat-templates/docker/services/ceph-ansible/ceph-mon.yaml
  OS::TripleO::Services::CephOSD: /usr/share/openstack-tripleo-
heat-templates/docker/services/ceph-ansible/ceph-osd.yaml
  OS::TripleO::Services::CephClient: /usr/share/openstack-
tripleo-heat-templates/docker/services/ceph-ansible/ceph-
client.yaml
```



IMPORTANT

This configuration is included in the **`/usr/share/openstack-tripleo-heat-templates/environments/ceph-ansible/ceph-ansible.yaml`** environment file, which you can include with all future deployment commands with **`-e`**.

b. Use the new **`CephAnsibleDisksConfig`** parameter to define how your disks are mapped. Previous versions of Red Hat OpenStack Platform used the **`ceph::profile::params::osds`** hieradata to define the OSD layout. Convert this hieradata to the structure of the **`CephAnsibleDisksConfig`** parameter. For example, if your hieradata contained the following:

```
parameter_defaults:
  ExtraConfig:
    ceph::profile::params::osd_journal_size: 512
    ceph::profile::params::osds:
      '/dev/sdb': {}
      '/dev/sdc': {}
      '/dev/sdd': {}
```

Then the **`CephAnsibleDisksConfig`** would look like this:

```
parameter_defaults:
  ExtraConfig: {}
  CephAnsibleDisksConfig:
    devices:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
    journal_size: 512
    osd_scenario: collocated
```

For a full list of OSD disk layout options used in **`ceph-ansible`**, view the sample file in **`/usr/share/ceph-ansible/group_vars/osds.yml.sample`**.

3. Make sure to include the new Ceph configuration environment files with future deployment commands using the **`-e`** option.

5.9. PREPARING STORAGE BACKENDS

Some storage backends have changed from using configuration hooks to their own composable service. If using a custom storage backend, check the associated environment file in the **`environments`** directory for new parameters and resources. Update any custom environment files for your backends. For example:

- For the **NetApp Block Storage (cinder)** backend, use the new **`environments/cinder-netapp-config.yaml`** in your deployment.
- For the **Dell EMC Block Storage (cinder)** backend, use the new **`environments/cinder-dellsc-config.yaml`** in your deployment.
- For the **Dell EqualLogic Block Storage (cinder)** backend, use the new **`environments/cinder-dellps-config.yaml`** in your deployment.

For example, the **NetApp Block Storage (cinder)** backend used the following resources for these respective versions:

- OpenStack Platform 10 and below: **`OS::TripleO::ControllerExtraConfigPre: ../puppet/extraconfig/pre_deploy/controller/cinder-netapp.yaml`**
- OpenStack Platform 11 and above: **`OS::TripleO::Services::CinderBackendNetApp: ../puppet/services/cinder-backend-netapp.yaml`**

As a result, you now use the new **`OS::TripleO::Services::CinderBackendNetApp`** resource and its associated service template for this backend.

5.10. PREPARING ACCESS TO THE UNDERCLOUD'S PUBLIC API OVER SSL/TLS

The overcloud requires access to the undercloud's OpenStack Object Storage (swift) Public API during the upgrade. If your undercloud uses a self-signed certificate, you need to add the undercloud's certificate authority to each overcloud node.

Prerequisites

- The undercloud uses SSL/TLS for its Public API

Procedure

1. The director's dynamic Ansible script has updated to the OpenStack Platform 12 version, which uses the **`RoleNetHostnameMap`** Heat parameter in the overcloud plan to define the inventory. However, the overcloud currently uses the OpenStack Platform 11 template versions, which do not have the **`RoleNetHostnameMap`** parameter. This means you need to create a temporary static inventory file, which you can generate with the following command:

```
$ openstack server list -c Networks -f value | cut -d"=" -f2 >
overcloud_hosts
```

2. Create an Ansible playbook (**`undercloud-ca.yml`**) that contains the following:

```
---
- name: Add undercloud CA to overcloud nodes
  hosts: all
  user: heat-admin
  become: true
  vars:
    ca_certificate: /etc/pki/ca-trust/source/anchors/cm-local-ca.pem
  tasks:
    - name: Copy undercloud CA
```

```
      copy:
        src: "{{ ca_certificate }}"
        dest: /etc/pki/ca-trust/source/anchors/
    - name: Update trust
      command: "update-ca-trust extract"
    - name: Get the swift endpoint
      shell: |
        sudo hiera swift::keystone::auth::public_url | awk -F/
'{print $3}'
      register: swift_endpoint
      delegate_to: 127.0.0.1
      become: yes
      become_user: stack
    - name: Verify URL
      uri:
        url: https://{{ swift_endpoint.stdout }}/healthcheck
        return_content: yes
      register: verify
    - name: Report output
      debug:
        msg: "{{ ansible_hostname }} can access the undercloud's
Public API"
      when: verify.content == "OK"
```

This playbook contains multiple tasks that perform the following on each node:

- Copy the undercloud's certificate authority file to the overcloud node. If generated by the undercloud, the default location is **`/etc/pki/ca-trust/source/anchors/cm-local-ca.pem`**.
- Execute the command to update the certificate authority trust database on the overcloud node.
- Checks the undercloud's Object Storage Public API from the overcloud node and reports if successful.

3. Run the playbook with the following command:

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml
```

This uses the temporary inventory to provide Ansible with your overcloud nodes.

If using a custom certificate authority file, you can change the **`ca_certificate`** variable to a location. For example:

```
$ ansible-playbook -i overcloud_hosts undercloud-ca.yml -e
ca_certificate=/home/stack/ssl/ca.crt.pem
```

4. The resulting Ansible output should show a debug message for node. For example:

```
ok: [192.168.24.100] => {
    "msg": "overcloud-controller-0 can access the undercloud's
Public API"
}
```

Related Information

- For more information on running Ansible automation on your overcloud, see "Running the dynamic inventory script" in the *Director Installation and Usage* guide.

5.11. CONFIGURING REGISTRATION FOR FAST FORWARD UPGRADES

The fast forward upgrade process uses a new method to switch repositories. This means you need to remove the old **`rhel-registration`** environment files from your deployment command. For example:

- environment-rhel-registration.yaml
- rhel-registration-resource-registry.yaml

The fast forward upgrade process uses a script to change repositories during each stage of the upgrade. This script is included as part of the **`OS::TripleO::Services::TripleoPackages`** composable service (**`puppet/services/tripleo-packages.yaml`**) using the **`FastForwardCustomRepoScriptContent`** parameter. This is the script:

```
#!/bin/bash
set -e
case $1 in
  ocata)
    subscription-manager repos --disable=rhel-7-server-openstack-10-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-11-rpms
    ;;
  pike)
    subscription-manager repos --disable=rhel-7-server-openstack-11-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-12-rpms
    ;;
  queens)
    subscription-manager repos --disable=rhel-7-server-openstack-12-rpms
    subscription-manager repos --enable=rhel-7-server-openstack-13-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-osd-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-mon-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-mon-rpms
    subscription-manager repos --disable=rhel-7-server-rhceph-2-tools-rpms
    subscription-manager repos --enable=rhel-7-server-rhceph-3-tools-rpms
    ;;
  *)
    echo "unknown release $1" >&2
    exit 1
esac
```

The director passes the upstream codename of each OpenStack Platform version to the script:

Codename	Version
`ocata`	OpenStack Platform 11
`pike`	OpenStack Platform 12

Codename	Version
queens	OpenStack Platform 13

The change to **queens** also disables Ceph Storage 2 repositories and enables the Ceph Storage 3 MON and Tools repositories. The change does not enable the Ceph Storage 3 OSD repositories because these are now containerized.

In some situations, you might need to use a custom script. For example:

- Using Red Hat Satellite with custom repository names.
- Using a disconnected repository with custom names.
- Additional commands to execute at each stage.

In these situations, include your custom script by setting the **`FastForwardCustomRepoScriptContent`** parameter:

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    [INSERT UPGRADE SCRIPT HERE]
```

For example, use the following script to change repositories with a set of Satellite 6 activation keys:

```
parameter_defaults:
  FastForwardCustomRepoScriptContent: |
    set -e
    URL="satellite.example.com"
    case $1 in
      ocata)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp11 --org=Default_Organization
        ;;
      pike)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp12 --org=Default_Organization
        ;;
      queens)
        subscription-manager register --baseurl=https://$URL --force --
activationkey=rhosp13 --org=Default_Organization
        ;;
      *)
        echo "unknown release $1" >&2
        exit 1
    esac
```

Later examples in this guide include an **`custom_repositories_script.yaml`** environment file that includes your custom script.

5.12. 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'
```

5.13. 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]
```

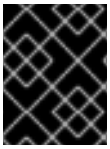
This configuration takes the interface configuration stored in the **`$network_config`** variable and injects it as a part of the **`run-os-net-config.sh`** script.

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".

5.14. NEXT STEPS

The overcloud preparation stage is complete. You can now perform an upgrade of the overcloud from 10 to 13 using the steps in Chapter 6, *Upgrading the overcloud*.

CHAPTER 6. UPGRADING THE OVERCLOUD

This section upgrades the overcloud. This includes the following workflow:

- Running the fast forward upgrade preparation command
- Running the fast forward upgrade command
- Upgrading the Controller nodes
- Upgrading the Compute nodes
- Upgrading the Ceph Storage nodes
- Finalizing the fast forward upgrade.

Once you begin this workflow, you should not expect full control over the overcloud's OpenStack services until completing all steps. This means workloads are unmanageable until all nodes have been successfully upgraded to OpenStack Platform 13. The workloads themselves will remain unaffected and continue to run. Changes or additions to any overcloud workloads need to wait until the fast forward upgrade is completed.

6.1. PERFORMING THE FAST FORWARD UPGRADE OF THE OVERCLOUD

The fast forward upgrade requires running two commands that perform the following tasks:

- Updates the overcloud plan to OpenStack Platform 13.
- Prepares the nodes for the fast forward upgrade.
- Runs through upgrade steps of each subsequent version within the fast forward upgrade, including:
  - Version-specific tasks for each OpenStack Platform service.
  - Changing the repository to each sequential OpenStack Platform version within the fast forward upgrade.
  - Updates certain packages required for upgrading the database.
  - Performing database upgrades for each subsequent version.
- Prepares the overcloud for the final upgrade to OpenStack Platform 13.

Procedure

1. Source the **`stackrc`** file:

```
$ source ~/stackrc
```

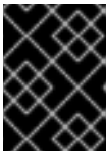
2. Run the fast forward upgrade preparation command:

```
$ openstack overcloud ffwd-upgrade prepare \
    --templates \
```

```
    -e /home/stack/templates/overcloud_images.yaml \
    -e /home/stack/templates/deprecated_cli_options.yaml \
    -e /home/stack/templates/custom_repositories_script.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.
  - If applicable, an environment file that maps deprecated CLI options to Heat parameters using **deprecated_cli_options.yaml**.
  - If applicable, an environment file with your custom repository script using **custom_repositories_script.yaml**.
  - If using Ceph Storage nodes, the relevant environment files.
  - 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**.



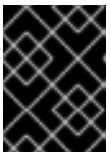
IMPORTANT

A prompt will ask if you are sure you want to perform the **ffwd-upgrade** command. Enter **yes**.

3. The overcloud plan updates to the OpenStack Platform 13 version. Wait until the fast forward upgrade preparation completes.
4. Run the fast forward upgrade command:

```
$ openstack overcloud ffwd-upgrade run
```

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



IMPORTANT

A prompt will ask if you are sure you want to perform the **ffwd-upgrade** command. Enter **yes**.

5. Wait until the fast forward upgrade completes.

At this stage:

- Workloads are still running
- The overcloud database has been upgraded to the OpenStack Platform 12 version
- All overcloud services are disabled
- Ceph Storage nodes are still at version 2

This means the overcloud is now at a state to perform the standard upgrade steps to reach OpenStack Platform 13.

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 --skip-tags
validation
```

NOTE

The command uses **`--skip-tags validation`** because OpenStack Platform services are inactive on the overcloud and cannot be validated.

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

3. Wait until the Controller node upgrade completes.

At this stage:

- Workloads are still running
- The overcloud database has been upgraded to the OpenStack Platform 13 version
- The Controller nodes have been upgraded to OpenStack Platform 13
- All Controller services are enabled
- The Compute nodes still require an upgrade
- Ceph Storage nodes are still at version 2 and require an upgrade

WARNING

Although Controller services are enabled, do not perform any workload operations while Compute node and Ceph Storage services are disabled. This can cause orphaned virtual machines. Wait until the entire environment is upgraded.

6.3. UPGRADING TEST COMPUTE NODES

This process upgrades Compute nodes selected for testing. The process involves running the **`openstack overcloud upgrade run`** command and including the **`--nodes`** option to restrict operations to the test nodes only. This procedure uses **`--nodes compute-0`** as an example in commands.

Procedure

1. Source the **`stackrc`** file:

```
$ source ~/stackrc
```

2. Run the upgrade command:

```
$ openstack overcloud upgrade run --nodes compute-0 --skip-tags
validation
```

NOTE

The command uses **`--skip-tags validation`** because OpenStack Platform services are inactive on the overcloud and cannot be validated.

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

3. Wait until the test node upgrade completes.

6.4. 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 --skip-tags
```

```
validation
```

NOTE

The command uses **`--skip-tags validation`** because OpenStack Platform services are inactive on the overcloud and cannot be validated.

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

3. Wait until the Compute node upgrade completes.

At this stage:

- Workloads are still running
- The Controller nodes and Compute nodes have been upgraded to OpenStack Platform 13
- Ceph Storage nodes are still at version 2 and require an upgrade

6.5. 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 --skip-tags
validation
```

NOTE

The command uses **`--skip-tags validation`** because OpenStack Platform services are inactive on the overcloud and cannot be validated.

- 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 <ENVIRONMENT FILE> \
    -e /home/stack/templates/overcloud_images.yaml \
    -e /home/stack/templates/deprecated_cli_options.yaml \
    -e /home/stack/templates/custom_repositories_script.yaml
    -e /usr/share/openstack-tripleo-heat-
templates/environments/ceph-ansible/ceph-ansible.yaml \
    -e /home/stack/templates/ceph-customization.yaml \
    --ceph-ansible-playbook '/usr/share/ceph-ansible/infrastructure-
playbooks/switch-from-non-containerized-to-containerized-ceph-
daemons.yml,/usr/share/ceph-ansible/infrastructure-
playbooks/rolling_update.yml'
```

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.
  - If applicable, an environment file that maps deprecated CLI options to Heat parameters using **deprecated_cli_options.yaml**.
  - If applicable, an environment file with your custom repository script using **custom_repositories_script.yaml**.
  - 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**.
- The following ansible playbooks:
- **/usr/share/ceph-ansible/infrastructure-playbooks/switch-from-non-containerized-to-containerized-ceph-daemons.yml**
- **/usr/share/ceph-ansible/infrastructure-playbooks/rolling_update.yml**

5. Wait until the Ceph Storage node upgrade completes.

At this stage:

- All nodes have been upgraded to OpenStack Platform 13 and workloads are still running

Although the environment is now upgraded, you must perform one last step to finalize the upgrade.

6.6. FINALIZING THE FAST FORWARD UPGRADE

The fast forward 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:

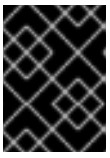
```
$ source ~/stackrc
```

2. Run the fast forward upgrade finalization command:

```
$ openstack overcloud ffwd-upgrade converge \
    --templates \
    -e /home/stack/templates/overcloud_images.yaml \
    -e /home/stack/templates/deprecated_cli_options.yaml \
    -e /home/stack/templates/custom_repositories_script.yaml \
    -e /usr/share/openstack-tripleo-heat-
templates/environments/ceph-ansible/ceph-ansible.yaml \
    -e /home/stack/templates/ceph-customization.yaml \
    -e <OTHER ENVIRONMENT FILES>
```

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.
  - If applicable, an environment file that maps deprecated CLI options to Heat parameters using **deprecated_cli_options.yaml**.
  - If applicable, an environment file with your custom repository script using **custom_repositories_script.yaml**.
  - If using Ceph Storage nodes, the relevant environment files.
  - 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**.



IMPORTANT

A prompt will ask if you are sure you want to perform the **ffwd-upgrade** command. Enter **yes**.

3. Wait until the fast forward upgrade finalization completes.

6.7. NEXT STEPS

The overcloud upgrade is complete. You can now perform any relevant post-upgrade overcloud configuration using the steps in Chapter 7, *Executing Post Upgrade Steps*. For future deployment operations, make sure to include all environment files relevant to your OpenStack Platform 13 environment, including new environment files created or converted during the upgrade.

CHAPTER 7. EXECUTING POST UPGRADE STEPS

This process implements final steps after completing the main upgrade process. This includes changing images and any additional configuration steps or considerations after the fast forward upgrade process completes.

7.1. 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 Reqirements" 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

7.2. 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.

7.3. 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.

7.4. TESTING A DEPLOYMENT

Although the overcloud has been upgraded, it is recommended to run a test deployment to ensure successful deployment operations in the future.

Procedure

1. Source the **`stackrc`** file:

```
$ source ~/stackrc
```

2. Run the deploy command and include all environment files relevant to your overcloud:

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

Include the following options relevant to your environment:

- Custom configuration environment files using **`-e`**.
- If applicable, your custom roles (**`roles_data`**) file using **`--roles-file`**.

3. Wait until the deployment completes.

7.5. CONCLUSION

This concludes the fast forward upgrade process.

APPENDIX A. RESTORING THE UNDERCLOUD

The following restore procedure assumes your undercloud node has failed and is in an unrecoverable state. This procedure involves restoring the database and critical filesystems on a fresh installation. It assumes the following:

- You have re-installed the latest version of Red Hat Enterprise Linux 7.
- The hardware layout is the same.
- The hostname and undercloud settings of the machine are the same.
- The backup archive has been copied to the **root** directory.

Procedure

1. Log into your undercloud as the **root** user.
2. Create the **stack** user:

```
[root@director ~]# useradd stack
```

3. Set a password for the user:

```
[root@director ~]# passwd stack
```

4. Disable password requirements when using **sudo**:

```
[root@director ~]# echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/stack
[root@director ~]# chmod 0440 /etc/sudoers.d/stack
```

5. Register your system with the Content Delivery Network, entering your Customer Portal user name and password when prompted:

```
[root@director ~]# subscription-manager register
```

6. Attach the Red Hat OpenStack Platform entitlement:

```
[root@director ~]# subscription-manager attach --pool=Valid-Pool-
Number-123456
```

7. Disable all default repositories, and then enable the required Red Hat Enterprise Linux repositories:

```
[root@director ~]# subscription-manager repos --disable=*
[root@director ~]# subscription-manager repos --enable=rhel-7-
server-rpms --enable=rhel-7-server-extras-rpms --enable=rhel-7-
server-rh-common-rpms --enable=rhel-ha-for-rhel-7-server-rpms --
enable=rhel-7-server-openstack-13-rpms
```

8. Perform an update on your system to make sure you have the latest base system packages:

```
[root@director ~]# yum update -y
[root@director ~]# reboot
```

9. Ensure the time on your undercloud is synchronized. For example:

```
[root@director ~]# yum install -y ntp
[root@director ~]# systemctl start ntpd
[root@director ~]# systemctl enable ntpd
[root@director ~]# ntpdate pool.ntp.org
[root@director ~]# systemctl restart ntpd
```

10. Create a temporary directory for the backup

```
[root@director ~]# mkdir /var/tmp/undercloud_backup
```

11. Extract the filesystem backup archive into the temporary directory:

```
[root@director ~]# tar -xvf /root/undercloud-backup-[timestamp].tar
-C /var/tmp/undercloud_backup --xattrs || true
```

12. Install **rsync**:

```
[root@director ~]# yum -y install rsync
```

13. Synchronize the following directories with backup content:

```
[root@director ~]# rsync -a -X
/var/tmp/undercloud_backup/home/stack/ /home/stack
[root@director ~]# rsync -a -X
/var/tmp/undercloud_backup/etc/haproxy/ /etc/haproxy/
[root@director ~]# rsync -a -X
/var/tmp/undercloud_backup/etc/pki/instack-certs/ /etc/pki/instack-
certs/
[root@director ~]# mkdir -p /etc/puppet/hieradata/
[root@director ~]# rsync -a -X
/var/tmp/undercloud_backup/etc/puppet/hieradata/
/etc/puppet/hieradata/
[root@director ~]# rsync -a -X /var/tmp/undercloud_backup/srv/node/
/srv/node/
[root@director ~]# rsync -a -X
/var/tmp/undercloud_backup/var/lib/glance/ /var/lib/glance/
```

14. Install the **openstack-keystone** package and synchronize its configuration data:

```
[root@director ~]# yum -y install openstack-keystone
[root@director ~]# rsync -a /var/tmp/undercloud_backup/etc/keystone/
/etc/keystone/
```

15. Install the **policycoreutils-python** package:

```
[root@director ~]# yum -y install policycoreutils-python
```

16. If using SSL in the undercloud, refresh the CA certificates:

```
[root@director ~]# semanage fcontext -a -t etc_t "/etc/pki/instack-
certs(/.*)?"
[root@director ~]# restorecon -R /etc/pki/instack-certs
[root@director ~]# update-ca-trust extract
```

17. Install the database server and client tools:

```
[root@director ~]# yum install -y mariadb mariadb-server python-
tripleoclient
```

18. Start the database:

```
[root@director ~]# systemctl start mariadb
[root@director ~]# systemctl enable mariadb
```

19. Increase the allowed packets to accommodate the size of our database backup:

```
[root@director ~]# mysql -uroot -e"set global max_allowed_packet =
1073741824;"
```

20. Restore the database backup:

```
[root@director ~]# mysql -u root <
/var/tmp/undercloud_backup/root/undercloud-all-databases.sql
```

21. Restart Mariadb to refresh the permissions from the backup file:

```
[root@director ~]# systemctl restart mariadb
```

1. Get a list of old user permissions:

```
[root@director ~]# mysql -e 'select host, user, password from
mysql.user;'
```

2. Remove the old user permissions for each host listed. For example:

```
[root@director ~]# HOST="192.0.2.1"
[root@director ~]# USERS=$(mysql -Nse "select user from mysql.user
WHERE user != \"root\" and host = \"$HOST\";" | uniq | xargs)
[root@director ~]# for USER in $USERS ; do mysql -e "drop user
\"$USER\"@\"$HOST\"" || true ;done
[root@director ~]# mysql -e 'flush privileges'
```

3. Install the **openstack-glance** package and restore its file permissions:

```
[root@director ~]# yum install -y openstack-glance
[root@director ~]# chown -R glance: /var/lib/glance/images
```

4. Install the **openstack-swift** packages and restore its file permissions:

```
[root@director ~]# yum install -y openstack-swift
[root@director ~]# chown -R swift: /srv/node
```

5. Switch to the new **stack** user:

```
[root@director ~]# su - stack
[stack@director ~]$
```

6. Run the undercloud installation command. Ensure to run it in the **stack** user's home directory:

```
[stack@director ~]$ openstack undercloud install
```

7. Wait until the install completes. The undercloud automatically restores its connection to the overcloud. The nodes will continue to poll OpenStack Orchestration (heat) for pending tasks.

APPENDIX B. RESTORING THE OVERCLOUD

B.1. RESTORING THE OVERCLOUD CONTROL PLANE SERVICES

WARNING

This procedure is currently under assessment by Red Hat due to some known issues.

The following procedure restores backups of the overcloud databases and configuration. In this situation, it is recommended to open three terminal windows so that you can perform certain operations simultaneously on all three Controller nodes. It is also recommended to select a Controller node to perform high availability operations. This procedure refers to this Controller node as the *bootstrap Controller node*.

IMPORTANT

This procedure only restores control plane services. It does not include restore Compute node workloads nor data on Ceph Storage nodes.

Procedure

1. If you are restoring from a failed major version upgrade, you might need to reverse any **yum** transactions that occurred on all nodes. This involves the following on each node:

   a. Enable the repositories for previous versions. For example:

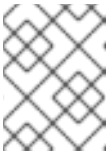
   ```
   # sudo subscription-manager repos --enable=rhel-7-server-
   openstack-10-rpms
   # sudo subscription-manager repos --enable=rhel-7-server-
   openstack-11-rpms
   # sudo subscription-manager repos --enable=rhel-7-server-
   openstack-12-rpms
   ```

   b. Check the **yum** history:

   ```
   # sudo yum history list all
   ```

   Identify transactions that occurred during the upgrade process. Most of these operations will have occurred on one of the Controller nodes (the Controller node selected as the bootstrap node during the upgrade). If you need to view a particular transaction, view it with the **history info** subcommand:

   ```
   # sudo yum history info 25
   ```



NOTE

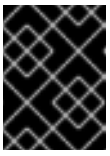
To force **yum history list all** to display the command ran from each transaction, set **history_list_view=commands** in your **yum.conf** file.

c. Revert any **yum** transactions that occurred since the upgrade. For example:

```
# sudo yum history undo 25
# sudo yum history undo 24
# sudo yum history undo 23
...
```

Make sure to start from the last transaction and continue in descending order. You can also revert multiple transactions in one execution using the **rollback** option. For example, the following command rolls back transaction from the last transaction to 23:

```
# sudo yum history rollback 23
```



IMPORTANT

It is recommended to use **undo** for each transaction instead of **rollback** so that you can verify the reversal of each transaction.

d. Once the relevant **yum** transaction have reversed, enable only the original OpenStack Platform repository on all nodes. For example:

```
# sudo subscription-manager repos --disable=rhel-7-server-
openstack-*-rpms
# sudo subscription-manager repos --enable=rhel-7-server-
openstack-10-rpms
```

2. Restore the database:

a. Copy the database backups to the bootstrap Controller node.

b. Stop connections to the database port on all Controller nodes:

```
# MYSQLIP=$(hiera mysql_bind_host)
# sudo /sbin/iptables -I INPUT -d $MYSQLIP -p tcp --dport 3306 -j
DROP
```

This isolates all the database traffic to the nodes.

c. On the bootstrap Controller node, disable pacemaker management of Galera:

```
# pcs resource unmanage galera
```

d. Comment out the **wsrep_cluster_address** parameter from **/etc/my.cnf.d/galera.cnf** on all Controller nodes.

```
# grep wsrep_cluster_address /etc/my.cnf.d/galera.cnf
# vi /etc/my.cnf.d/galera.cnf
```

e. Stop the MariaDB database on all the Controller nodes:

```
# mysqladmin -u root shutdown
```

NOTE

You might get a warning from HAProxy that the database is disabled.

f. Move existing MariaDB data directories and prepare new data directories on all Controller nodes,

```
# mv /var/lib/mysql/ /var/lib/mysql.old
# mkdir /var/lib/mysql
# chown mysql:mysql /var/lib/mysql
# chmod 0755 /var/lib/mysql
# mysql_install_db --datadir=/var/lib/mysql --user=mysql
# chown -R mysql:mysql /var/lib/mysql/
# restorecon -R /var/lib/mysql
```

g. Move the root configuration and cluster check to a backup file on all Controller nodes:

```
# sudo mv /root/.my.cnf /root/.my.cnf.old
# sudo mv /etc/sysconfig/clustercheck
/etc/sysconfig/clustercheck.old
```

h. On the bootstrap Controller node, set Pacemaker to manage the Galera cluster:

```
# pcs resource manage galera
# pcs resource cleanup galera
```

i. Wait for the Galera cluster to come up properly. Run the following command to see if all nodes are set as masters:

```
# watch "pcs status | grep -C3 galera"
Master/Slave Set: galera-master [galera]
Masters: [ overcloud-controller-0 overcloud-controller-1
overcloud-controller-2 ]
```

If the cleanup does not show all controller nodes as masters, run the cleanup command again:

```
# pcs resource cleanup galera
```

j. On the bootstrap Controller node, restore the OpenStack database. This will be replicated to the other Controller nodes by Galera:

```
# mysql -u root < openstack_database.sql
```

k. On the bootstrap controller node, restore the users and permissions:

```
# mysql -u root < grants.sql
```

l. On the bootstrap controller node, reset the database password to its original password:

```
# /usr/bin/mysqladmin -u root password "$(hiera
mysql::server::root_password)"
```

m. On the bootstrap controller node, run **pcs status** to show the Galera resource:

```
# pcs status | grep -C3 galera
```

The command might report an error because the database is now using the wrong username and password to connect and poll the database status. On all Controller nodes, restore the database configuration:

```
# sudo mv /root/.my.cnf.old /root/.my.cnf
# sudo mv /etc/sysconfig/clustercheck.old
/etc/sysconfig/clustercheck
```

n. Test the cluster check locally for each Controller node:

```
# /bin/clustercheck
```

o. On the bootstrap controller node, perform a cleanup in Pacemaker to reprobe the state of Galera:

```
# pcs resource cleanup galera
```

p. Test cluster check on each controller node:

```
# curl overcloud-controller-0:9200
# curl overcloud-controller-1:9200
# curl overcloud-controller-2:9200
```

q. Remove the firewall rule from each node for the services to restore access to the database:

```
# sudo /sbin/iptables -D INPUT -d $MYSQLIP -p tcp --dport 3306 -j
DROP
```

r. Uncomment the **wsrep_cluster_address** parameter in **/etc/my.cnf.d/galera.cnf** on all Controller nodes:

```
# vi /etc/my.cnf.d/galera.cnf
```

3. Restore the filesystem:

a. Copy the backup **tar** file for each Controller node to a temporary directory and uncompress all the data:

```
# mkdir /var/tmp/filesystem_backup/data/
# cd /var/tmp/filesystem_backup/data/
# mv <backup_file>.tar.gz .
# tar -xvzf --xattrs <backup_file>.tar.gz
```

NOTE

Do not extract directly to the `/` directory. This overrides your current filesystem. It is recommended to extract the file in a temporary directory.

b. Restore the **`/usr/libexec/os-apply-config/templates/etc/os-net-config/config.json`** file:

```
$ cp /var/tmp/filesystem_backup/data/usr/libexec/os-apply-config/templates/etc/os-net-config/config.json /usr/libexec/os-apply-config/templates/etc/os-net-config/config.json
```

c. Retain this directory in case you need any configuration files.

4. Cleanup the redis resource:

```
# pcs resource cleanup redis
```

5. After restoring the overcloud control plane data, check each relevant service is enabled and running correctly:

a. For high availability services on controller nodes:

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

b. For System services on controller and compute nodes:

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

The next few sections provide a reference of services that should be enabled.

B.2. RESTORED HIGH AVAILABILITY SERVICES

The following is a list of high availability services that should be active on OpenStack Platform 10 Controller nodes after a restore. If any of these service are disabled, use the following commands to enable them:

```
# pcs resource enable [SERVICE]
# pcs resource cleanup [SERVICE]
```

Controller Services
galera
haproxy
openstack-cinder-volume

Controller Services
rabbitmq
redis

B.3. RESTORED CONTROLLER SERVICES

The following is a list of core Systemd services that should be active on OpenStack Platform 10 Controller nodes after a restore. If any of these service are disabled, use the following commands to enable them:

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Controller Services
httpd
memcached
neutron-dhcp-agent
neutron-l3-agent
neutron-metadata-agent
neutron-openvswitch-agent
neutron-ovs-cleanup
neutron-server
ntpd
openstack-aodh-evaluator
openstack-aodh-listener
openstack-aodh-notifier
openstack-ceilometer-central
openstack-ceilometer-collector
openstack-ceilometer-notification

Controller Services
openstack-cinder-api
openstack-cinder-scheduler
openstack-glance-api
openstack-glance-registry
openstack-gnocchi-metricd
openstack-gnocchi-statsd
openstack-heat-api-cfn
openstack-heat-api-cloudwatch
openstack-heat-api
openstack-heat-engine
openstack-nova-api
openstack-nova-conductor
openstack-nova-consoleauth
openstack-nova-novncproxy
openstack-nova-scheduler
openstack-swift-account-auditor
openstack-swift-account-reaper
openstack-swift-account-replicator
openstack-swift-account
openstack-swift-container-auditor
openstack-swift-container-replicator
openstack-swift-container-updater

Controller Services
openstack-swift-container
openstack-swift-object-auditor
openstack-swift-object-expirer
openstack-swift-object-replicator
openstack-swift-object-updater
openstack-swift-object
openstack-swift-proxy
openvswitch
os-collect-config
ovs-delete-transient-ports
ovs-vswitchd
ovsdb-server
pacemaker

B.4. RESTORED OVERCLOUD COMPUTE SERVICES

The following is a list of core Systemd services that should be active on OpenStack Platform 10 Compute nodes after a restore. If any of these service are disabled, use the following commands to enable them:

```
# systemctl start [SERVICE]
# systemctl enable [SERVICE]
```

Compute Services
neutron-openvswitch-agent
neutron-ovs-cleanup
ntpd
openstack-ceilometer-compute

Compute Services
openstack-nova-compute
openvswitch
os-collect-config

APPENDIX C. SAMPLE YAML FILES FOR NFV UPDATE

These sample YAML files upgrade OVS for an OVS-DPDK deployment.

C.1. RED HAT OPENSTACK PLATFORM 10 OVS 2.9 UPDATE FILES

C.1.1. `post-install.yaml`

```
heat_template_version: 2014-10-16

description: >
  Example extra config for post-deployment

parameters:
  servers:
    type: json

resources:

  ExtraDeployments:
    type: OS::Heat::StructuredDeployments
    properties:
      servers:  {get_param: servers}
      config: {get_resource: ExtraConfig}
      actions: ['CREATE','UPDATE']

  ExtraConfig:
    type: OS::Heat::SoftwareConfig
    properties:
      group: script
      config: |
        #!/bin/bash
        set -x
        function tuned_service_dependency() {
          tuned_service=/usr/lib/systemd/system/tuned.service
          grep -q "network.target" $tuned_service
          if [ "$?" -eq 0 ]; then
              sed -i '/After=.*/s/network.target//g' $tuned_service
          fi
          grep -q "Before=.*network.target" $tuned_service
          if [ ! "$?" -eq 0 ]; then
              grep -q "Before=.*" $tuned_service
              if [ "$?" -eq 0 ]; then
                  sed -i 's/^\(Before=.*\)/\1 network.target
openvswitch.service/g' $tuned_service
              else
                  sed -i '/After/i Before=network.target
openvswitch.service' $tuned_service
              fi
          fi
        }

        if hiera -c /etc/puppet/hiera.yaml service_names | grep -q
```

```
neutron_ovs_dpdk_agent; then
            tuned_service_dependency
        fi
```