Red Hat OpenStack Platform 8

Upgrading Red Hat OpenStack Platform

Upgrading Red Hat OpenStack Platform

Last Updated: 2019-06-20

Red Hat OpenStack Platform 8 Upgrading Red Hat OpenStack Platform

Upgrading Red Hat OpenStack Platform

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Abstract

This document lays out the different methods through which users can upgrade from Red Hat OpenStack Platform 7 (Kilo) to 8 (Liberty). These methods assume that you will be upgrading to and from an OpenStack deployment installed on Red Hat Enterprise Linux 7.

Table of Contents

CHAPTER 1. INTRODUCTION

This document provides processes for keeping Red Hat OpenStack Platform up-to-date. This document focuses on upgrades and updates that targets **Red Hat OpenStack Platform 8 (Liberty)**.

Red Hat only supports upgrades to Red Hat OpenStack Platform 8 on Red Hat Enterprise Linux 7. In addition, Red Hat recommends the following different scenarios based on whether:

- You are using the director-based Overcloud or a manually created environment.
- You are using high availability tools to manage a set of Controller nodes in a cluster.

The Section 1.1, "Upgrade Scenario Comparison" provides descriptions of all upgrade scenarios. These scenarios allow you to upgrade to a working Red Hat OpenStack Platform 8 release and provide minor updates within that version.

1.1. UPGRADE SCENARIO COMPARISON

Red Hat recommends the following upgrade scenarios for Red Hat OpenStack Platform 8. The following table provides a brief description of each.

Table 1.1. Upgrade Scenarios

Method	Description
Director-Based Environments: Performing Updates to Minor Versions	This scenario is for updating from one minor version of Red Hat OpenStack Platform 8 to a newer version of Red Hat OpenStack Platform 8. This involves updating the director packages, then using the director to launch a package update on all nodes in the Overcloud.
Director-Based Environments: Performing Upgrades to Major Versions	This scenario is for upgrading from a major versions of Red Hat OpenStack Platform. In this case, the procedure upgrades from version 7 to version 8. This involves updating the director packages, then using the director to provide a set of upgrade scripts on each node, and then performing an upgrade of the Overcloud stack.
Non-Director Environments: Upgrading OpenStack Services Simultaneously	This scenario is for upgrading all packages in a Red Hat OpenStack Platform 8 environment that does not use the director for management (i.e. environments created manually). In this scenario, all packages are upgraded simultaneously.
Non-Director Environments: Upgrading Individual OpenStack Services (Live Compute) in a Standard Environment	This scenario is for upgrading all packages in a Red Hat OpenStack Platform environment 8 that does not use the director for management (i.e. environments created manually). In this scenario, you update each OpenStack service individually.

Method	Description
Non-Director Environments: Upgrading Individual OpenStack Services (Live Compute) in a High Availability Environment	This scenario is for upgrading all packages in a Red Hat OpenStack Platform 8 environment that does not use the director for management (i.e. environments created manually) and are using high availability tools for Controller-based OpenStack services. In this scenario, you update each OpenStack service individually.

For all methods:

- Ensure you have enabled the correct repositories for this release on all hosts.
- The upgrade will involve some service interruptions.
- Running instances will not be affected by the upgrade process unless you either reboot a Compute node or explicitly shut down an instance.

WARNING

Red Hat does not support upgrading any Beta release of Red Hat OpenStack Platform to any supported release.

1.2. REPOSITORY REQUIREMENTS

Both the undercloud and overcloud require access to Red Hat repositories either through the Red Hat Content Delivery Network, or through Red Hat Satellite 5 or 6. If using a Red Hat Satellite Server, synchronize the required repositories to your OpenStack Platform environment. Use the following list of CDN channel names as a guide:

Table 1.2. OpenStack Platform Repositories

Name	Repository	Description of Requirement
Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rpms**	Base operating system repository.
Red Hat Enterprise Linux 7 Server - Extras (RPMs)	**rhel-7-server-extras-rpms**	Contains Red Hat OpenStack Platform dependencies.
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	**rhel-7-server-rh-common-rpms**	Contains tools for deploying and configuring Red Hat OpenStack Platform.

Red Hat Satellite Tools for RHEL 7 Server RPMs x86_64	**rhel-7-server-satellite-tools-6.1-rpms**	Tools for managing hosts with Red Hat Satellite 6.
Red Hat Enterprise Linux High Availability (for RHEL 7 Server) (RPMs)	**rhel-ha-for-rhel-7-server-rpms**	High availability tools for Red Hat Enterprise Linux. Used for Controller node high availability.
Red Hat OpenStack Platform 8 director for RHEL 7 (RPMs)	**rhel-7-server-openstack-8-director-rpms**	Red Hat OpenStack Platform director repository. Also provides some tools for use on the director-deployed Overclouds.
Red Hat OpenStack Platform 8 for RHEL 7 (RPMs)	**rhel-7-server-openstack-8-rpms**	Core Red Hat OpenStack Platform repository.

If you have a Ceph cluster, the following repositories are also required:

Red Hat Ceph Storage OSD 1.3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-1.3-osd-rpms**	(For Ceph Storage Nodes) Repository for Ceph Storage Object Storage daemon. Installed on Ceph Storage nodes.
Red Hat Ceph Storage MON 1.3 for Red Hat Enterprise Linux 7 Server (RPMs)	**rhel-7-server-rhceph-1.3-mon-rpms**	(For Ceph Storage Nodes) Repository for Ceph Storage Monitor daemon. Installed on Controller nodes in OpenStack environments using Ceph Storage nodes.

NOTE

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

CHAPTER 2. DIRECTOR-BASED ENVIRONMENTS: PERFORMING UPDATES TO MINOR VERSIONS

This section explores how to update packages for your Red Hat OpenStack Platform environment within the same version. In this case, it is upgrades within Red Hat OpenStack Platform 8. This includes updating aspects of both the Undercloud and Overcloud.

WARNING

With High Availaibility for Compute instances (or Instance HA, as described in High Availability for Compute Instances), upgrades or scale-up operations are not possible. Any attempts to do so will fail.

If you have Instance HA enabled, disable it before performing an upgrade or scale-up. To do so, perform a *rollback* as described in Rollback.

This procedure for both situations involves the following workflow:

1. Update the Red Hat OpenStack Platform director packages
2. Update the Overcloud images on the Red Hat OpenStack Platform director
3. Update the Overcloud packages using the Red Hat OpenStack Platform director

2.1. UPDATING THE DIRECTOR PACKAGES

The director relies on standard RPM methods to update your environment. This involves ensuring your director's host uses the latest packages through **yum**.

1. Log into the director 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 update.

3. Update the **python-tripleoclient** package and its dependencies to ensure you have the latest scripts for the minor version update:

```
$ yum update python-tripleoclient
```

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

```
$ openstack undercloud upgrade
```

5. Check all services are active on the undercloud:

```
# sudo systemctl -t service
```

6. Verify the existence of your Overcloud and its nodes:

```
$ source ~/stackrc
$ openstack server list
$ openstack baremetal node list
$ openstack stack list
```

2.2. UPDATING THE OVERCLOUD IMAGES

The Undercloud update process might download new image archives from the **rhosp-director-images** and **rhosp-director-images-ipa** packages. Check the **yum** log to determine if new image archives are available:

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

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/*
```

Copy the new image archives:

```
$ cp /usr/share/rhosp-director-images/overcloud-full-latest-8.0.tar ~/images/.
$ cp /usr/share/rhosp-director-images/ironic-python-agent-latest-8.0.tar ~/images/.
```

Extract the archives:

```
$ cd ~/images
$ for tarfile in *.tar; do tar -xf $tarfile; done
```

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

```
$ openstack overcloud image upload --update-existing --image-path ~/images/.
$ openstack baremetal configure boot
```

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

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

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

2.3. UPDATING THE CONFIGURATION AGENT

IMPORTANT

This section is only required if updating from the Red Hat OpenStack Platform 8.0 initial release. If you performed this step in a previous update, you can ignore it.

Due to a known issue (see BZ#1278181), the Overcloud requires some manual configuration of its configuration agent. This involves copying a new version of the configuration agent script from the director to each node in the Overcloud.

Log in as the **stack** user on the director host and source the Undercloud configuration:

```
$ source ~/stackrc
```

Copy the configuration agent (**55-heat-config**) to each Overcloud node. Use the following command to do this for all hosts:

```
$ for i in $(nova list | awk '/Running/ {print $(NF-1)}' | awk -F"=" '{print $NF}' ) ; do echo $i ; scp -o StrictHostKeyChecking=no /usr/share/openstack-heat-templates/software-config/elements/heat-config/os-refresh-config/configure.d/55-heat-config heat-admin@${i}: ; ssh -o StrictHostKeyChecking=no heat-admin@${i} 'sudo /bin/bash -c "cp /home/heat-admin/55-heat-config /usr/libexec/os-refresh-config/configure.d/55-heat-config"' ; done
```

This ensures the configuration agent is up-to-date.

This Overcloud also needs to recreate some post-deployment files. The director includes a script to achieve this. Copy and execute the **heat-config-rebuild-deployed** script on each node. Use the following command to do this for all nodes:

```
$ for i in $(nova list | awk '/Running/ {print $(NF-1)}' | awk -F"=" '{print $NF}') ; do echo $i ; scp  -o StrictHostKeyChecking=no /usr/share/openstack-heat-templates/software-config/elements/heat-config/bin/heat-config-rebuild-deployed heat-admin@${i}: ; ssh -o StrictHostKeyChecking=no heat-admin@${i} 'sudo /bin/bash -c "mkdir -p /usr/share/openstack-heat-templates/software-config/elements/heat-config/bin ; cp heat-config-rebuild-deployed /usr/share/openstack-heat-templates/software-config/elements/heat-config/bin/heat-config-rebuild-deployed ; chmod +x /usr/share/openstack-heat-templates/software-config/elements/heat-config/bin/heat-config-rebuild-deployed ; /usr/share/openstack-heat-templates/software-config/elements/heat-config/bin/heat-config-rebuild-deployed"' ; done
```

2.4. UPDATING THE OVERCLOUD PACKAGES

The Overcloud relies on standard RPM methods to update the environment. This involves performing an update on all nodes using the **openstack overcloud update** from the director.

Use the **-e** to include environment files relevant to your Overcloud and its upgrade 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:

- The **overcloud-resource-registry-puppet.yaml** file from the heat template collection. Although this file is included automatically when you run the **openstack overcloud deploy** command, **you must include this file when you run the openstack overcloud update command**.
- 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.

Running an update on all nodes in parallel might cause problems. For example, an update of a package might involve restarting a service, which can disrupt other nodes. This is why the update process updates each node using a set of breakpoints. This means nodes are updated one by one. When one node completes the package update, the update process moves on to the next node. The update process also requires the **-i** option, which puts the command in an interactive mode that requires confirmation at each breakpoint. Without the **-i** option, the update remains paused at the first breakpoint.

The following is an example update command for updating the Overcloud:

```
$ openstack overcloud update stack overcloud -i \
  --templates  \
  -e /usr/share/openstack-tripleo-heat-templates/overcloud-resource-registry-puppet.yaml \
  -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>|...]
```

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

```
not_started: [u'overcloud-controller-0', u'overcloud-controller-1', u'overcloud-controller-2']
on_breakpoint: [u'overcloud-compute-0']
Breakpoint reached, continue? Regexp or Enter=proceed, 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. You can also type a node name to clear a breakpoint on a specific node, or a Python-based regular expression to clear breakpoints on multiple nodes at once. However, it is not recommended to clear breakpoints on multiple controller nodes at once. Continue this process until all nodes have completed their update.

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

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

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
```

IMPORTANT

The update process does not reboot any nodes in the Overcloud automatically. If required, perform a reboot manually after the update completes.

CHAPTER 3. DIRECTOR-BASED ENVIRONMENTS: PERFORMING UPGRADES TO MAJOR VERSIONS

WARNING

Before performing an upgrade to the latest major version, ensure the undercloud and overcloud are updated to the latest minor versions. This includes both OpenStack Platform services and the base operating system. For the process on performing a minor version update, see "Updating the Environment" in the Red Hat OpenStack Platform 7 *Director Installation and Usage* guide. Performing a major version upgrade without first performing a minor version update can cause failures in the upgrade process.

WARNING

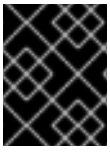
With High Availaibility for Compute instances (or Instance HA, as described in High Availability for Compute Instances), upgrades or scale-up operations are not possible. Any attempts to do so will fail.

If you have Instance HA enabled, disable it before performing an upgrade or scale-up. To do so, perform a *rollback* as described in Rollback.

This chapter explores how to upgrade your environment. This includes upgrading aspects of both the Undercloud and Overcloud. This upgrade process provides a means for you to move to the next major version. In this case, it is a upgrade from Red Hat OpenStack Platform 7 to Red Hat OpenStack Platform 8.

This procedure for both situations involves the following workflow:

1. Update the Red Hat OpenStack Platform director packages
2. Update the Overcloud images on the Red Hat OpenStack Platform director
3. Update the Overcloud stack and its packages using the Red Hat OpenStack Platform director



IMPORTANT

Make sure to read the information in Section 3.1, “Important Pre-Upgrade Notes” before attempting a version upgrade.

3.1. IMPORTANT PRE-UPGRADE NOTES

Make sure you have read the following notes before upgrading your environment.

- Upgrade in Red Hat OpenStack Platform director requires full testing with specific configurations before performed on any live production environment. Red Hat has tested most use cases and combinations that are offered as standard options through the director but, due to the number of possible combinations, this can never be a fully exhaustive list. In addition, if the configuration has been modified from the standard deployment, either manually or through post configuration hooks, testing the upgrade feature in a non-production environment becomes even more critical. Therefore, we advise you to:
  - Perform a backup of your Undercloud node before starting any steps in the upgrade procedure. See the Back Up and Restore Red Hat OpenStack Platform guide for backup procedures.
  - Run the upgrade procedure in a test environment that includes all of the changes made before running the procedure in your production environment.
  - Please contact Red Hat and request guidance and assistance on the upgrade process before proceeding if you feel uncomfortable about performing this upgrade.
- The upgrade process outlined in this section only accommodates customizations through the director. If you customized an Overcloud feature outside of director then disable the feature, upgrade the Overcloud, and re-enable the feature after the upgrade completes. This means the customized feature is unavailable until the completion of the entire upgrade.
- Red Hat OpenStack Platform director 8 can manage pervious Overcloud versions. See the support matrix below for information.

Table 3.1. Support Matrix for Red Hat OpenStack Platform director 8

Version	Overcloud Updating	Overcloud Deploying	Overcloud Scaling
Red Hat OpenStack Platform 7	7.0.4 and newer	7.0.4 and newer	7.0.4 and newer
Red Hat OpenStack Platform 8	All versions	All versions	All versions

- If using managing and older Overcloud version, use the following Heat template collections:
  - For Red Hat OpenStack Platform 7: **/usr/share/openstack-tripleo-heat-templates/kilo/** For example:

```
$ openstack overcloud deploy -templates /usr/share/openstack-tripleo-heat-templates/kilo/ [OTHER_OPTIONS]
```

- If managing a Red Hat OpenStack Platform 7 Overcloud, set the RabbitMQ password to the version 7 default in the **/home/stack/tripleo-overcloud-passwords** file:

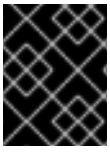
```
OVERCLOUD_RABBITMQ_PASSWORD=guest
```

- If using an environment file for Satellite registration, make sure to update the following parameters in the environment file:

- **rhel_reg_repos** - Repositories to enable for your Overcloud, including the new Red Hat OpenStack Platform 8 repositories. See Section 1.2, "Repository Requirements" for repositories to enable.
- **rhel_reg_activation_key** - The new activation key for your Red Hat OpenStack Platform 8 repositories.
- **rhel_reg_sat_repo** - A new parameter that defines the repository containing Red Hat Satellite 6's management tools, such as **katello-agent**. Make sure to add this parameter if registering to Red Hat Satellite 6.

- The default timezone for Red Hat OpenStack Platform 8 is now UTC. The default timezone for Red Hat OpenStack Platform 7 was EST. If necessary, include an environment file to specify the timezone.
- If using an external load balancer, update your load balancing settings to accommodate new services in Red Hat OpenStack Platform 8. For a full list of services and example configuration, see "Services Configuration Reference" in the *External Load Balancing for the Overcloud* guide.
- Make sure that you have upgraded your undercloud and overcloud to the latest minor release of Red Hat OpenStack Platform 7 and Red Hat Enterprise Linux 7 before attempting a major upgrade to Red Hat OpenStack Platform 8. See "Updating the Environment" in the Red Hat OpenStack Platform 7 *Director Installation and Usage* guide for instructions on performing a package update to your undercloud and overcloud. If the kernel updates to the latest version, perform a reboot so that new kernel parameters take effect.
- Apply a version lock to **libvirt** as described in *Solutions*.

3.2. UPGRADING THE DIRECTOR



IMPORTANT

Make sure to read the information in Section 3.1, "Important Pre-Upgrade Notes" before attempting any step in the following procedure.

Log into the director as the **stack** user and 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 update.

To update the director packages to the latest major version, change the OpenStack Platform repository from the old version to the new version. For example:

```
$ sudo subscription-manager repos --disable=rhel-7-server-openstack-7.0-rpms --disable=rhel-7-server-openstack-7.0-director-rpms
$ sudo subscription-manager repos --enable=rhel-7-server-openstack-8-rpms --enable=rhel-7-server-openstack-8-director-rpms
```

This sets **yum** to use the latest repositories. Use **yum** to update the director:

```
$ sudo yum upgrade
```

Some OpenStack services might fail after **yum update** completes. This is expected behavior. The Undercloud's upgrade command corrects the configuration of these services.

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

```
$ openstack undercloud upgrade
```

This refreshes the director's configuration and populates any settings that are unset since the version change. Running this command does not delete any stored data, such Overcloud stack data or data for existing nodes in your environment.

Once the update completes, check the director's OpenStack services:

```
$ sudo systemctl list-units openstack-*
```

NOTE

The **openstack-keystone** might appear as a failed service. This is because the service now runs through the **httpd** service as a WSGI application. The **openstack-keystone** service is safe to disable after updating director packages and running **openstack undercloud upgrade**.

To finalize the update, verify the existence of your Overcloud and its nodes:

```
$ source ~/stackrc
$ openstack server list
$ ironic node-list
$ heat stack-list
```

Be aware of the following notes after upgrading the Overcloud to Red Hat OpenStack Platform 8:

- Underclouds using SSL might experience a loss of access to VIPs during the upgrade. If so, restart the **keepalived** service on the Undercloud:

  ```
  $ systemctl restart keepalived
  ```

- The Undercloud's **admin** user might require an additional role (**_member_**) not included with Red Hat OpenStack Platform 8. This role is important for Overcloud communication. Check for this role:

  ```
  $ keystone role-list
  ```

  If the role does not exist, create it:

  ```
  $ keystone role-create --name _member_
  ```

  Add the role to the **admin** user on the **admin** tenant:

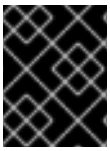
  ```
  $ keystone user-role-add --user admin --role _member_ --tenant admin
  ```

- If using customized core Heat templates, make sure to check for differences between the updated core Heat templates and your current set. Red Hat provides updates to the Heat template collection over subsequent releases. Using a modified template collection can lead to a divergence between your custom copy and the original copy in **/usr/share/openstack-tripleo-heat-templates**. Run the following command to see differences between your custom Heat template collection and the updated original version:

  ```
  # diff -Nary /usr/share/openstack-tripleo-heat-templates/ ~/templates/my-overcloud/
  ```

  Make sure to either apply these updates to your custom Heat template collection, or create a new copy of the templates in **/usr/share/openstack-tripleo-heat-templates/** and apply your customizations.

3.3. UPGRADING THE OVERCLOUD IMAGES ON THE DIRECTOR



IMPORTANT

Make sure to read the information in Section 3.1, "Important Pre-Upgrade Notes" before attempting any step in the following procedure.

This procedure ensures you have the latest images for node discovery and Overcloud deployment. The new images from the **rhosp-director-images** and **rhosp-director-images-ipa** packages are already updated from the Undercloud upgrade.

Remove any existing images from the **images** directory on the **stack** user's home (**/home/stack/images**), then copy the new image archives:

```
$ rm -rf ~/images/*
$ cp /usr/share/rhosp-director-images/overcloud-full-latest-8.0.tar ~/images/.
$ cp /usr/share/rhosp-director-images/ironic-python-agent-latest-8.0.tar ~/images/.
```

Extract the archives:

```
$ cd ~/images
$ for tarfile in *.tar; do tar -xf $tarfile; done
```

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

```
$ openstack overcloud image upload --update-existing --image-path ~/images/.
$ openstack baremetal configure boot
```

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

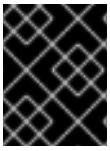
```
$ openstack image list
$ ls -l /httpboot
```

The director is now upgraded with the latest images.

IMPORTANT

Make sure the Overcloud image version corresponds to the Undercloud version.

3.4. UPGRADING THE OVERCLOUD



IMPORTANT

Make sure to read the information in Section 3.1, "Important Pre-Upgrade Notes" before attempting any step in the following procedure.

This section details the steps required to upgrade the Overcloud. Make sure to follow each section in order and only apply the sections relevant to your environment.

This process requires you to run your original **openstack overcloud deploy** command multiple times to provide a staged method of upgrading. Each time you run the command, you include a different upgrade environment file along with your existing environment files. These new upgrade environments files are:

- **major-upgrade-pacemaker-init.yaml** - Provides the initialization for the upgrade. This includes updating the Red Hat OpenStack Platform repositories on each node in your Overcloud and provides special upgrade scripts to certain nodes.
- **major-upgrade-pacemaker.yaml** - Provides an upgrade for the Controller nodes.
- **major-upgrade-pacemaker-converge.yaml** - The finalization for the Overcloud upgrade. This aligns the resulting upgrade to match the contents for the director's latest Heat template collection.

In between these deployment commands, you run the **upgrade-non-controller.sh** script on various node types. This script updates the packages on a non-Controller node.

Workflow

The Overcloud upgrade process uses the following workflow:

1. Run your deployment command including the **major-upgrade-pacemaker-init.yaml** environment file.
2. Run the **upgrade-non-controller.sh** on each Object Storage node.
3. Run your deployment command including the **major-upgrade-pacemaker.yaml** environment file.
4. Run the **upgrade-non-controller.sh** on each Compute node.
5. Run the **upgrade-non-controller.sh** on each Ceph Storage node.
6. Run your deployment command including the **major-upgrade-pacemaker-converge.yaml** environment file.

3.4.1. Including the Management Network

If using custom NIC templates from Red Hat OpenStack Platform 7, add the **ManagementSubnetIp** parameter to the **parameters** section of your NIC templates. For example:

```
parameters:
  ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
```

```
    description: IP address/subnet on the management network
    type: string
```

3.4.2. Installing the Upgrade Scripts

This step installs scripts on each non-Controller node. These script perform the major version package upgrades and configuration. Each script differs depending on the node type. For example, Compute nodes receive different upgrade scripts to Ceph Storage nodes.

This initialization step also updates enabled repositories on all overcloud nodes. This means you do not need to disable old repositories and enable new repositories manually.

Run the **openstack overcloud deploy** from your Undercloud and include the **major-upgrade-pacemaker-init.yaml** environment file. Make sure you also include all custom environment files relevant to your environment, such as network isolation and storage.

This following is an example of the **openstack overcloud deploy** command with the added file. For example:

```
$ openstack overcloud deploy --templates \
  --control-scale 3 \
  --compute-scale 3 \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml  \
  -e /home/stack/templates/network_env.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/major-upgrade-pacemaker-init.yaml
```

Wait until the Overcloud updates with the new environment file's configuration.

3.4.3. Upgrading Object Storage Nodes

The director uses the **upgrade-non-controller.sh** command to run the upgrade script passed to each non-Controller node from the **major-upgrade-pacemaker-init.yaml** environment file. For this step, you upgrade each Object Storage node with the following command:

```
$ for NODE in `nova list | grep swift | awk -F "|" '{ print $2 }'` ; do upgrade-non-controller.sh --upgrade
$NODE ; done
```

Wait until each Object Storage node completes its upgrade.

3.4.4. Upgrading Controller Nodes

Upgrading the Controller nodes involves including another environment file (**major-upgrade-pacemaker.yaml**) that provides a full upgrade to Controller nodes running high availability tools.

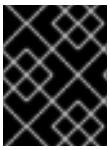
Run the **openstack overcloud deploy** from your Undercloud and include the **major-upgrade-pacemaker.yaml** environment file. Make sure you also include all custom environment files relevant to your environment, such as network isolation and storage.

This following is an example of the **openstack overcloud deploy** command with the added file. For example:

```
$ openstack overcloud deploy --templates \
  --control-scale 3 \
```

```
--compute-scale 3 \
-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml \
-e network_env.yaml \
-e /usr/share/openstack-tripleo-heat-templates/environments/major-upgrade-pacemaker.yaml
```

Wait until the Overcloud updates with the new environment file's configuration.



IMPORTANT

This step disables the Neutron server and L3 Agent during the Controller upgrade. This means floating IP address are unavailable during this step.

IMPORTANT

If the Overcloud stack fails during this step, log into one of your Controller nodes, run **sudo pcs cluster start**, then rerun **openstack overcloud deploy** on the director.

3.4.5. Upgrading Compute Nodes

The director uses the **upgrade-non-controller.sh** command to run the upgrade script passed to each non-Controller node from the **major-upgrade-pacemaker-init.yaml** environment file.

Obtain a list of Compute nodes and their UUIDs:

```
$ nova list | grep "compute"
```

We perform the following task on each node individually to ensure zero downtime. Select a Compute node to upgrade.

1. Migrate its workload. See "Migrating VMs from an Overcloud Compute Node" in the *Red Hat OpenStack Platform Director Installation and Usage Guide*.

2. Ensure the Compute service for the chosen node is disabled:

   ```
   $ source ~/overcloudrc
   $ nova service-list
   $ nova service-disable [hostname] nova-compute
   ```

3. Upgrade each Compute node with the following command:

   ```
   $ source ~/stackrc
   $ upgrade-non-controller.sh --upgrade NODE_UUID
   ```

   Replace *NODE_UUID* with the UUID of the chosen Compute node

4. Wait until the Compute node completes its upgrade. After the upgrade completes, make sure the Compute nodes are enabled using the following command:

   ```
   $ source ~/overcloudrc
   $ nova service-list
   $ nova service-enable [hostname] nova-compute
   ```

Repeat these steps on each Compute node.

3.4.6. Upgrading Ceph Storage Nodes

The director uses the **upgrade-non-controller.sh** command to run the upgrade script passed to each non-Controller node from the **major-upgrade-pacemaker-init.yaml** environment file. For this step, you upgrade each Ceph Storage node with the following command:

Upgrade each Ceph Storage nodes:

```
$ for NODE in `nova list | grep ceph | awk -F "|" '{ print $2 }'` ; do upgrade-non-controller.sh --upgrade $NODE ; done
```

3.4.7. Finalizing the Upgrade

The director needs to run through the upgrade finalization to ensure the Overcloud stack is synchronized with the current Heat template collection. This involves an environment file (**major-upgrade-pacemaker-converge.yaml**), which you include using the **openstack overcloud deploy** command.

Run the **openstack overcloud deploy** from your Undercloud and include the **major-upgrade-pacemaker-converge.yaml** environment file. Make sure you also include all custom environment files relevant to your environment, such as network isolation and storage.

This following is an example of the **openstack overcloud deploy** command with the added file. For example:

```
$ openstack overcloud deploy --templates \
  --control-scale 3 \
  --compute-scale 3 \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml \
  -e network_env.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/major-upgrade-pacemaker-converge.yaml
```

Wait until the Overcloud updates with the new environment file's configuration.

This completes the Overcloud upgrade procedure.

3.4.8. Post-Upgrade Notes

Be aware of the following notes after upgrading the Overcloud to Red Hat OpenStack Platform 8:

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

  ```
  $ sudo systemctl restart neutron-openvswitch-agent
  ```

- The update process does not reboot any nodes in the Overcloud automatically. If required, perform a reboot manually after the update command completes. Make sure to reboot cluster-based nodes (such as Ceph Storage nodes and Controller nodes) individually and wait for the node to rejoin the cluster. For Ceph Storage nodes, check with the **ceph health** and make sure the cluster status is **HEALTH OK**. For Controller nodes, check with the **pcs resource** and make sure all resources are running for each node.

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

```
$ sudo systemctl restart corosync
```

- 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
```

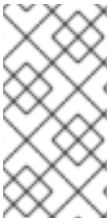
- The next time you update or scale the Overcloud stack (i.e. running the **openstack overcloud deploy** command), you need to reset the identifier that triggers package updates in the Overcloud. Add a blank **UpdateIdentifier** parameter to an environment file and include it when you run the **openstack overcloud deploy** command. The following is an example of such an environment file:

```
parameter_defaults:
  UpdateIdentifier:
```

CHAPTER 4. NON-DIRECTOR ENVIRONMENTS: UPGRADING OPENSTACK SERVICES SIMULTANEOUSLY

This scenario upgrades from Red Hat OpenStack Platform 7 to Red Hat OpenStack Platform 8 in environments that **do not use the director**. This procedure upgrades all services on all nodes. This involves the following workflow:

1. Disabling all OpenStack services
2. Performing a package upgrade
3. Performing synchronization of all databases
4. Enabling all OpenStack services



NOTE

The procedures in this chapter follow the architectural naming convention followed by all Red Hat OpenStack Platform documentation. If you are unfamiliar with this convention, refer to Architecture Guide available at: Red Hat OpenStack Platform Documentation Suite before proceeding.

4.1. DISABLING ALL OPENSTACK SERVICES

The first step to performing a complete upgrade of Red Hat OpenStack Platform on a node involves shutting down all Openstack services. This step differs based on whether the node OpenStack uses high availability tools for management (e.g. using Pacemaker on Controller nodes). This step contains instructions for both node types.

Standard Nodes

Install the **openstack-utils** package on all standard nodes:

```
# yum install openstack-utils
```

Disable all OpenStack services on all standard nodes.

```
# openstack-service stop
```

High Availability Nodes

We need to disable all OpenStack services but leave the database and load balancing services active. For example, switch the HAProxy, Galera, and MongoDB services to unmanaged in Pacemaker:

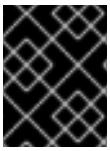
```
# pcs resource unmanage haproxy
# pcs resource unmanage galera
# pcs resource unmanage mongod
```

Disable the remaining Pacemaker-managed resources by setting the **stop-all-resources** property on the cluster. Run the following on a single member of your Pacemaker cluster:

```
# pcs property set stop-all-resources=true
```

Wait until all Pacemaker-managed resources have stopped. Run the **pcs status** command to see the status of each resources.

```
# pcs status
```



IMPORTANT

HAProxy might show a broadcast message for unavailable services. This is normal behavior.

4.2. PERFORMING A PACKAGE UPGRADE

The next step upgrades all packages on a node. Perform this step on each node with OpenStack services.

Change to the Red Hat OpenStack Platform 8 repository using the **subscription-manager** command.

```
# subscription-manager repos --disable=rhel-7-server-openstack-7.0-rpms
# subscription-manager repos --enable=rhel-7-server-openstack-8-rpms
```

Run the **yum update** command on the node:

```
# yum update
```

Wait until the package upgrade completes.

Review the resulting configuration files. The upgraded packages will have installed **.rpmnew** files appropriate to the Red Hat OpenStack Platform 8 version of the service. New versions of OpenStack services may deprecate certain configuration options. You should also review your OpenStack logs for any deprecation warnings, because these may cause problems during future upgrades. For more information on the new, updated and deprecated configuration options for each service, see Configuration Reference available from: Red Hat OpenStack Platform Documentation Suite .

Perform the package upgrade on each node in your environment.

4.3. PERFORMING SYNCHRONIZATION OF ALL DATABASES

The next step upgrades the database for each service.

NOTE

Flush expired tokens in the Identity service to decrease the time required to synchronize the database.

```
# keystone-manage token_flush
```

Upgrade the database schema for each service that uses the database. Run the following command on the node hosting the service's database:

```
# openstack-db --service SERVICENAME --update
```

Use the service's project name as the SERVICENAME. For example, to upgrade the database schema of the Identity service:

```
# openstack-db --service keystone --update
```

Table 4.1. Project Name of OpenStack services

Service	Project name
Identity	keystone
Image Service	glance
Block Storage	cinder
Orchestration	heat
Compute	nova
Networking	neutron

The Telemetry service uses a separate command for database upgrades:

```
# ceilometer-dbsync
```

4.4. ENABLING ALL OPENSTACK SERVICES

The final step enables the OpenStack services on the node. This step differs based on whether the node OpenStack uses high availability tools for management. For example, using Pacemaker on Controller nodes. This step contains instructions for both node types.

Standard Nodes

Enable all OpenStack services:

```
# openstack-service stop
```

High Availability Nodes

Restart your resources through Pacemaker. Reset the **stop-all-resources** property on a single member of your Pacemaker cluster. For example:

```
# pcs property set stop-all-resources=false
```

Wait until all resources have started. Run the **pcs status** command to see the status of each resources.

```
# pcs status
```

Enable Pacemaker management for any unmanaged resources, such as the databases and load balancer:

-

```
# pcs resource manage haproxy
# pcs resource manage galera
# pcs resource manage mongod
```

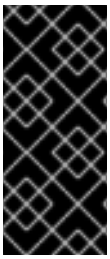
4.5. POST-UPGRADE NOTES

New versions of OpenStack services may deprecate certain configuration options. You should also review your OpenStack logs for any deprecation warnings, because these may cause problems during a future upgrade. For more information on the new, updated and deprecated configuration options for each service , see Configuration Reference available from: Red Hat OpenStack Platform Documentation Suite.

CHAPTER 5. NON-DIRECTOR ENVIRONMENTS: UPGRADING INDIVIDUAL OPENSTACK SERVICES (LIVE COMPUTE) IN A STANDARD ENVIRONMENT

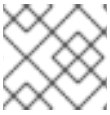
This section describes the steps you should follow to upgrade your cloud deployment by updating one service at a time with live compute in a non High Availability (HA) environment. This scenario upgrades from Red Hat OpenStack Platform 7 to Red Hat OpenStack Platform 8 in environments that **do not use the director**.

A live Compute upgrade minimizes interruptions to your Compute service, with only a few minutes for the smaller services, and a longer migration interval for the workloads moving to newly-upgraded Compute hosts. Existing workloads can run indefinitely, and you do not need to wait for a database migration.



IMPORTANT

Due to certain package dependencies, upgrading the packages for one OpenStack service might cause Python libraries to upgrade before other OpenStack services upgrade. This might cause certain services to fail prematurely. In this situation, continue upgrading the remaining services. All services should be operational upon completion of this scenario.



NOTE

This method may require additional hardware resources to bring up the Compute nodes.

NOTE

The procedures in this chapter follow the architectural naming convention followed by all Red Hat OpenStack Platform documentation. If you are unfamiliar with this convention, refer to the Architecture Guide available in the Red Hat OpenStack Platform Documentation Suite before proceeding.

5.1. PRE-UPGRADE TASKS

On each node, change to the Red Hat OpenStack Platform 8 repository using the **subscription-manager** command.

```
# subscription-manager repos --disable=rhel-7-server-openstack-7.0-rpms
# subscription-manager repos --enable=rhel-7-server-openstack-8-rpms
```

Upgrade the **openstack-selinux** package:

```
# yum upgrade openstack-selinux
```

This is necessary to ensure that the upgraded services will run correctly on a system with SELinux enabled.

5.2. UPGRADING IDENTITY (KEYSTONE) AND DASHBOARD (HORIZON)

Disable the Identity service and the Dashboard service. Depending on your configuration, this involves either:

1. Disabling **httpd** if both the Dashboard and the Identity service are running as WSGI applets:

```
# systemctl stop httpd
```

2. Disabling **openstack-keystone** for the Identity service if it is running as a separate service, then disabling **httpd** for the Dashboard:

```
# openstack-service stop keystone
# systemctl stop httpd
```

Update the packages for both services:

```
# yum -d1 -y upgrade \*keystone\*
# yum -y upgrade \*horizon\* \*openstack-dashboard\*
# yum -d1 -y upgrade \*horizon\* \*python-django\*
```

It is possible that the Identity service's token table has a large number of expired entries. This can dramatically increase the time it takes to complete the database schema upgrade. To flush expired tokens from the database and alleviate the problem, the **keystone-manage** command can be used before running the Identity database upgrade.

```
# keystone-manage token_flush
# openstack-db --service keystone --update
```

This flushes expired tokens from the database. You can arrange to run this command periodically using **cron**.

Restart the services. Depending on your configuration, this involves either:

1. Enabling **httpd** if both the Dashboard and the Identity service are running as WSGI applets:

```
# systemctl start httpd
```

2. Enabling **openstack-keystone** for the Identity service if it is running as a separate service, then disabling **httpd** for the Dashboard:

```
# openstack-service start keystone
# systemctl start httpd
```

5.3. UPGRADING OBJECT STORAGE (SWIFT)

On your Object Storage hosts, run:

```
# openstack-service stop swift
# yum -d1 -y upgrade \*swift\*
# openstack-service start swift
```

5.4. UPGRADING IMAGE SERVICE (GLANCE)

On your Image Service host, run:

```
# openstack-service stop glance
# yum -d1 -y upgrade \*glance\*
# openstack-db --service glance --update
# openstack-service start glance
```

5.5. UPGRADING BLOCK STORAGE (CINDER)

On your Block Storage host, run:

```
# openstack-service stop cinder
# yum -d1 -y upgrade \*cinder\*
# openstack-db --service cinder --update
# openstack-service start cinder
```

5.6. UPGRADING ORCHESTRATION (HEAT)

On your Orchestration host, run:

```
# openstack-service stop heat
# yum -d1 -y upgrade \*heat\*
# openstack-db --service heat --update
# openstack-service start heat
```

5.7. UPGRADING TELEMETRY (CEILOMETER)

1. On all nodes hosting Telemetry component services, run:

   ```
   # openstack-service stop ceilometer
   # yum -d1 -y upgrade \*ceilometer\*
   ```

2. On the controller node, where database is installed, run:

   ```
   # ceilometer-dbsync
   ```

3. After completing the package upgrade, restart the Telemetry service by running the following command on all nodes hosting Telemetry component services:

   ```
   # openstack-service start ceilometer
   ```

5.8. UPGRADING COMPUTE (NOVA)

1. If you are performing a rolling upgrade of your compute hosts you need to set explicit API version limits to ensure compatibility in your environment.
   Before starting Compute services on Controller or Compute nodes, set the **compute** option in the **[upgrade_levels]** section of **nova.conf** to the previous Red Hat OpenStack Platform version (**kilo**):

   ```
   # crudini --set /etc/nova/nova.conf upgrade_levels compute kilo
   ```

You need to make this change on your Controller and Compute nodes.

You should undo this operation after upgrading all of your Compute nodes.

2. On your Compute host, run:

```
# openstack-service stop nova
# yum -d1 -y upgrade \*nova\*
# openstack-db --service nova --update
```

3. After you have upgraded all of your hosts, you will want to remove the API limits configured in the previous step. On all of your hosts:

```
# crudini --del /etc/nova/nova.conf upgrade_levels compute
```

4. Restart the Compute service on all the Controller and Compute nodes:

```
# openstack-service start nova
```

5.9. UPGRADING OPENSTACK NETWORKING (NEUTRON)

1. On your OpenStack Networking host, run:

```
# openstack-service stop neutron
# yum -d1 -y upgrade \*neutron\*
# openstack-db --service neutron --update
```

2. Restart the OpenStack Networking service:

```
# openstack-service start neutron
```

5.10. POST-UPGRADE TASKS

After completing all of your individual service upgrades, you should perform a complete package upgrade on all of your systems:

```
# yum upgrade
```

This will ensure that all packages are up-to-date. You may want to schedule a restart of your OpenStack hosts at a future date in order to ensure that all running processes are using updated versions of the underlying binaries.

Review the resulting configuration files. The upgraded packages will have installed **.rpmnew** files appropriate to the Red Hat OpenStack Platform 8 version of the service.

New versions of OpenStack services may deprecate certain configuration options. You should also review your OpenStack logs for any deprecation warnings, because these may cause problems during a future upgrade. For more information on the new, updated and deprecated configuration options for each service , see Configuration Reference available from: Red Hat OpenStack Platform Documentation Suite.

CHAPTER 6. NON-DIRECTOR ENVIRONMENTS: UPGRADING INDIVIDUAL OPENSTACK SERVICES (LIVE COMPUTE) IN A HIGH AVAILABILITY ENVIRONMENT

This chapter describes the steps you should follow to upgrade your cloud deployment by updating one service at a time with live compute in a High Availability (HA) environment. This scenario upgrades from Red Hat OpenStack Platform 7 to Red Hat OpenStack Platform 8 in environments that **do not use the director**.

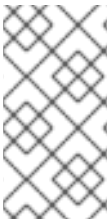
A live Compute upgrade minimizes interruptions to your Compute service, with only a few minutes for the smaller services, and a longer migration interval for the workloads moving to newly-upgraded Compute hosts. Existing workloads can run indefinitely, and you do not need to wait for a database migration.

IMPORTANT

Due to certain package dependencies, upgrading the packages for one OpenStack service might cause Python libraries to upgrade before other OpenStack services upgrade. This might cause certain services to fail prematurely. In this situation, continue upgrading the remaining services. All services should be operational upon completion of this scenario.

NOTE

This method may require additional hardware resources to bring up the Red Hat OpenStack Platform 8 Compute nodes.



NOTE

The procedures in this chapter follow the architectural naming convention followed by all Red Hat OpenStack Platform documentation. If you are unfamiliar with this convention, refer to the Architecture Guide available in the Red Hat OpenStack Platform Documentation Suite before proceeding.

6.1. PRE-UPGRADE TASKS

On each node, change to the Red Hat OpenStack Platform 8 repository using the **subscription-manager** command.

```
# subscription-manager repos --disable=rhel-7-server-openstack-7.0-rpms
# subscription-manager repos --enable=rhel-7-server-openstack-8-rpms
```

Upgrade the **openstack-selinux** package:

```
# yum upgrade openstack-selinux
```

This is necessary to ensure that the upgraded services will run correctly on a system with SELinux enabled.

6.2. UPGRADING MARIADB

Perform the follow steps on each host running MariaDB. Complete the steps on one host before starting the process on another host.

1. Stop the service from running on the local node:

```
# pcs resource ban galera-master $(crm_node -n)
```

2. Wait until **pcs status** shows that the service is no longer running on the local node. This may take a few minutes. The local node transitions to slave mode:

```
Master/Slave Set: galera-master [galera]
Masters: [ overcloud-controller-1 overcloud-controller-2 ]
Slaves: [ overcloud-controller-0 ]
```

   The node eventually transitions to stopped:

```
Master/Slave Set: galera-master [galera]
Masters: [ overcloud-controller-1 overcloud-controller-2 ]
Stopped: [ overcloud-controller-0 ]
```

3. Upgrade the relevant packages.

```
# yum upgrade '*mariadb*' '*galera*'
```

4. Allow Pacemaker to schedule the **galera** resource on the local node:

```
# pcs resource clear galera-master
```

5. Wait until **pcs status** shows that the galera resource is running on the local node as a master. The **pcs status** command should provide output similar to the following:

```
Master/Slave Set: galera-master [galera]
Masters: [ overcloud-controller-0 overcloud-controller-1 overcloud-controller-2 ]
```

Perform this procedure on each node individually until the MariaDB cluster completes a full upgrade.

6.3. UPGRADING MONGODB

This procedure upgrades MongoDB, which acts as the backend database for the OpenStack Telemetry service.

1. Remove the **mongod** resource from Pacemaker's control:

```
# pcs resource unmanage mongod-clone
```

2. Stop the service on all Controller nodes. On each Controller node, run the following:

```
# systemctl stop mongod
```

3. Upgrade the relevant packages:

```
#  yum upgrade 'mongodb*' 'python-pymongo*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Restart the **mongod** service on your controllers by running, on each controller:

```
# systemctl start mongod
```

6. Clean up the resource to Pacemaker control:

```
# pcs resource cleanup mongod-clone
```

7. Return the resource to Pacemaker control:

```
# pcs resource manage mongod-clone
```

8. Wait until the output of **pcs status** shows that the above resources are running.

6.4. UPGRADING IDENTITY SERVICE (KEYSTONE)

This procedure upgrades the packages for the Identity service on all Controller nodes simultaneously.

1. Remove Identity service from Pacemaker's control:

```
# pcs resource unmanage openstack-keystone-clone
```

2. Stop the Identity service by running the following on each Controller node:

```
# systemctl stop openstack-keystone
```

3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-keystone*' 'python-keystone*'
```

4. Reload **systemd** to account for updated unit files on each Controller node:

```
# systemctl daemon-reload
```

5. Earlier versions of the installer may not have configured your system to automatically purge expired Keystone token, it is possible that your token table has a large number of expired entries. This can dramatically increase the time it takes to complete the database schema upgrade.
   Flush expired tokens from the database to alleviate the problem. Run the **keystone-manage** command before running the Identity database upgrade.

```
# keystone-manage token_flush
```

   This flushes expired tokens from the database. You can arrange to run this command periodically (e.g., daily) using **cron**.

6. Update the Identity service database schema:

```
# openstack-db --service keystone --update
```

7. Restart the service by running the following on each Controller node:

```
# systemctl start openstack-keystone
```

8. Clean up the Identity service using Pacemaker:

```
# pcs resource cleanup openstack-keystone-clone
```

9. Return the resource to Pacemaker control:

```
# pcs resource manage openstack-keystone-clone
```

10. Wait until the output of **pcs status** shows that the above resources are running.

6.5. UPGRADING IMAGE SERVICE (GLANCE)

This procedure upgrades the packages for the Image service on all Controller nodes simultaneously.

1. Stop the Image service resources in Pacemaker:

```
# pcs resource disable openstack-glance-registry-clone
# pcs resource disable openstack-glance-api-clone
```

2. Wait until the output of **pcs status** shows that both services have stopped running.
3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-glance*' 'python-glance*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Update the Image service database schema:

```
# openstack-db --service glance --update
```

6. Clean up the Image service using Pacemaker:

```
# pcs resource cleanup openstack-glance-api-clone
# pcs resource cleanup openstack-glance-registry-clone
```

7. Restart Image service resources in Pacemaker:

```
# pcs resource enable openstack-glance-api-clone
# pcs resource enable openstack-glance-registry-clone
```

8. Wait until the output of **pcs status** shows that the above resources are running.

6.6. UPGRADING BLOCK STORAGE SERVICE (CINDER)

This procedure upgrades the packages for the Block Storage service on all Controller nodes simultaneously.

1. Stop all Block Storage service resources in Pacemaker:

```
# pcs resource disable openstack-cinder-api-clone
# pcs resource disable openstack-cinder-scheduler-clone
# pcs resource disable openstack-cinder-volume
```

2. Wait until the output of **pcs status** shows that the above services have stopped running.

3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-cinder*' 'python-cinder*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Update the Block Storage service database schema:

```
# openstack-db --service cinder --update
```

6. Clean up the Block Storage service using Pacemaker:

```
# pcs resource cleanup openstack-cinder-volume
# pcs resource cleanup openstack-cinder-scheduler-clone
# pcs resource cleanup openstack-cinder-api-clone
```

7. Restart all Block Storage service resources in Pacemaker:

```
# pcs resource enable openstack-cinder-volume
# pcs resource enable openstack-cinder-scheduler-clone
# pcs resource enable openstack-cinder-api-clone
```

8. Wait until the output of **pcs status** shows that the above resources are running.

6.7. UPGRADING ORCHESTRATION (HEAT)

This procedure upgrades the packages for the Orchestration service on all Controller nodes simultaneously.

1. Stop Orchestration resources in Pacemaker:

```
# pcs resource disable openstack-heat-api-clone
# pcs resource disable openstack-heat-api-cfn-clone
# pcs resource disable openstack-heat-api-cloudwatch-clone
# pcs resource disable openstack-heat-engine-clone
```

2. Wait until the output of **pcs status** shows that the above services have stopped running.

3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-heat*' 'python-heat*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Update the Orchestration database schema:

```
# openstack-db --service heat --update
```

6. Clean up the Orchestration service using Pacemaker:

```
# pcs resource cleanup openstack-heat-clone
# pcs resource cleanup openstack-heat-api-cloudwatch-clone
# pcs resource cleanup openstack-heat-api-cfn-clone
# pcs resource cleanup openstack-heat-api-clone
```

7. Restart Orchestration resources in Pacemaker:

```
# pcs resource enable openstack-heat-clone
# pcs resource enable openstack-heat-api-cloudwatch-clone
# pcs resource enable openstack-heat-api-cfn-clone
# pcs resource enable openstack-heat-api-clone
```

8. Wait until the output of **pcs status** shows that the above resources are running.

6.8. UPGRADING TELEMETRY (CEILOMETER)

This procedure upgrades the packages for the Telemetry service on all Controller nodes simultaneously.

1. Stop all Telemetry resources in Pacemaker:

```
# pcs resource disable openstack-ceilometer-central
# pcs resource disable openstack-ceilometer-api-clone
# pcs resource disable openstack-ceilometer-alarm-evaluator-clone
# pcs resource disable openstack-ceilometer-collector-clone
# pcs resource disable openstack-ceilometer-notification-clone
# pcs resource disable openstack-ceilometer-alarm-notifier-clone
# pcs resource disable delay-clone
```

2. Wait until the output of **pcs status** shows that the above services have stopped running.

3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-ceilometer*' 'python-ceilometer*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Use the following command to update Telemetry database schema.

```
# ceilometer-dbsync
```

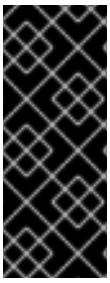
6. Clean up the Telemetry service using Pacemaker:

```
# pcs resource cleanup delay-clone
# pcs resource cleanup openstack-ceilometer-alarm-notifier-clone
# pcs resource cleanup openstack-ceilometer-notification-clone
# pcs resource cleanup openstack-ceilometer-collector-clone
# pcs resource cleanup openstack-ceilometer-alarm-evaluator-clone
# pcs resource cleanup openstack-ceilometer-api-clone
# pcs resource cleanup openstack-ceilometer-central
```

7. Restart all Telemetry resources in Pacemaker:

```
# pcs resource enable delay-clone
# pcs resource enable openstack-ceilometer-alarm-notifier-clone
# pcs resource enable openstack-ceilometer-notification-clone
# pcs resource enable openstack-ceilometer-collector-clone
# pcs resource enable openstack-ceilometer-alarm-evaluator-clone
# pcs resource enable openstack-ceilometer-api-clone
# pcs resource enable openstack-ceilometer-central
```

8. Wait until the output of **pcs status** shows that the above resources are running.



IMPORTANT

Previous versions of the Telemetry service used an value for the **rpc_backend** parameter that is now deprecated. Check the **rpc_backend** parameter in the **/etc/ceilometer/ceilometer.conf** file is set to the following:

```
rpc_backend=rabbit
```

6.9. UPGRADING THE COMPUTE SERVICE (NOVA) ON CONTROLLER NODES

This procedure upgrades the packages for the Compute service on all Controller nodes simultaneously.

1. Stop all Compute resources in Pacemaker:

```
# pcs resource disable openstack-nova-novncproxy-clone
# pcs resource disable openstack-nova-consoleauth-clone
# pcs resource disable openstack-nova-conductor-clone
# pcs resource disable openstack-nova-api-clone
# pcs resource disable openstack-nova-scheduler-clone
```

2. Wait until the output of **pcs status** shows that the above services have stopped running.

3. Upgrade the relevant packages:

```
# yum upgrade 'openstack-nova*' 'python-nova*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Update the Compute database schema:

```
# openstack-db --service nova --update
```

6. If you are performing a rolling upgrade of your compute hosts you need to set explicit API version limits to ensure compatibility between your Kilo and Liberty environments.
Before starting Compute services on Controller or Compute nodes, set the **compute** option in the **[upgrade_levels]** section of **nova.conf** to the previous Red Hat OpenStack Platform version (**kilo**):

```
# crudini --set /etc/nova/nova.conf upgrade_levels compute kilo
```

This ensures the Controller node can still communicate to the Compute nodes, which are still using the previous version.

You will need to first unmanage the Compute resources by running **pcs resource unmanage** on one Controller node:

```
# pcs resource unmanage openstack-nova-novncproxy-clone
# pcs resource unmanage openstack-nova-consoleauth-clone
# pcs resource unmanage openstack-nova-conductor-clone
# pcs resource unmanage openstack-nova-api-clone
# pcs resource unmanage openstack-nova-scheduler-clone
```

Restart all the services on all controllers:

```
# openstack-service restart nova
```

You should return control to the Pacemaker after upgrading all of your compute hosts to OpenStack Liberty.

```
# pcs resource manage openstack-nova-scheduler-clone
# pcs resource manage openstack-nova-api-clone
# pcs resource manage openstack-nova-conductor-clone
# pcs resource manage openstack-nova-consoleauth-clone
# pcs resource manage openstack-nova-novncproxy-clone
```

7. Clean up all Compute resources in Pacemaker:

```
# pcs resource cleanup openstack-nova-scheduler-clone
# pcs resource cleanup openstack-nova-api-clone
# pcs resource cleanup openstack-nova-conductor-clone
# pcs resource cleanup openstack-nova-consoleauth-clone
# pcs resource cleanup openstack-nova-novncproxy-clone
```

8. Restart all Compute resources in Pacemaker:

```
# pcs resource enable openstack-nova-scheduler-clone
# pcs resource enable openstack-nova-api-clone
```

```
# pcs resource enable openstack-nova-conductor-clone
# pcs resource enable openstack-nova-consoleauth-clone
# pcs resource enable openstack-nova-novncproxy-clone
```

9. Wait until the output of **pcs status** shows that the above resources are running.

6.10. UPGRADING OPENSTACK NETWORKING (NEUTRON)

This procedure upgrades the packages for the Networking service on all Controller nodes simultaneously.

1. Prevent Pacemaker from triggering the OpenStack Networking cleanup scripts:

```
# pcs resource unmanage neutron-ovs-cleanup-clone
# pcs resource unmanage neutron-netns-cleanup-clone
```

2. Stop OpenStack Networking resources in Pacemaker:

```
# pcs resource disable neutron-server-clone
# pcs resource disable neutron-openvswitch-agent-clone
# pcs resource disable neutron-dhcp-agent-clone
# pcs resource disable neutron-l3-agent-clone
# pcs resource disable neutron-metadata-agent-clone
```

3. Upgrade the relevant packages

```
# yum upgrade 'openstack-neutron*' 'python-neutron*'
```

4. Install packages for the advanced Openstack Networking services enabled in the **neutron.conf** file. For example, to upgrade the **openstack-neutron-vpnaas**, **openstack-neutron-fwaas** and **openstack-neutron-lbaas** services:

```
# yum install openstack-neutron-vpnaas
# yum install openstack-neutron-fwaas
# yum install openstack-neutron-lbaas
```

   Installing these packages will create the corresponding configuration files.

5. For the VPNaaS, LBaaS service entries in the **neutron.conf** file, copy the service_provider entries to the corresponding **neutron-*aas.conf** file located in **/etc/neutron** and comment these entries from the **neutron.conf** file.
   For the FWaaS service entry, the **service_provider** parameters **should remain** in the **neutron.conf** file.

6. On every node that runs the LBaaS agents, install the **openstack-neutron-lbaas** package.

```
# yum install openstack-neutron-lbaas
```

7. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

8. Update the OpenStack Networking database schema:

```
# openstack-db --service neutron --update
```

9. Clean up OpenStack Networking resources in Pacemaker:

```
# pcs resource cleanup neutron-metadata-agent-clone
# pcs resource cleanup neutron-l3-agent-clone
# pcs resource cleanup neutron-dhcp-agent-clone
# pcs resource cleanup neutron-openvswitch-agent-clone
# pcs resource cleanup neutron-server-clone
```

10. Restart OpenStack Networking resources in Pacemaker:

```
# pcs resource enable neutron-metadata-agent-clone
# pcs resource enable neutron-l3-agent-clone
# pcs resource enable neutron-dhcp-agent-clone
# pcs resource enable neutron-openvswitch-agent-clone
# pcs resource enable neutron-server-clone
```

11. Return the cleanup agents to Pacemaker control:

```
# pcs resource manage neutron-ovs-cleanup-clone
# pcs resource manage neutron-netns-cleanup-clone
```

12. Wait until the output of **pcs status** shows that the above resources are running.

6.11. UPGRADING DASHBOARD (HORIZON)

This procedure upgrades the packages for the Dashboard on all Controller nodes simultaneously.

1. Stop the Dashboard resource in Pacemaker:

```
# pcs resource disable httpd-clone
```

2. Wait until the output of **pcs status** shows that the service has stopped running.
3. Upgrade the relevant packages:

```
# yum upgrade httpd 'openstack-dashboard*' 'python-django*'
```

4. Reload **systemd** to account for updated unit files:

```
# systemctl daemon-reload
```

5. Restart the web server on all your controllers to apply all changes:

```
# service httpd restart
```

6. Clean up the Dashboard resource in Pacemaker:

```
#  pcs resource cleanup httpd-clone
```

7. Restart the Dashboard resource in Pacemaker:

```
# pcs resource enable httpd-clone
```

8. Wait until the output of **pcs status** shows that the above resource is running.

6.12. UPGRADING COMPUTE (NOVA) NODES

This procedure upgrades the packages for on a single Compute node. Run this procedure on each Compute node individually.

If you are performing a rolling upgrade of your compute hosts you need to set explicit API version limits to ensure compatibility between your Kilo and Liberty environments.

Before starting Compute services on Controller or Compute nodes, set the **compute** option in the **[upgrade_levels]** section of **nova.conf** to the previous Red Hat OpenStack Platform version (**kilo**):

```
# crudini --set /etc/nova/nova.conf upgrade_levels compute kilo
```

This ensures the Controller node can still communicate to the Compute nodes, which are still using the previous version.

1. Stop all OpenStack services on the host:

```
# openstack-service stop
```

2. Upgrade all packages:

```
# yum upgrade
```

3. Start all openstack services on the host:

```
# openstack-service start
```

4. After you have upgraded all of your hosts, remove the API limits configured in the previous step. On all of your hosts:

```
# crudini --del /etc/nova/nova.conf upgrade_levels compute
```

5. Restart all openstack services on the host:

```
# openstack-service restart
```

6.13. POST-UPGRADE TASKS

After completing all of your individual service upgrades, you should perform a complete package upgrade on all nodes:

```
# yum upgrade
```

This will ensure that all packages are up-to-date. You may want to schedule a restart of your OpenStack hosts at a future date in order to ensure that all running processes are using updated versions of the underlying binaries.

Review the resulting configuration files. The upgraded packages will have installed **.rpmnew** files appropriate to the Red Hat OpenStack Platform 8 version of the service.

New versions of OpenStack services may deprecate certain configuration options. You should also review your OpenStack logs for any deprecation warnings, because these may cause problems during a future upgrade. For more information on the new, updated and deprecated configuration options for each service , see Configuration Reference available from: Red Hat OpenStack Platform Documentation Suite.

CHAPTER 7. TROUBLESHOOTING DIRECTOR-BASED UPGRADES

This section provides advice for troubleshooting issues with both.

7.1. UNDERCLOUD UPGRADES

In situations where an Undercloud upgrade command (**openstack undercloud upgrade**) fails, use the following advice to locate the issue blocking upgrade progress:

- The **openstack undercloud upgrade** command prints out a progress log while it runs. If an error occurs at any point in the upgrade process, the command halts at the point of error. Use this information to identify any issues impeding upgrade progress.
- The **openstack undercloud upgrade** command runs Puppet to configure Undercloud services. This generates useful Puppet reports in the following directories:
  - **/var/lib/puppet/state/last_run_report.yaml** - The last Puppet reports generated for the Undercloud. This file shows any causes of failed Puppet actions.
  - **/var/lib/puppet/state/last_run_summary.yaml** - A summary of the **last_run_report.yaml** file.
  - **/var/lib/puppet/reports** - All Puppet reports for the Undercloud.
    Use this information to identify any issues impeding upgrade progress.
- Check for any failed services:

```
$ sudo systemctl -t service
```

  If any services have failed, check their corresponding logs. For example, if **openstack-ironic-api** failed, use the following commands to check the logs for that service:

```
$ sudo journalctl -xe -u openstack-ironic-api
$ sudo tail -n 50 /var/log/ironic/ironic-api.log
```

After correcting the issue impeding the Undercloud upgrade, rerun the upgrade command:

```
$ openstack undercloud upgrade
```

The upgrade command begins again and configures the Undercloud.

7.2. OVERCLOUD UPGRADES

In situations where an Overcloud upgrade process fails, use the following advice to locate the issue blocking upgrade progress:

- Check the Heat stack listing and identify any stacks that have an **UPDATE_FAILED** status. The following command identifies these stacks:

```
$ heat stack-list --show-nested | awk -F "|" '{ print $3,$4 }' | grep "UPDATE_FAILED" | column -t
```

View the failed stack and its template to identify how the stack failed:

```
$ heat stack-show overcloud-Controller-qyoy54dyhrll-1-gtwy5bgta3np
$ heat template-show overcloud-Controller-qyoy54dyhrll-1-gtwy5bgta3np
```

- Check that Pacemaker is running correctly on all Controller nodes. If necessary, log into a Controller node and restart the Controller cluster:

```
$ sudo pcs cluster start
```

After correcting the issue impeding the Overcloud upgrade, rerun the **openstack overcloud deploy** command for the failed upgrade step you attempted. This following is an example of the first **openstack overcloud deploy** command in the upgrade process, which includes the **major-upgrade-pacemaker-init.yaml**:

```
$ openstack overcloud deploy --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/net-single-nic-with-vlans.yaml  \
  -e network_env.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/major-upgrade-pacemaker-init.yaml
```

The **openstack overcloud deploy** retries the Overcloud stack update.