Red Hat Enterprise Linux OpenStack Platform 7
Upgrading OpenStack

Upgrading Red Hat Enterprise Linux OpenStack Platform

OpenStack Documentation Team

Upgrading Red Hat Enterprise Linux OpenStack Platform

OpenStack Documentation Team
Red Hat Customer Content Services
rhos-docs@redhat.com

Legal Notice

Abstract

This document lays out the different methods through which users can upgrade from Red Hat Enterprise Linux OpenStack Platform 6 (Juno) to 7 (Kilo). 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 an overview of the process for upgrading to Red Hat Enterprise Linux OpenStack Platform 7 from Red Hat Enterprise Linux OpenStack Platform 6.

If you are currently on the RHEL OpenStack Platform 5 (Icehouse) release and are looking to upgrade to RHEL OpenStack Platform 7 (Kilo) release, you need to ensure that you upgrade to the RHEL OpenStack Platform 6 (Juno) release first. The procedures to perform this upgrade are available under *Upgrading* from: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/6/

Red Hat only supports upgrades to Red Hat Enterprise Linux OpenStack Platform 7 on Red Hat Enterprise Linux 7. In addition, Red Hat recommends the following different methods to support this upgrade path, each of which are described better in Section 1.1, “Upgrade Methods Comparison”. Each of these methods have corresponding documentation available in the *Upgrade Methods Comparison*. For more details on the components, refer *Components Overview* available from: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/

For both the *Upgrade All at Once* and the *Upgrading Individually* scenario, this guide describes both the High Availability (HA) as well as the non-HA procedures. The major difference between the HA and non-HA procedures is that for the HA upgrade, you need to ensure your Pacemaker services are disabled and enabled appropriately.

All of the upgrade paths described below, allow you to upgrade to a working RHEL OpenStack Platform 7 (Kilo) release. It is recommended that you perform further upgrade procedures to ensure that deployment is RHEL OpenStack Platform 7 director ready using the parallel cloud migration . Parallel cloud migration for RHEL OpenStack Platform 7 deals with upgrading your existing OpenStack clouds deployed with another installer to the RHEL OpenStack Platform 7 director.

1.1. Upgrade Methods Comparison

Red Hat recommends the following methods for upgrading to Red Hat Enterprise Linux OpenStack Platform 7. The following table provides a brief description of each.

Table 1.1. Upgrade Methods

Method	Description	Pros	Cons
All at Once	In this method, you take down all of the OpenStack services at the same time, do the upgrade, then bring all services back up after the upgrade process is complete. For more information, see Chapter 3, *Upgrade OpenStack All at Once*.	This upgrade process is simple. Because everything is down, no orchestration is required. Although services will be down, VM workloads can be kept running if there are no requirements to move from one version of Red Hat Enterprise Linux to another (that is, from v7.0 to v7.1).	All of your services will be unavailable at the same time. In a large environment, the upgrade can result in a potentially extensive downtime while you wait for database-schema upgrades to complete. Downtime can be mitigated by proper dry-runs of your upgrade procedure on a test environment as well as scheduling a specific downtime window for the upgrade.

Method	Description	Pros	Cons
Service by Service with Live Compute Upgrade	This method is a variation of the service-by-service upgrade, with a change in how the Compute service is upgraded. With this method, you can take advantage of Red Hat Enterprise Linux OpenStack Platform 7 features that allow to run older compute nodes in parallel with upgraded compute nodes. For more information, see Chapter 4, *Upgrade OpenStack by Updating Each Service Individually, with Live Compute*.	This method 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.	Additional hardware resources may be required to bring up the Red Hat Enterprise Linux OpenStack Platform 7 (Kilo) Compute nodes.

For all methods:

- Ensure you have subscribed to the correct channels 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.
- To upgrade OpenStack Networking, you will need to set the correct **`libvirt_vif_driver`** in **`/etc/nova/nova.conf`** as the old hybrid driver is now deprecated. To do so, run the following on your Compute API host:

```
# openstack-config --set /etc/nova/nova.conf \
DEFAULT libvirt_vif_driver nova.virt.libvirt.vif.LibvirtGenericVIFDriver
```

Warning

Red Hat does not support:

- Upgrading any Beta release of Red Hat Enterprise Linux OpenStack Platform to any supported release (for example, 6 or 7).
- Upgrading Compute Networking (nova-networking) to OpenStack Networking (neutron) in Red Hat Enterprise Linux OpenStack Platform 7. The only supported networking upgrade is between versions of OpenStack Networking (neutron) from Red Hat Enterprise Linux OpenStack Platform version 6 to version 7 .

Chapter 2. Prerequisites

This section discusses channel and repository settings required for deploying Red Hat Enterprise Linux OpenStack Platform 7.

Warning

Although older Red Hat OpenStack repositories are available, you must ensure that your system can no longer access them before installing Red Hat Enterprise Linux OpenStack Platform 7. For example, unsubscribe from or disable the following:

- Red Hat Enterprise Linux OpenStack Platform 4 (Havana) -- rhel-6-server-openstack-4.0-rpms
- Red Hat Enterprise Linux OpenStack Platform 5 (Icehouse) -- rhel-7-server-openstack-5.0-rpms
- Red Hat Enterprise Linux OpenStack Platform 6 (Juno) -- rhel-7-server-openstack-6.0-rpms

Packstack registers systems to Red Hat Network using Subscription Manager. You might encounter problems if your systems have already been registered and subscribed to the Red Hat OpenStack channels using RHN Classic.

Note

The Red Hat Common for RHEL Server channel is recommended for use if creating custom Red Hat Enterprise Linux guest images that require **`cloud-init`**.

For Red Hat Enterprise Linux 7, run:

```
# subscription-manager repos --enable=rhel-7-server-rh-common-rpms
```

2.1. Configure Content Delivery Network (CDN) Channels

You can install Red Hat Enterprise Linux OpenStack Platform 7 through the Content Delivery Network (CDN). To do so, configure **`subscription-manager`** to use the correct channels.

Run the following command to enable a CDN channel:

```
# subscription-manager repos --enable=[reponame]
```

Run the following command to disable a CDN channel:

```
# subscription-manager repos --disable=[reponame]
```

Red Hat Enterprise Linux 7

The following tables outline the channels for Red Hat Enterprise Linux 7.

Table 2.1. Required Channels

Channel	Repository Name
Red Hat Enterprise Linux 7 Server (RPMS)	**rhel-7-server-rpms**
Red Hat OpenStack 7.0 for Server 7 (RPMS)	**rhel-7-server-openstack-7.0-rpms**
Red Hat Enterprise Linux 7 Server - RH Common (RPMs)	**rhel-7-server-rh-common-rpms**

Table 2.2. Optional Channels

Channel	Repository Name
Red Hat Enterprise Linux 7 Server - Optional	**rhel-7-server-optional-rpms**

Red Hat Enterprise Linux OpenStack Platform Director

The following tables outline the channels for the Red Hat Enterprise Linux OpenStack Platform Director.

Table 2.3. Required Channels

Channel	Repository Name
Red Hat Enterprise Linux OpenStack Platform Director 7.0 (RPMs)	**rhel-7-server-openstack-7.0-director-rpms**
Red Hat Enterprise Linux 7 Server (RPMS)	**rhel-7-server-rpms**
Red Hat Software Collections RPMs for Red Hat Enterprise Linux 7 Server	**rhel-server-rhscl-7-rpms**
Images on CDN (Optional)	**rhel-7-server-openstack-7.0-files**
Red Hat Enterprise Linux OpenStack Platform 7.0 Operational Tools	**rhel-7-server-openstack-7.0-optools-rpms**

Disable Channels

The following table outlines the channels you must disable to ensure Red Hat Enterprise Linux OpenStack Platform 7 functions correctly.

Table 2.4. Disable Channels

Channel	Repository Name
Red Hat CloudForms Management Engine	**"cf-me-*"**
Red Hat CloudForms Tools for RHEL 6	**"rhel-6-server-cf-*"**
Red Hat Enterprise Virtualization	**"rhel-6-server-rhev*"**
Red Hat Enterprise Linux 6 Server - Extended Update Support	**"*-eus-rpms"**

Chapter 3. Upgrade OpenStack All at Once

This chapter discusses the procedure to upgrade your deployment from RHEL OpenStack Platform 6 to RHEL OpenStack Platform 7 by upgrading all the services at once which involves disabling all the OpenStack services at the same time, performing the upgrade, then enabling all the services back up after the upgrade process is complete. Of all methods for upgrading to Red Hat Enterprise Linux OpenStack Platform 7, this one is the simplest.

If you do not need to upgrade the base operating system to the latest version of Red Hat Enterprise Linux 7 before upgrading the packages, no orchestration is required as all the OpenStack services are down.

In a large environment, the upgrade can result in a potentially extensive downtime while you wait for database-schema upgrades to complete. Downtime can be mitigated by proper dry-runs of your upgrade procedure on a test environment as well as scheduling a specific downtime window for the upgrade.

Note

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

Before you begin upgrading your OpenStack deployment, subscribe to the proper channels. For more information, refer Chapter 2, *Prerequisites*

3.1. Upgrade All OpenStack Services Simultaneously

The following procedure describes the steps you need to follow to upgrade your cloud deployment from Juno to Kilo by updating all the components at once. Some of the steps have options for both the highly available (HA) and non-highly available (non-HA) scenarios. If your cloud is highly available, then you need to use the Pacemaker commands for starting and stopping your cloud environment.

Run the following steps on all of your hosts:

Procedure 3.1. Upgrading OpenStack components on a host

1. Install the yum repository for Red Hat Enterprise Linux OpenStack Platform 7 (Kilo).
2. Check that the **`openstack-utils`** package is installed:

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

3. Take down all OpenStack services on all the nodes. This step depends on how your services are distributed among your nodes.

   A. **In a non-HA environment:**

   To stop all the OpenStack services running on a node, login to the node and run:

   ```
   # openstack-service stop
   ```

   B. **In an HA environment:**

a. To stop all the OpenStack services running on a node, login to the node and run:

```
# openstack-service stop
```

b. Disable all 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
```

Then wait until the output of **pcs status** shows that all resources have stopped.

4. Perform a complete upgrade of all packages, and then flush expired tokens in the Identity service (might decrease the time required to synchronize the database):

```
# yum upgrade
```

5. Perform the necessary configuration updates on each of your services.

a. **Identity service**

In the RHEL OpenStack Platform 7 (Kilo) release, the location of the token persistence backends has changed. You need to update the **driver** option in the **[token]** section of the **keystone.conf**. To do this, replace any instance of **keystone.token.backends** with **keystone.token.persistence.backends**.

```
# sed -i
's/keystone.token.backends/keystone.token.persistence.backends/g
' \
/etc/keystone/keystone.conf
```

Package updates may include new **systemd** unit files, so confirm that **systemd** is aware of any updated files.

```
# systemctl daemon-reload
```

b. **OpenStack Networking service**

Once you have completed upgrading the OpenStack Networking service, you need to edit the **rootwrap dhcp.filter** configuration file.

To do so, in the **/usr/share/neutron/rootwrap/dhcp.filters** file, replace the value of **dnsmasq**. For example, replace:

```
dnsmasq: EnvFilter, env, root, CONFIG_FILE=, NETWORK_ID=,
dnsmasq
```

with:

```
dnsmasq: CommandFilter, dnsmasq, root
```

6. Upgrade the database schema for each service that uses the database. To do so, login to the node hosting the service and run:

```
# 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 3.1. Project name of each OpenStack service that uses the database

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

Certain services require additional database maintenance as part of the Juno to Kilo upgrade that is not covered by the **openstack-db** command:

a. **Identity service**

Earlier versions of the installer may not have configured your system to automatically purge expired Keystone tokens. 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.

You can alleviate this problem by running the following command before beginning the Keystone database upgrade process:

```
# keystone-manage token_flush
```

This will flush expired tokens from the database. You should arrange to run this command periodically (for example, daily) using **cron**.

b. **Compute**

After fully upgrading to Kilo (that is, all nodes running Kilo), you should start a background migration of flavor information. Kilo conductor nodes will do this on the fly when necessary, but the rest of the idle data needs to be migrated in the the background. Run the following command as a **nova** user:

```
# runuser -u nova -- nova-manage db migrate_flavor_data
```

7. Review the resulting configuration files. The upgraded packages will have installed **.rpmnew** files appropriate to the Red Hat Enterprise Linux OpenStack Platform 7 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 Enterprise Linux OpenStack Platform Documentation Suite.

8. If the package upgrades you performed require a reboot (for example, if a new kernel was installed as part of the upgrade), reboot the affected hosts now while the OpenStack service is still disabled.

A. **In a non-HA environment:**

To restart the OpenStack service, run the following command on each node:

```
# openstack-service start
```

B. **In a HA environment:**

a. Allow Pacemaker to restart your resources by resetting the **stop-all-resources** property. On a single member of your Pacemaker cluster, run:

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

Then wait until the output of **pcs status** shows that all resources have started (this may take several minutes).

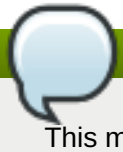
b. Restart OpenStack services on the compute nodes. On each compute node, run:

```
# openstack-service start
```

Chapter 4. Upgrade OpenStack by Updating Each Service Individually, with Live Compute

With this procedure, all services are upgraded individually. Each service upgrade involves a package update and a database schema update.

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.



Note

This method may require additional hardware resources to bring up the Red Hat Enterprise Linux OpenStack Platform 7 Compute nodes.

Note

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

Before you begin upgrading your OpenStack deployment, subscribe to the proper channels. For more information, refer Chapter 2, *Prerequisites*

4.1. Upgrading OpenStack by Updating Service Individually, With Live Compute in a Non-HA 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.

1. **Pre-upgrade tasks:**

   On all of your hosts:

   a. Install the yum repository for Red Hat Enterprise Linux OpenStack Platform 7 (Kilo).

   b. Upgrade the **`openstack-selinux`** package, if available:

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

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

2. **Upgrade each of your services:**

   The following steps provide specific instructions for each service, and the order in which they should be upgraded.

   a. **Identity (keystone)**

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.

To flush expired tokens from the database and alleviate the problem, the **keystone-manage** command can be used before running the Identity database upgrade.

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

On your Identity host, run:

```
# openstack-service stop keystone
# yum -d1 -y upgrade \*keystone\*
# keystone-manage token_flush
# openstack-db --service keystone --update
# openstack-service start keystone
```

b. **Object Storage (swift)**

On your Object Storage hosts, run:

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

c. **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
```

d. **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
```

e. **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
```

f. **Telemetry (ceilometer)**

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

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

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

```
# ceilometer-dbsync
```

This command allows you to configure MySQL as a back-end for Telemetry service.

For a list of Telemetry component services, refer to Launch the Telemetry API and Agents.

c. 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
```

g. **Compute (nova)**

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

Before starting Kilo controller or compute services, you need to set the **`compute`** option in the **`[upgrade_levels]`** section of **`nova.conf`** to **`juno`**:

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

You need to make this change on your controllers and on your compute hosts.

You should undo this operation after upgrading all of your compute hosts to OpenStack Kilo.

b. On your Compute host, run:

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

c. After fully upgrading to Kilo (i.e. all nodes are running Kilo), you should start a background migration of flavor information. Kilo conductor nodes will do this on the fly when necessary, but the rest of the idle data needs to be migrated in the the background. Run the following command as a **`nova`** user:

```
# runuser -u nova -- nova-manage db migrate_flavor_data
```

d. After you have upgraded all of your hosts to Kilo, 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
```

e. Restart the Compute service on all the compute hosts and controllers:

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

h. **OpenStack Networking (neutron)**

a. On your OpenStack Networking host, run:

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

b. Once you have completed upgrading the OpenStack Networking service, you need to edit the **`rootwrap dhcp.filter`** configuration file.

To do so, in the **`/usr/share/neutron/rootwrap/dhcp.filters`** file, replace the value of **`dnsmasq`**. For example, replace:

```
dnsmasq: EnvFilter, env, root, CONFIG_FILE=, NETWORK_ID=,
dnsmasq
```

with:

```
dnsmasq: CommandFilter, dnsmasq, root
```

c. Restart the OpenStack Networking service:

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

i. **Dashboard (horizon)**

On your Dashboard host, run:

```
# yum -y upgrade \*horizon\* \*openstack-dashboard\*
# yum -d1 -y upgrade \*horizon\* \*python-django\*
# systemctl restart httpd
```

3. **Post-upgrade tasks:**

a. 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.

b. Review the resulting configuration files. The upgraded packages will have installed **`.rpmnew`** files appropriate to the Red Hat Enterprise Linux OpenStack Platform 7 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 Enterprise Linux OpenStack Platform Documentation Suite.

4.2. Upgrading OpenStack by Updating Service Individually, With Live Compute in an HA 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 High Availability (HA) environment.

1. **Pre-upgrade tasks:**

   On all of your hosts:

   a. If you are running **`Puppet`** as configured by Staypuft, you must disable it:

   ```
   # systemctl stop puppet
   # systemctl disable puppet
   ```

   This ensures that the Staypuft-configured puppet will not revert changes made as part of the upgrade process.

   b. Install the yum repository for Red Hat Enterprise Linux OpenStack Platform 7 (Kilo).

   c. Manually upgrade all the python packages.

   ```
   # yum upgrade python*
   ```

   d. Upgrade the **`openstack-selinux`** package, if available:

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

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

2. **Service upgrades:**

   Upgrade each of your services. The following is a reasonable order in which to perform the upgrades on your controllers:

   Upgrade MariaDB:

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

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

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

   b. 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 will first transition to slave mode:

   ```
   Master/Slave Set: galera-master [galera]
   Masters: [ pcmk-mac525400aeb753 pcmk-mac525400bab8ae ]
   Slaves: [ pcmk-mac5254004bd62f ]
   ```

It will eventually transition to stopped:

```
Master/Slave Set: galera-master [galera]
Masters: [ pcmk-mac525400aeb753 pcmk-mac525400bab8ae ]
Stopped: [ pcmk-mac5254004bd62f ]
```

c. Upgrade the relevant packages.

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

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

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

e. Wait until **pcs status** shows that the galera resource is running on the local node as a master. The output from **pcs status** should include something like:

```
Master/Slave Set: galera-master [galera]
Masters: [ pcmk-mac5254004bd62f pcmk-mac525400aeb753
pcmk-mac525400bab8ae ]
```

Upgrade MongoDB:

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

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

b. Stop the service on all of your controllers. On each controller, run:

```
# systemctl stop mongod
```

c. Upgrade the relevant packages:

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

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

```
# systemctl daemon-reload
```

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

```
# systemctl start mongod
```

f. Clean up the resource to Pacemaker control:

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

g. Return the resource to Pacemaker control:

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

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

Upgrade Identity service (keystone):

a. Remove Identity service from Pacemaker's control:

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

b. Stop the Identity service by running the following on each of your controllers:

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

c. Upgrade the relevant packages:

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

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

```
# systemctl daemon-reload
```

e. In the RHEL OpenStack Platform 7 (Kilo) release, the location of the token persistence backends has changed. You need to update the **driver** option in the **[token]** section of the **keystone.conf**. To do this, replace any instance of **keystone.token.backends** with **keystone.token.persistence.backends**.

```
# sed -i
's/keystone.token.backends/keystone.token.persistence.backends/g
' \
/etc/keystone/keystone.conf
```

Package updates may include new **systemd** unit files, so ensure that **systemd** is aware of any updated files.

```
# systemctl daemon-reload
```

f. 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.

To flush expired tokens from the database and alleviate the problem, the **keystone-manage** command can be used before running the Identity database upgrade.

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

```
# keystone-manage token_flush
```

g. Update the Identity service database schema:

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

h. Restart the service by running the following on each of your controllers:

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

i. Clean up the Identity service using Pacemaker:

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

j. Return the resource to Pacemaker control:

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

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

Upgrade Image service (glance):

a. Stop the Image service resources in Pacemaker:

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

b. Wait until the output of **pcs status** shows that both services have stopped running.

c. Upgrade the relevant packages:

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

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

```
# systemctl daemon-reload
```

e. Update the Image service database schema:

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

f. Clean up the Image service using Pacemaker:

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

g. Restart Image service resources in Pacemaker:

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

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

Upgrade Block Storage service (cinder):

a. Stop all Block Storage service resources in Pacemaker:

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

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

c. Upgrade the relevant packages:

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

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

```
# systemctl daemon-reload
```

e. Update the Block Storage service database schema:

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

f. Clean up the Block Storage service using Pacemaker:

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

g. Restart all Block Storage service resources in Pacemaker:

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

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

Upgrade Orchestration (heat):

a. Stop Orchestration resources in Pacemaker:

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

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

c. Upgrade the relevant packages:

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

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

```
# systemctl daemon-reload
```

e. Update the Orchestration database schema:

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

f. Clean up the Orchestration service using Pacemaker:

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

g. Restart Orchestration resources in Pacemaker:

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

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

Upgrade Telemetry (ceilometer):

a. 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 ceilometer-delay-clone
```

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

c. Upgrade the relevant packages:

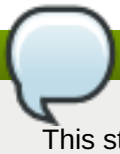
```
# yum upgrade 'openstack-ceilometer*' 'python-ceilometer*'
```

d. Reload **`systemd`** to account for updated unit files:

```
# systemctl daemon-reload
```

e. If you are using the MySQL backend for Telemetry, update the Telemetry database schema.

```
# openstack-db --service ceilometer --update
```



Note

This step is not necessary of you are using the MongoDB backend.

f. Clean up the Telemetry service using Pacemaker:

```
# pcs resource cleanup ceilometer-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
```

g. Restart all Telemetry resources in Pacemaker:

```
# pcs resource enable ceilometer-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
```

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

Upgrade Compute (nova):

a. 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
```

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

c. Upgrade the relevant packages:

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

d. Reload **`systemd`** to account for updated unit files:

```
# systemctl daemon-reload
```

e. Update the Compute database schema:

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

After fully upgrading to Kilo (i.e. all nodes are running Kilo), you should start a background migration of flavor information. Kilo conductor nodes will do this on the fly when necessary, but the rest of the idle data needs to be migrated in the the background. Run the following command as a **`nova`** user:

```
# runuser -u nova -- nova-manage db migrate_flavor_data
```

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

Before starting Kilo controller or compute services, you need to set the **compute** option in the **[upgrade_levels]** section of **nova.conf** to **juno**:

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

You will need to first unmanage the Compute resources by running **pcs resource unmanage** on one of your controllers:

```
# 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 Kilo.

```
# 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
```

g. 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
```

h. 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
```

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

Upgrade OpenStack Networking (neutron):

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

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

b. 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
```

c. Upgrade the relevant packages

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

d. Install packages for the advanced Openstack Networking services enabled in the **neutron.conf** file, for example, **openstack-neutron-vpnaas**, **openstack-neutron-fwaas** and **openstack-neutron-lbaas**.

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

Installing these packages will create the corresponding configuration files.

e. 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.

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

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

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

```
# systemctl daemon-reload
```

h. Update the OpenStack Networking database schema:

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

i. Once you have completed upgrading the OpenStack Networking service, you need to edit the **rootwrap dhcp.filter** configuration file.

To do so, in the **/usr/share/neutron/rootwrap/dhcp.filters** file, replace the value of **dnsmasq**. For example, replace:

```
dnsmasq: EnvFilter, env, root, CONFIG_FILE=, NETWORK_ID=,
dnsmasq
```

with

```
dnsmasq: CommandFilter, dnsmasq, root
```

j. 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
```

k. 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
```

l. Return the cleanup agents to Pacemaker control:

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

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

Upgrade Dashboard (horizon):

a. Stop the Dashboard resource in Pacemaker:

```
# pcs resource disable horizon-clone
```

b. Wait until the output of **`pcs status`** shows that the service has stopped running.

c. Upgrade the relevant packages:

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

d. Reload **`systemd`** to account for updated unit files:

```
# systemctl daemon-reload
```

e. Correct the Dashboard configuration:

Fix Apache Configuration:

The **`openstack-dashboard`** package installs **`/etc/httpd/conf.d/openstack-dashboard.conf`** file, but the Staypuft installer replaces this with the **`/etc/httpd/conf.d/15-horizon_vhost.conf`** file. After upgrading horizon, you will have the following configuration files:

- **`15-horizon_vhost.conf`**
- **`openstack-dashboard.conf`**
- **`openstack-dashboard.conf.rpmnew`**

Ensure you make the following changes:

- Remove the **openstack-dashboard.conf.rpmnew** file:

```
# rm openstack-dashboard.conf.rpmnew
```

- Modify the **15-horizon_vhost.conf** file by replacing:

```
Alias /static "/usr/share/openstack-dashboard/static"
```

with

```
Alias /dashboard/static "/usr/share/openstack-
dashboard/static"
```

Fix Dashboard Configuration:

The **openstack-dashboard** package installs the **/etc/openstack-dashboard/local_settings** file. After an upgrade, you will find the following configuration files:

- **/etc/openstack-dashboard/local_settings**
- **/etc/openstack-dashboard/local_settings.rpmnew**

Ensure you make the following changes:

- Backup your existing **local_settings** file:

```
# cp local_settings local_settings.old
```

- Rename the **local_settings.rpmnew** file to **local_settings** file:

```
# mv local_settings.rpmnew local_settings
```

- Replace the following configuration options with the corresponding value from your **local_settings.old** file:
  - ALLOWED_HOSTS
  - SECRET_KEY
  - CACHES
  - OPENSTACK_KEYSTONE_URL
- Restart the web server on all your controllers to apply all changes:

```
# service httpd restart
```

f. Clean up the Dashboard resource in Pacemaker:

```
# pcs resource cleanup horizon-clone
```

g. Restart the Dashboard resource in Pacemaker:

```
# pcs resource enable horizon-clone
```

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

Upgrade Compute hosts (nova):

On each compute host:

a. Stop all OpenStack services on the host:

```
# openstack-service stop
```

b. Upgrade all packages:

```
# yum upgrade
```

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

Before starting Kilo controller or compute services, you need to set the **`compute`** option in the **`[upgrade_levels]`** section of **`nova.conf`** to **`juno`**:

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

You need to make this change on your controllers and on your compute hosts.

d. Start all openstack services on the host:

```
# openstack-service start
```

e. After you have upgraded all of your hosts to Kilo, 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
```

Post-upgrade tasks:

a. 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.

b. Review the resulting configuration files. The upgraded packages will have installed **`.rpmnew`** files appropriate to the Red Hat Enterprise Linux OpenStack Platform 7 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 Enterprise Linux OpenStack Platform Documentation Suite.