# Upgrading OpenStack by Updating All Services Simultaneously

Actualizado el hace 11 horas

## 1. Overview

This OpenStack upgrade method involves disabling all the OpenStack services at the same time, performing the upgrade, then enabling all services back up after the upgrade process is complete. Of all methods for upgrading to Red Hat Enterprise Linux OpenStack Platform 5, this one is the simplest. Because all OpenStack services are down, no orchestration is required. 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 v6.4 to v6.5).
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 document follow the architectural naming convention followed by all Red Hat Enterprise Linux OpenStack Platform documentation. If you are unfamiliar with this convention, refer to this link (https://access.redhat.com/articles/1177953#Architecture6) before proceeding.

## 2. RHN/CDN Channels

This section discusses channel and repository settings required for deploying Red Hat Enterprise Linux OpenStack Platform 5.

**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 5. For example, for CDN, unsubscribe from or disable the following:

- Red Hat OpenStack 1.0 (Essex) -- rhel-server-ost-6-preview-rpms
- Red Hat OpenStack 2.1 (Folsom) -- rhel-server-ost-6-folsom-rpms
- Red Hat Enterprise Linux OpenStack Platform 3 (Grizzly) -- rhel-server-ost-6-3-rpms
- Red Hat Enterprise Linux OpenStack Platform 4 Beta (Havana) -- rhel-6-server-openstack-beta-rpms
- Red Hat Enterprise Linux OpenStack Platform 4 (Havana) -- rhel-6-server-openstack-4.0-rpms

**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 6, run:

```
# subscription-manager repos \
   --enable=rhel-6-server-rh-common-rpms
```

### 2.1. Content Delivery Network (CDN) Channels

You can install Red Hat Enterprise Linux OpenStack Platform 5 through the Content Delivery Network. 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]
```

**Table 1. Required Channels**

| Channel | Repository Name |
|---|---|
| Red Hat OpenStack 5.0 Beta (RPMS) for Server 6 | rhel-6-server-openstack-5.0-rpms |
| Red Hat Enterprise Linux 6 Server (RPMS) | rhel-6-server-rpms |

**Table 2. Optional Channels**

| Channel | Repository Name |
|---|---|
| RHEL Server Load Balancer (v6 for 64-bit x86_64) | `rhel-lb-for-rhel-6-server-rpms` |
| Red Hat Enterprise Linux 6 Server - Optional | `rhel-6-server-optional-rpms` |

If you are using CDN, the following channels must be disabled to ensure Red Hat Enterprise Linux OpenStack Platform 5 functions correctly.

**Table 3. 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"` |
| Red Hat Software Collections | `"rhel-server-rhscl-6-rpms"` |

## 2.2. Red Hat Network (RHN) Channels

You can install Red Hat Enterprise Linux OpenStack Platform 5 through Red Hat Network (RHN).
Run the following to add a channel via RHN:

```
# rhn-channel --add --channel=[reponame]
```

Run the following to remove a channel via RHN:

```
# rhn-channel --remove --channel=[reponame]
```

**Table 4. Required Channels**

| Channel | Repository Name |
|---|---|
| Red Hat OpenStack 5.0 for RHEL 6 Server x86_64 | `rhel-x86_64-server-6-ost-5` |
| Red Hat Enterprise Linux Server (v6 for 64-bit AMD64 / Intel64) | `rhel-x86_64-server-6` |

**Table 5. Optional Channels**

| Channel | Repository Name |
|---|---|
| RHEL Server Load Balancer (v6 for 64-bit x86_64) | `rhel-x86_64-server-lb-6` |
| RHEL Server Optional (v. 6 64-bit x86_64) | `rhel-x86_64-server-optional-6` |
| MRG Messaging v2 (for RHEL 6 Server x86_64) | `rhel-x86_64-server-6-mrg-messaging-2` |

# 3. Upgrade All OpenStack Services Simultaneously

To upgrade everything at once, run the following steps on all of your hosts:

**Procedure 1. Upgrading OpenStack components on a host**

1. Ensure the `openstack-utils` package is installed:

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

2. Take down all OpenStack services on all the nodes. This step depends on how your services are distributed among your nodes. To stop all the OpenStack services running on a node, log in to the node and run:

```
# openstack-service stop
```

3. 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
# keystone-manage token_flush
```

4. Upgrade the database schema for each service that uses the database. To do so, log in 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 6. 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 |
| Dashboard | horizon |

5. Review the resulting configuration files. The upgraded packages will have installed `.rpmnew` files appropriate to the Red Hat Enterprise Linux OpenStack Platform 5 version of the service.

## 4. Configure VIF Plugging

As part of the upgrade process, 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
```

Next, you will have to configure each Compute node to fail an instance boot if VIF plugging fails. To do so, run the following commands on each Compute node:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT vif_plugging_is_fatal true
# openstack-config --set /etc/nova/nova.conf DEFAULT vif_plugging_timeout VIFTIMEOUT
```

Replace *VIFTIMEOUT* with the time (in seconds) to wait before failing a VIF plugging attempt. We recommend a *VIFTIMEOUT* of `300`.

## 5. Configure Interactions Between Compute and Networking Services

The OpenStack Networking service in this release can now provide active notifications to the Compute service in response to requests to configure new interfaces. To configure this, perform the following procedure in the Networking controller node:

**Procedure 2. Configuring neutron and nova interactions**

1. Configure `neutron` to notify `nova` when the port status is active:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT notify_nova_on_port_status_changes true
```

2. Configure `neutron` to use the new address of your Compute API service:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT nova_url http://COMPUTEAPIADDRESS:8774/v2
```

Replace *COMPUTEAPIADDRESS* with the host name or IP address of the Compute API service (`openstack-nova-api`). In most deployments, this service is also hosted in the Compute controller node.

3. Provide `neutron` with the required credentials:

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT nova_admin_username NOVAADMIN
```

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT nova_admin_password NOVAPASSWD
```

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT nova_admin_tenant_id NOVATENANT
```

```
# openstack-config --set /etc/neutron/neutron.conf DEFAULT nova_admin_auth_url KEYSTONEENDPOINT
```

Where:

- *NOVAADMIN* is a user with administrative rights within the OpenStack environment.

  **Note**

  The Deploying OpenStack: Learning Environments (Manual Set Up) (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/5/html/Installation_and_Configuration_Guide/index.html) contains instructions on how to configure Compute service authentication (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/5/html/Installation_and_Configuration_Guide/sect-Install_a_Compute_Node.html#Creating_the_Compute_Service_Endpoint). Here, a user named `compute` with the role `admin` and tenant `services` is created specifically for the Compute service. This user is appropriate as *NOVAADMIN*.
- *NOVAPASSWD* is the corresponding password of *NOVAADMIN*.

  **Note**

  If you cannot find or determine the password of an existing *NOVAADMIN*, you can create a new administrative user instead. For instructions, see https://access.redhat.com/articles/1230113 (https://access.redhat.com/articles/1230113).
- *NOVATENANT* is the UUID of the primary tenant associated with *NOVAADMIN*.
- *KEYSTONEENDPOINT* is the administrative endpoint URL of the `keystone` service.

For more details on available settings relating to `neutron` and `nova` interaction, consult the `/etc/neutron/neutron.conf.rpmnew` file generated after upgrading OpenStack Networking.

## 6. Migrate OpenStack Networking Agent to ML2

The Open vSwitch (OVS) and LinuxBridge plug-ins for OpenStack Networking are deprecated in this release. As such, if your previous OpenStack Networking deployment used either plug-in, you will have to migrate the plug-in's database to ML2 after completing the upgrade. This release includes a python script that allows you to do so, namely **neutron.db.migration.migrate_to_ml2**.

**Warning**

Database changes applied by this script are irreversible. As such, back up your plug-in's database before attempting to test or use the script.
Before migrating to ML2, you will first have to set up the initial ML2 configuration. To do so:

**Procedure 3. Setting up initial ML2 configuration**

1. Install the openstack-neutron-ml2 package:

```
# yum install openstack-neutron-ml2
```

2. Create a symbolic link to direct Networking to the ML2 config file `ml2_conf.ini`:

```
# ln -s /etc/neutron/plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

3. Add the required plug-in settings to the `[ml2]` section of the `ml2_conf.ini` file:

**Table 7. Required plug-in settings for ML2**

| Configuration option | Description |
|---|---|
| mechanism_drivers | An ordered, comma-delimited list of networking mechanism driver entrypoints to be loaded from the neutron.ml2.mechanism_drivers namespace. Supported values are `local`, `flat`, `vlan`, `gre`, and `vxlan`. |
| tenant_network_types | An ordered, comma-delimited list of `network_types` to allocate as tenant networks. Supported values are `openvswitch`, `linuxbridge`, and `l2population`. |
| type_drivers | Comma-delimited list of network type driver entry points to be loaded from the neutron.ml2.type_drivers namespace. Supported values are the same as `mechanism_drivers`. |

4. Add the required settings for your chosen mechanism driver. Refer to the following table for the required setting/s per network type driver (`type_drivers`):

**Table 8. Required settings per driver type**

| type_driver | Section | Configuration option | Description |
|---|---|---|---|
| flat | [ml2_type_flat] | flat_networks | List of physical_network names with which flat networks can be created. Use `*` to allow flat networks with arbitrary `physical_network` names. |
| gre | [ml2_type_gre] | tunnel_id_ranges | Comma-separated list of *MIN:MAX* tunneling tuples enumerating ranges of GRE tunnel IDs that are available for tenant network allocation |
| vlan | [ml2_type_vlan] | network_vlan_ranges | List of physical network names usable for VLAN provider and tenant networks; each name in the list can include ranges of VLAN tags available for allocation to tenant networks. The syntax for each name in the list can be in the form *physical_network:vlan_min:vlan_max* or *physical_network*. |
| vxlan (1 of 2) | [ml2_type_vxlan] | vni_ranges | Comma-separated list of *vni_min:vni_max* tuples enumerating ranges of VXLAN VNI IDs that are available for tenant network allocation |
| vxlan (2 of 2) | [ml2_type_vxlan] | vxlan_group | Multicast group for VXLAN. If unset, disables VXLAN multicast mode. |

**Note**

If you need more information about setting up the ML2 plug-in, refer to Modular Layer 2 (ML2) Overview (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/5/html/Installation_and_Configuration_Guide/sect-OpenStack_Networking_Installation_Overview.html#Modular_Layer_2_ML2_Overview)
.
Once you have created the initial ML2 configuration, you can then migrate your old Networking agent and complete the Networking configuration update:

**Procedure 4. Migrating to ML2**

1. To migrate the plug-in database using **neutron.db.migration.migrate_to_ml2**, run:

```
# python -m neutron.db.migration.migrate_to_ml2 PLUGIN mysql://DBADMIN:DBPASS@DBHOST/DBNAME
```

Where:
- *PLUGIN* is the original plug-in used. Supported values are `linuxbridge` and `openvswitch`.
- *DBADMIN* is a valid admin username to the OpenStack Networking database (for example, `neutron`).
- *DBPASS* is the corresponding password of *DBADMIN*.
- *DBHOST* is the IP address of the OpenStack Networking database host.
- *DBNAME* is the name of the OpenStack Networking database. By default, PackStack assigns the database name `ovs_neutron` or `neutron_linux_bridge`, depending on the plug-in selected.

**Note**

The actual values used by the Networking service for *DBADMIN*, *DBPASS*, *DBHOST*, and *DBNAME* are set in the `[database]` section of the `/etc/neutron/neutron.conf`. Specifically, these values are configured in the `connection` setting with the following syntax:

```
connection = mysql://DBADMIN:DBPASS@DBHOST/DBNAME
```

Optional parameters include:
- **`--tunnel-type TUNNEL`**: if you configured your plug-in to use tunneling, then use this parameter to also migrate tunneling data across. Replace *TUNNEL* with the type of tunneling you configured; supported values are `GRE` and `VXLAN`.
- **`--vxlan-udp-port VXLANPORT`**: if you are migrating VXLAN tunneling data, then you can use this parameter to supply the UDP port that VXLAN tunnels should use (replacing *VXLANPORT* accordingly). If you do not use this parameter with **`--tunnel-type VXLAN`**, then VXLAN tunnels will use port `4789` by default.

2. After migrating the plug-in database, manually enable the ML2 plug-in. To do so, first open the `/etc/neutron/neutron.conf` configuration file and set the *`core_plugin`* parameter to the ML2 plug-in:

```
core_plugin = neutron.plugins.ml2.plugin.Ml2Plugin
```

3. Then, add the L3 router plugin (`neutron.services.l3_router.l3_router_plugin.L3RouterPlugin`) to the *`service_plugins`* parameter:

```
service_plugins = neutron.services.firewall.fwaas_plugin.FirewallPlugin,neutron.services.l3_router.l3_router_plugin.L3RouterPlugin
```

Both parameters are in the `DEFAULT` section of `/etc/neutron/neutron.conf`. When manually setting configuration files, ensure that each line has no whitespace at the beginning; these could cause parsing errors and prevent services from functioning properly.

**Note**

For more information about manually configuring the ML2 plug-in, see Set the Networking Service Plug-in (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/5/html/Installation_and_Configuration_Guide/sect-Configure_the_Networking_Service.html#Setting_the_Plug-in) in *Deploying OpenStack: Learning Environments (Manual Set Up)*.

## 7. Finalize OpenStack Networking Upgrade

After migrating the Networking Agent to ML2 (as in Section 6, "Migrate OpenStack Networking Agent to ML2" (#Configure_OpenStack_Networking_During_Upgrades)), you can now add the final configuration changes to complete the Networking service upgrade.

**Procedure 5. Configuring the OpenStack Networking service to complete its upgrade**

1. Set a longer heartbeat for all Networking service agents. The following commands set the recommended values for this release.
   On the Networking controller node, run:

   ```
   # openstack-config --set /etc/neutron/neutron.conf DEFAULT agent_down_time 75
   ```

   Then, on each node hosting a Networking service agent, run:

   ```
   # openstack-config --set /etc/neutron/neutron.conf agent report_interval 30
   ```

   With these settings, the Networking service will check every 30 seconds whether an agent is down; the agent is only declared `down` if it is unreachable for 75 seconds.
2. Kill the `dnsmasq` process on the Networking controller node (this is required for the DHCP agent to work):

   ```
   # killall dnsmasq
   ```

3. Restart the OpenStack Networking service to apply all the settings. To do so, run the following command on all nodes hosting Networking services:

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

## 8. Configure the Firewall to Allow Console Access to Instances

Some firewall configurations may prevent console access after upgrades. If this is the case, you will need to configure the firewall on all Compute nodes that host instances. To do so, perform the following procedure on all Compute nodes.

**Note**

In a single-node deployment, instances are hosted on the controller node. In such cases, perform the following procedure on the controller node if the firewall is preventing access to instances.

**Procedure 6. Configuring the firewall to allow Compute Service traffic**

1. Log in as the `root` user to the Compute node.
2. Open the `/etc/sysconfig/iptables` file in a text editor.
3. Add an INPUT rule allowing TCP traffic on ports used for console traffic by adding the following line to the file:

   ```
   -A INPUT -p tcp --dport 5900:6900 -j ACCEPT
   ```

   The new rule must appear before any INPUT rules that REJECT traffic.
4. Save the changes to the `/etc/sysconfig/iptables` file.
5. Restart the `iptables` service to ensure that the change takes effect.

   ```
   # service iptables restart
   ```

**Note**

You can safely ignore any failures relating to unloading modules, for example:

```
# service iptables restart
  iptables: Setting chains to policy ACCEPT: mangle nat filte[  OK  ]
  iptables: Flushing firewall rules:                 [  OK  ]
  iptables: Unloading modules:  iptable_nat iptable_filter ip[FAILED]t iptable_filter ip_tables
  iptables: Applying firewall rules:                 [  OK  ]
```

The firewall will not reload modules which are already in use, but this should not prevent the required firewall rules from being applied.

## 9. Manually Configure Dashboard Local Settings

In general, the Red Hat Enterprise Linux OpenStack Platform 5 services will run using the configuration files from your older deployment. However, because Dashboard's file was substantially changed between versions, it must be manually configured before its services will work correctly:

**Procedure 7. Manually configuring dashboard local settings for upgrade**

1. Back up your existing `/etc/openstack-dashboard/local_settings` file.
2. Replace the contents of `/etc/openstack-dashboard/local_settings` with the contents of `local_settings.rpmnew`.
3. Update your new `/etc/openstack-dashboard/local_settings` file with the `OPENSTACK_HOST` and `SECRET_KEY` values from your backup `/etc/openstack-dashboard/local_settings`.
   If you have any other required, non-default settings from the backup `/etc/openstack-dashboard/local_settings` file, copy them to the new `/etc/openstack-dashboard/local_settings` file as well.
4. If you are running Django 1.5 (or later), you must ensure that there is a correctly configured `ALLOWED_HOSTS` setting in your `local_settings` file. `ALLOWED_HOSTS` contains a list of host names that can be used to contact your Dashboard service:
   - If people will be accessing the Dashboard service using "http://dashboard.example.com", you would set:

     ```
     ALLOWED_HOSTS=['dashboard.example.com']
     ```

   - If you are running the Dashboard service on your local system, you can use:

     ```
     ALLOWED_HOSTS=['localhost']
     ```

   - If people will be using IP addresses instead of, or in addition to, hostnames, an example might be:

     ```
     ALLOWED_HOSTS=['dashboard.example.com', '192.168.122.200']
     ```

   **Note**

   For more information about the `ALLOWED_HOSTS` setting, see the Django Documentation (https://docs.djangoproject.com/en/1.5/ref/settings/#allowed-hosts) .
5. Restart the web server to apply all changes:

   ```
   # service httpd restart
   ```

## 10. Upgrade the Database Server

Starting with Red Hat Enterprise Linux OpenStack Platform 5.0, the supported database server for Red Hat Enterprise Linux OpenStack Platform is MariaDB. If you are upgrading from Red Hat Enterprise Linux OpenStack Platform 4.0 to Red Hat Enterprise Linux OpenStack Platform 5.0 and your database server is installed on a system running Red Hat Enterprise Linux 6.5 or earlier, you must manually upgrade your database server from MySQL to MariaDB.

**Important**

This procedure must be performed after you have disabled the Red Hat OpenStack 4.0 channel and enabled the Red Hat OpenStack 5.0 for Server 6 channel on the machine on which the database server is installed.

**Procedure 8. Upgrading the Database Server**

1. From each node in your Red Hat Enterprise Linux OpenStack Platform environment, stop all Red Hat Enterprise Linux OpenStack Platform services that access the database server.
2. On the system on which the database server is installed, stop the database server:

   ```
   # service mysqld stop
   ```

3. Remove the MySQL server package:

   ```
   # yum -y remove mysql-server
   ```

4. Ensure all packages are up to date:

   ```
   # yum -y update
   ```

5. Install the MariaDB server packages:

   ```
   # yum -y install mariadb-galera-server
   ```

6. Configure the database server to start at boot time:

```
# chkconfig mysqld on
```

7. Start the database server:

```
# service mysqld start
```

8. Upgrade the internal `mysql` database schema, entering the password for the administrative user when prompted:

```
# mysql_upgrade -u root -p
```

9. From each node in your Red Hat Enterprise Linux OpenStack Platform environment, start all Red Hat Enterprise Linux OpenStack Platform services that access the database server.

You have upgraded your database server from MySQL to MariaDB. All previous commands used to interact with the database server can be used as is.

## 11. Restart OpenStack

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

```
# openstack-service start
```

## 12. RabbitMQ and QPid

RabbitMQ is now the recommended message broker for Red Hat Enteprise Linux OpenStack Platform 5. As such, you may want to migrate to RabbitMQ after upgrading.
For instructions on how to do so, refer to https://access.redhat.com/articles/1167113 (https://access.redhat.com/articles/1167113) .

---

Tipo de articulo General Producto Red Hat Enterprise Linux OpenStack Platform Etiquetas openstack upgrade

---

## Comentarios

---