connect • monitor • manage

CONNECT - ExpressConfig Solution Design Guide

OpenStack Cloud Convergence with Emulex OCe14000 Ethernet Adapters

ExpressConfig

Configuring OpenStack Cloud with Emulex OneConnect® OCe14000 Adapters

ExpressConfig Solution Design

EMULEX®

Table of contents

The Emulex ExpressConfig™ solutions

This solution design guide is part of the first enterprise solution blueprint, ExpressConfig for OpenStack core networks, in the family of Emulex ExpressConfig Solutions, designed to help IT professionals optimize I/O connectivity for cloud, hyperscale and enterprise environments. By providing the necessary architecture, components, features, analysis, configuration settings and software drivers, Emulex ExpressConfig Solutions help customers get the most value out of their pre-designed integrated system. The Emulex ExpressConfig Solutions are designed for use with Emulex OneConnect® OCe14000 family of 10Gb and 40Gb Ethernet (10GbE and 40GbE) Network Adapters and Converged Network Adapters (CNAs), and provide powerful tools, information and features that enable enterprises to save time, lower costs and increase confidence as they scale their cloud and hyperscale environments.

Executive summary

The Emulex OneConnect OCe14000 family of Ethernet 10GbE and 40GbE Network Adapters are optimized for virtualized data centers that have increased demand for accommodating multiple tenants in cloud computing applications. This document provides an easy to follow blueprint leveraging unique Emulex I/O connectivity capabilities for allocating bandwidth, converging multiple protocols, and safely isolating OpenStack core networks or applications. This step-by-step solution design guide will help you to configure OpenStack (Icehouse release) in Red Hat Enterprise 6.5 with Emulex OneConnect OCe14100 10GbE adapters using Emulex Network Interface Card (NIC) partitioning technology.

Intended audience – System and network architects and administrators

OpenStack overview

OpenStack is a free set of software and tools for building and managing cloud computing environments for public and private clouds. As described by the OpenStack Foundation:

> "OpenStack aims to produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable."

OpenStack is considered a cloud operating system that has the ability to control large pools of compute, networks and storage resources throughout a data center. OpenStack provides the following capabilities:

- Networks
- Virtual machines (VMs) on demand
- Storage for VMs and arbitrary files
- Multi-tenancy

Emulex Universal Multi-Channel™ overview

Universal Multi-Channel (UMC) is an adapter partitioning technology developed by Emulex that provides powerful traffic management and provisioning capabilities such as dynamic rate control, priorities, MAC configuration, and Virtual Local Area Network (VLAN) assignment. With UMC, the physical functions are presented to an operating system or hypervisor as independent adapters. UMC channels are presented to the operating system as a physical port with separate MAC address and bandwidth assignments.

Key UMC features include the following:

- Up to 16 physical functions per adapter on the Emulex OneConnect OCe14000 family of 10GbE and 40GbE Network Adapters
- Supports a mix of NIC, iSCSI and Fibre Channel over Ethernet (FCoE) protocols
- Channels can be allocated for separate storage, management, VM migration or external networks
- Bandwidths for each function can be specified as a percentage of the full port speed for the adapter
- Each channel can be set up to use different Tx/Rx queues
- VLANs can be isolated by mapping them to different channels
- The UMC function is transparent to the operating system (OS), hypervisor and network devices

OpenStack server hardware requirements

A basic functional OpenStack environment doesn't require a significant amount of resources. For example:

- The controller node can have one processor, 2 GB memory and 5GB storage
- The network node can have one processor, 512 MB memory, and 5GB storage
- The compute node can have one process, 2 GB memory, and 10GB storage

Although it's possible to set up OpenStack using minimum compute resources like above, it's highly recommended that your environment exceeds these minimums. In addition, you should only install a 64-bit version of your distribution to prevent problems in starting 32-bit instances. If a 32-bit version of your distribution is installed on the compute node, attempting to start an instance using a 64-bit will fail. The servers used in the OpenStack environment described in this document consist of the following:

- The controller node: two 16-core processors, 64GB memory and 1TB storage
- The network node: two 16-core processors, 64MB memory, and 500GB storage
- The compute node: two 16-core processors, 96GB memory, and 1TB storage

Emulex OCe14000 Network Adapter requirements

- OCe14101-NM, 10GbE, 1 port, short reach optical
- OCe14101-NX, 10GbE, 1 port, direct attach copper
- OCe14102-NM, 10GbE, 2 port, short reach optical
- OCe14102-NX, 10GbE, 2 port, direct attach copper
- OCe14401-NX, 40GbE, 1 port, direct attach copper

Note – Verify that you installed the most recent Emulex adapter driver and firmware.

Emulex Downloads and Documentation: www.emulex.com/downloads

Configure Emulex adapters with UMC

Servers deployed with Emulex Oce14000 10GbE and 40GE adapters can utilize Emulex UMC technology for an OpenStack network providing both traffic separation and bandwidth optimization. UMC provides data centers with significant cost savings for cabling, adapters, switches, and power. UMC is switch-agnostic and works with any 10GbE or 40GbE switch. OpenStack network traffic can be configured with unique VLAN assignments where each channel has its own independent broadcast and multicast domain. You can isolate OpenStack management, instance tunnels and external traffic on a single 10GbE or 40GbE adapter port. The range for a UMC Logical Port VLAN ID (LPVID) is 2-4094 and the switch must be configured to correspond with the adapter VLAN assignments. Each UMC channel provides separate transmit and receive queues, and bandwidth can be configured across all configured UMC channels for a maximum of 100 percent. UMC can provide up to 16 NIC functions per adapter depending on the adapter model. In this document, however, a minimum number of NIC functions assignments are defined for a basic OpenStack three-node solution shown in Figure 1.

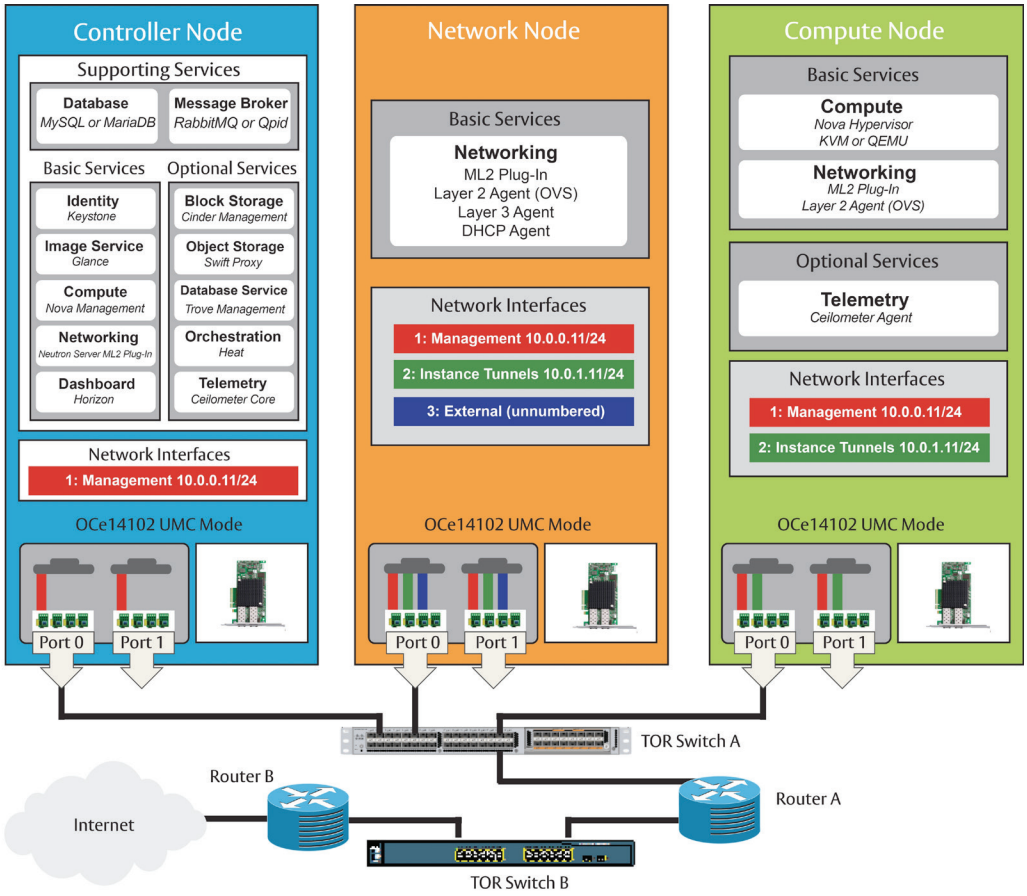


Figure 1. Three-node OpenStack solution.

UMC partitioning can be configured from a separate Windows or Linux host using Emulex OneCommand® Manager graphical user interface (GUI) or command line interface (CLI). You must first install the OneCommand Manager Core Kit on all

OpenStack nodes and select the desired mode of operation to allow management from a remote host in your environment. To enable UMC mode from BIOS. enter <CNTRL P> when prompted to access the UMC menu. See Figures 2 - 6:

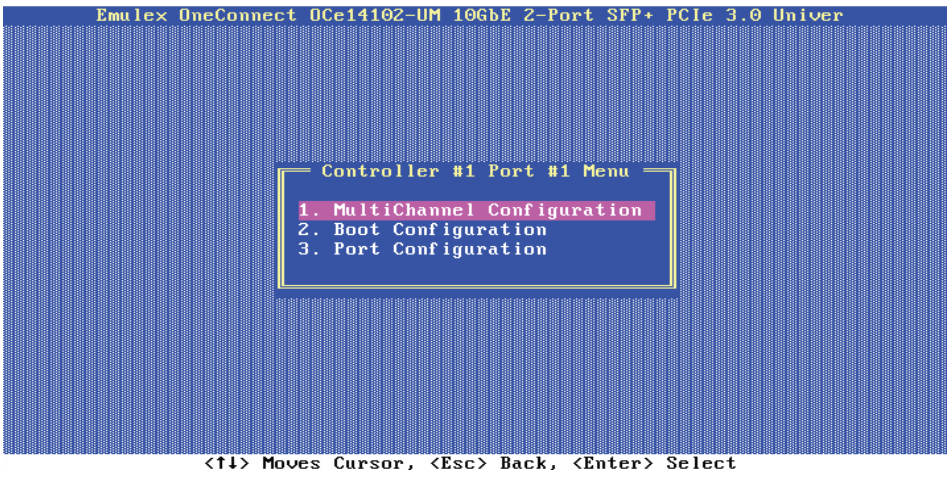


Figure 2. Multi-Channel configuration memo (BIOS).

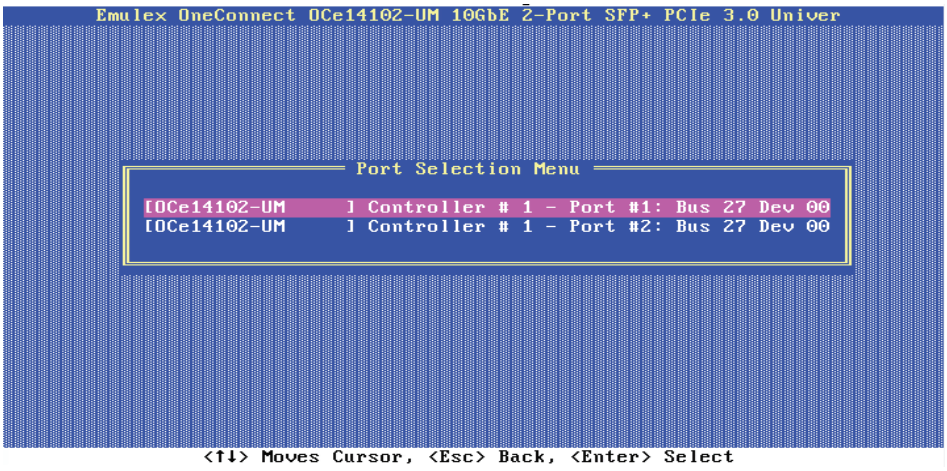


Figure 3. Select controller to be configured for UMC.

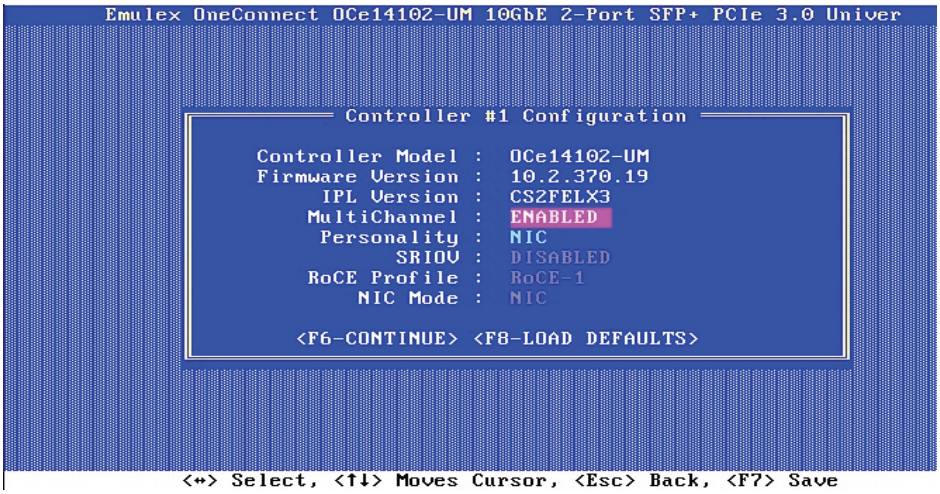


Figure 4. Enable UMC.

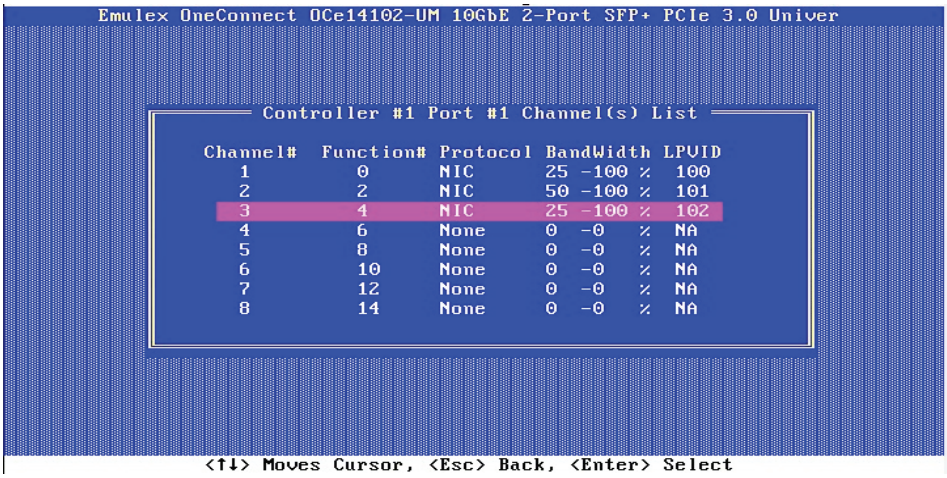


Figure 5. Configure bandwidth for NICs.

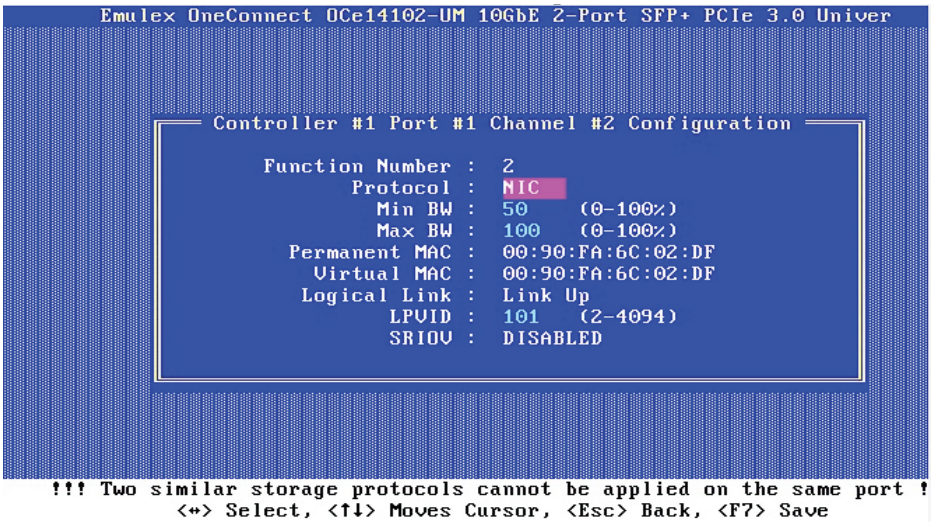


Figure 6. Configure LPVID (2-4094) for each NIC channel and verify the link is up.

You can also enable OneCommand Manager by selecting the UMC redial button in the OneCommand Manager GUI, however, this will require you to reboot the server. See Figure 7.

For additional information, the following reference documents are available at www.emulex.com:

- Emulex Universal Multi-Channel Reference Guide for instructions on how to configure UMC, including LPVIDs and bandwidth requirements.
- Emulex OneCommand Manager Enterprise Application (GUI)
- Emulex OneCommand Manager Core Application (CLI)

Using OneCommand Manager, the UMC configuration of a basic OpenStack network node requiring three separate NICs is shown in Figure 7. For example, for physical Port 0:

- **function 0** is for management allocated at 10% bandwidth
- **function 2** is for instance tunnels allocated at 80% bandwidth
- **function 4** is for external network traffic allocated at 10% bandwidth

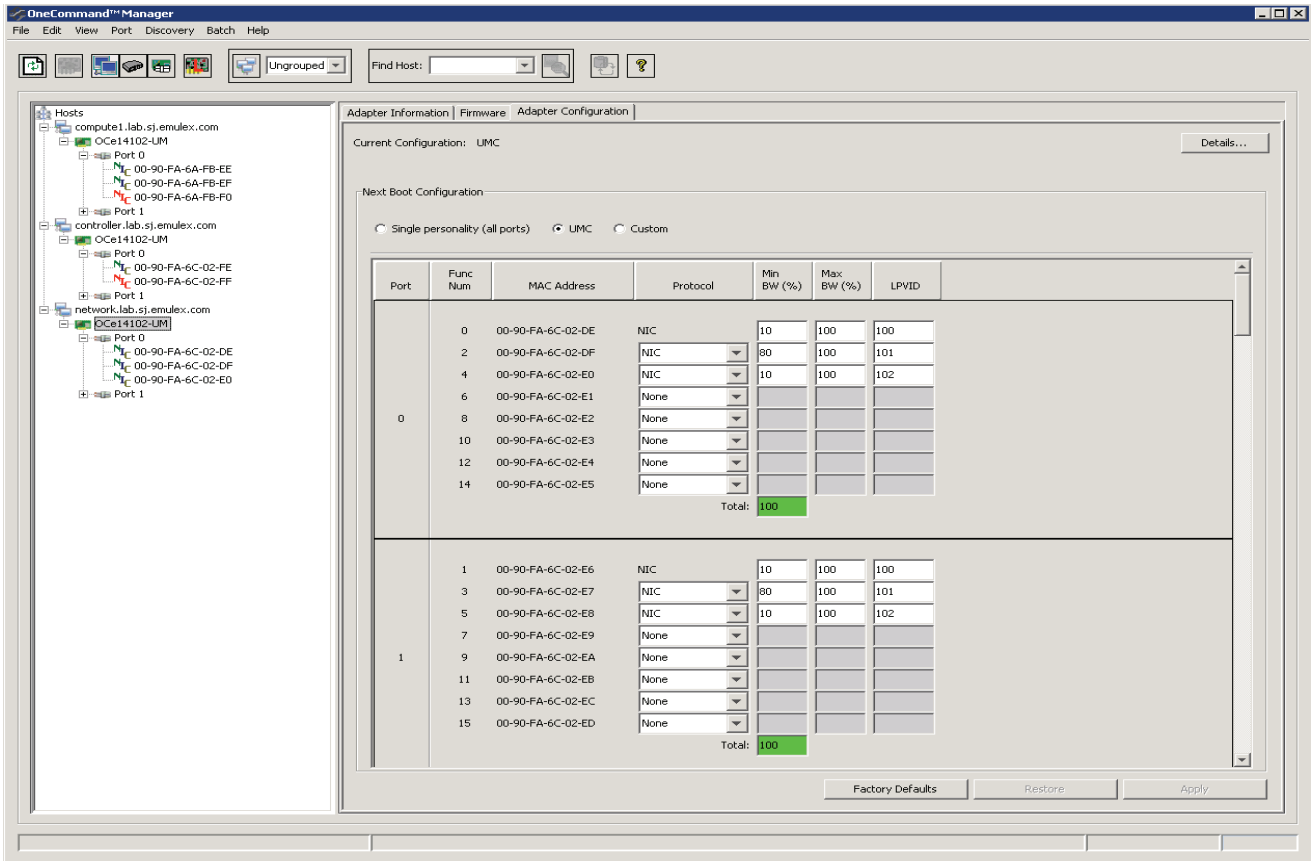


Figure 7. Configure UMC settings from OneCommand Manager.

The ifconfig command will list the final result of the UMC configuration. In Figure 8, UMC was configured on the network node and displays three separate adapters: p6p1, p6p3 and p6p5.

```
p6p1      Link encap:Ethernet  HWaddr 00:90:FA:6C:02:DE
          inet addr:10.193.35.2  Bcast:10.193.35.255  Mask:255.255.255.0
          inet6 addr: fe80::290:faff:fe6c:2de/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:34 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1180 (1.1 KiB)  TX bytes:1772 (1.7 KiB)
          Memory:d5260000-d5280000

p6p3      Link encap:Ethernet  HWaddr 00:90:FA:6C:02:DF
          inet addr:10.4.5.1  Bcast:10.4.5.255  Mask:255.255.255.0
          inet6 addr: fe80::290:faff:fe6c:2df/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:746 (746.0 b)
          Memory:d52e0000-d5300000

p6p5      Link encap:Ethernet  HWaddr 00:90:FA:6C:02:E0
          inet6 addr: fe80::290:faff:fe6c:2e0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:21 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1176 (1.1 KiB)  TX bytes:648 (648.0 b)
          Memory:d5360000-d5380000
```

Figure 8. Network node NIC interfaces in RHEL OS as a result of the UMC configuration.

Table 1 is a complete layout of all adapters and details for all nodes in this OpenStack solution blueprint.

OpenStack UMC network configuration table																			
Controller node							Network node							Compute node					
Traffic type	LPVID	BW	Host device	SW port	MAC	IP address	Traffic type	LPVID	BW	Host device	SW port	MAC	IP address	Traffic type	LPVID	BW	Host device	SW port	MAC	IP address
Mgmt	100	100	P6p1	5	6C-02-FE	10.193.35.1	Mgmt	100	10	P6p1	7	6C-02-DE	10.193.35.2	Mgmt	100	50	Eth1	1	6A-FB-EE	10.193.35.3
N/A	N/A	N/A	N/A	5	N/A	N/A	Data	101	80	P6p3	7	6C-02-DF	10.4.5.1	Data	101	50	Eth3	1	6A-FB-EF	10.4.5.2
N/A	102	N/A	N/A	5	N/A	N/A	Ext	102	10	P6p5	7	6C-02-E0	N/A	N/A	N/A	N/A	N/A	1	N/A	N/A

Table 1. UMC network configuration table.

Physical network layout and switch requirements

Before configuring OpenStack, your physical network requirements will need to be determined, including switches, routers, subnets, IP addresses and VLAN assignments, etc. Please consult with your IT network administrator prior to setting your OpenStack environment.

The example in Figure 9 shows three OpenStack nodes connected to switch 1. Switch 1 ports 1/0/1, 1/0/5 and 1/0/7 are configured as trunks allowing network traffic based on the UMC LPVID configuration. Switch 1 ports 1/0/15 and 1/0/16 are access ports connected to router 1. Router 1 routes OpenStack cloud external traffic on 10.193.36.0 subnet and management traffic on subnet 10.193.35.0. The data network used for instance tunnels is a private network on VLAN 101 and only accessible by network and compute nodes. Finally, external and management networks are routed up to a distribution switch via switch 2. The router and firewall (not shown) provides access to the internet.

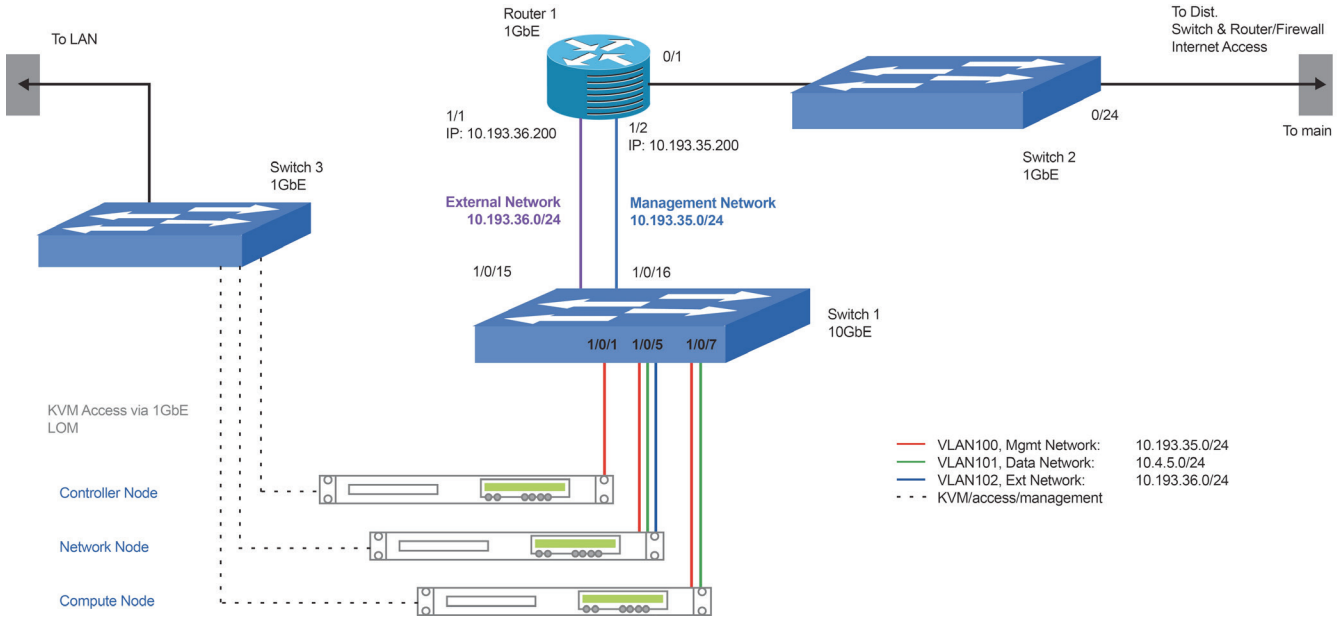


Figure 9. OpenStack cloud network layout.

Deploy a three-node OpenStack environment

Figure 9 in the previous section illustrates a basic three-node OpenStack network configuration. Red Hat Enterprise Linux 6.5 is installed on all three servers. The nodes are accessed via a Secure Shell (SSH) from a remote server. The nodes can also be accessible via KVM over IP (IPKVM) device providing an alternate connection (not shown) to manage the nodes.

The basic controller node runs the Identity service, the Image service, the management portion of Compute, and the Dashboard service necessary to launch a simple instance. It also includes supporting services such as a Database, Message Broker and Network Time Protocol (NTP). Optionally, the controller node also runs portions of Block Storage, Object Storage, Database, Orchestration, and Telemetry. These components provide additional features for your environment.

The basic compute node runs the hypervisor portion of Compute, which operates tenant vms or instances. By default, Compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups. You can run more than one compute node.

As an option, the compute node also runs the Telemetry agent, which provides additional features for your environment.

1. Install the OneConnect Ethernet Adapter in an available PCIe 3.0 x8 slot (all nodes)
2. Install Red Hat Enterprise Linux 6.5
3. If required, install the latest version of the Emulex adapter driver and firmware
4. On server boot, configure UMC LPVIDs, bandwidth, etc.
5. Enter the server BIOs setup and make sure the virtualization technology, Intel® VT-d, is enabled (compute node only).
6. Install a recent version of Emulex OneCommand Core kit (optional)
7. Disable any automated network management tools, such as NetworkManager
8. Disable firewalld servers. RHEL6.5 combines both firewalld and iptable services. If you decide to use iptable services, you must configure these based on your network requirements.
9. Edit the appropriate files for you distribution. In some cases you may have to register your system using the Red Hat Subscription Manager to install packages using Yellowdog Updater, Modified (yum). You may have to use other repositories (repos), for example, epel.repo, rpmforge.repo.rpmnew, puppetlabs.repos, rpmforge.repo, etc.
10. Configure the adapter interface files found in /etc/sysconfig/network-scripts/ifcfg-p6p1. Below are example settings used on the OpenStack controller interface:

```
DEVICE=p6p1
HWADDR=00:90:FA:6C:02:FE
TYPE=Ethernet
UUID=b2e95c81-1052-451e-bb78-f2dc61f489f7
ONBOOT=yes
NM_CONTROLLED=no
BOOTPROTO=no
IPV6INIT=no
USERCTL=no
IPADDR=10.193.35.1
NETMASK=255.255.255.0
GATEWAY=10.193.35.200
```

11. Configure the name resolution on controller, network, and compute nodes. Edit /etc/hosts file on each node by adding all three node names and IP addresses. Below is an example of how to configure the /etc/hosts file:

```
# controller node
10.193.35.1     controller.lab.sj.emulex.com  controller

# network node
10.193.35.2     network.lab.sj.emulex.com     network

# compute1 node
10.193.35.3     compute1.lab.sj.emulex.com compute1
```

12. Enter your DNS IP requirements in the resolve.conf file.

13. Restart networking:

```
[root@controller]#   service network restart
[root@network]#   service network restart
[root@compute1]#  service network restart
```

Note – OpenStack doesn't require an IP address configured for external networks for controller, compute and network nodes. However, you should use the ifconfig command to configure a temporary IP address on the network node external interface to ensure you have a connection to northbound switches and routers in your network. Once you validate the connection on the external interface, remove the IP address.

Use the ping command to validate all other network connections between nodes as well as outside the OpenStack infrastructure as follows:

1. From the controller node, ping openstack.org (check ext. access)

```
[root@controller ~]#   ping -c 2 openstack.org
PING openstack.org (192.237.193.83) 56(84) bytes of data.
64 bytes from 192.237.193.83: icmp_seq=1 ttl=44 time=49.6 ms
64 bytes from 192.237.193.83: icmp_seq=2 ttl=44 time=46.5 ms
```

2. From the controller node, ping the network node management interface:

```
[root@controller ~]#   ping -c 2 network
PING network (10.193.35.2) 56(84) bytes of data.
64 bytes from network (10.193.35.2): icmp_seq=1 ttl=64 time=1.39 ms
64 bytes from network (10.193.35.2): icmp_seq=2 ttl=64 time=0.094 ms
```

3. From the controller node, ping compute1 node management interface:

```
[root@controller ~]#   ping -c 2 compute1
PING compute1 (10.193.35.3) 56(84) bytes of data.
64 bytes from compute1 (10.193.35.3): icmp_seq=1 ttl=64 time=0.538 ms
64 bytes from compute1 (10.193.35.3): icmp_seq=2 ttl=64 time=0.084 ms
```

4. From the network node, ping openstack.org (check external access):

```
[root@network etc]#   ping -c 2 openstack.org
PING openstack.org (192.237.193.83) 56(84) bytes of data.
64 bytes from 192.237.193.83: icmp_seq=1 ttl=44 time=46.9 ms
64 bytes from 192.237.193.83: icmp_seq=2 ttl=44 time=45.5 ms
```

5. From the network node, ping the controller node management interface:

```
[root@network etc]#  ping -c 2 controller
PING controller (10.193.35.1) 56(84) bytes of data.
64 bytes from controller (10.193.35.1): icmp_seq=1 ttl=64 time=0.087 ms
64 bytes from controller (10.193.35.1): icmp_seq=2 ttl=64 time=0.068 ms
```

6. From the network node, ping the instance tunnels interface on the compute node:

```
[root@network network-scripts]#  ping -c 2 10.4.5.2
PING 10.4.5.2 (10.4.5.2) 56(84) bytes of data.
64 bytes from 10.4.5.2: icmp_seq=1 ttl=64 time=0.080 ms
64 bytes from 10.4.5.2: icmp_seq=2 ttl=64 time=0.070 ms
```

7. From the compute node, ping openstack.org on the internet (check external access):

```
[root@compute1 network-scripts]#  ping -c 2 openstack.org
PING openstack.org (192.237.193.83) 56(84) bytes of data.
64 bytes from 192.237.193.83: icmp_seq=1 ttl=44 time=51.1 ms
64 bytes from 192.237.193.83: icmp_seq=2 ttl=44 time=45.7 ms
```

8. From the compute node, ping the controller node management interface:

```
[root@compute1 network-scripts]#  ping -c 2 controller
PING controller (10.193.35.1) 56(84) bytes of data.
64 bytes from controller (10.193.35.1): icmp_seq=1 ttl=64 time=0.082 ms
64 bytes from controller (10.193.35.1): icmp_seq=2 ttl=64 time=0.104 ms
```

9. From the compute node, ping the instance tunnels interface on the network node:

```
[root@compute1 network-scripts]#  ping -c 2 10.4.5.1
PING 10.4.5.1 (10.4.5.1) 56(84) bytes of data.
64 bytes from 10.4.5.1: icmp_seq=1 ttl=64 time=0.565 ms
64 bytes from 10.4.5.1: icmp_seq=2 ttl=64 time=0.103 ms
```

10. Install the NTP server on your controller so that it receives data by modifying ntp.conf file and restart the service. Configure both the compute and network nodes to sync to the controller, as shown in Figure 10.

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
#server 0.rhel.pool.ntp.org iburst
#server 1.rhel.pool.ntp.org iburst
#server 2.rhel.pool.ntp.org iburst
#server 3.rhel.pool.ntp.org iburst

# NTP sync with the controller node or ntp1.emulex.com
server controller.lab.sj.emulex.com iburst
```

Figure 10. NTP server setup.

Configure OpenStack cloud environment

Install MySQL Database

1. Install MySQL client and server packages and Python library on the **controller** node only. .

   root@controller]# **yum install mysql mysql-server MySQL-python**

2. Install MySQL Python library on controller node and all other nodes.

   [root@controller]# **yum install MySQL-python**

Configure controller node

1. For the **controller node** only, use vi editor to modify MySQL configuration to work with OpenStack.

   [root@controller]# **vi /etc/my.cnf**

   [mysqld] datadir=/var/lib/mysql
   socket=/var/lib/mysql/mysql.sock
   user=mysql
   # Disabling symbolic-links is recommended to prevent assorted security risks
   symbolic-links=0
   ---
   default-storage-engine = innodb
   innodb_file_per_table
   collation-server = utf8_general_ci
   init-connect = 'SET NAMES utf8'
   character-set-server = utf8
   bind-address = **10.193.35.1**

   [mysqld_safe]
   log-error=/var/log/mysqld.log
   pid-file=/var/run/mysqld/mysqld.pid

2. Start the mysqld service on the controller node and set it to start automatically when the system boots.

   [root@controller etc]# **service mysqld start**
   [root@controller etc]# **chkconfig mysqld on**

3. Create a password for the MySQL root user.

   [root@controller etc]# **/usr/bin/mysqladmin –u root password 'new-password'**

4. Delete possible anonymous users that are created when the database is first started.

   [root@controller etc]# **mysql_install_db**
   [root@controller etc]# **mysql_secure_installation**

Install OpenStack packages

1. Figure 11 lists the repos used to configure OpenStack Icehouse v3 in RHEL6.5.

```
[root@controller yum.repos.d]# ls
epel.repo          mirrors-rpmforge-extras   redhat.repo                                rpmforge.repo.rpmnew
epel-testing.repo  mirrors-rpmforge-testing  rhel-source.repo                           rpmforge.repo.rpmsave
foreman.repo       puppetlabs.repo           rpmforge-release-0.5.2-2.el6.rf.x86_64.rpm
mirrors-rpmforge   rdo-release.repo          rpmforge.repo
[root@controller yum.repos.d]#
```

Figure 11. Repo list.

2. Install the yum-plugin-priorities. This functionality is used by the RDO release packages.

   [root@controller etc]# **yum install yum-plugin-priorities**

3. Enable the RDO repository, download and install the rdo-release-icehouse package.

   [root@controller etc]# **yum install**
   [root@controller etc]# **yum install** http://repos.fedorapeople.org/repos/openstack/openstack- icehouse/rdo-release-icehouse-3.noarch.rpm

   For RHEL6x, install the EPEL package that includes GPG keys for package signing and repository information:

   [root@controller etc]# **yum install** http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6- 8.noarch.rpm

4. The openstack-utils package contains utility programs that make installation and configuration easier. These programs are used throughout this guide. Install openstack-util. This verifies that you can access the RDO repository:

   [root@controller]# **yum install install openstack-utils**

5. Install the openstack-selinux package that includes required policy files for required to configure SELinux during OpenStack installation on RHEL.

   [root@controller etc]# **yum install openstack-selinux**

6. Upgrade system packages.

   [root@controller etc]# **yum upgrade**

7. The yum upgrade may have included a new kernel package. Reboot to confirm the new kernel is running.

   [root@controller etc]# **reboot**

Install a Message Broker service

1. Coordination of operations and status information is provided by a message broker service such as RabbitMQ, Qpid and ZeroMQ. In this deployment, we used the Qpid message service used by RHEL.

   [root@controller etc]# **yum install qpid-cpp-server**

2. Edit the /etc/qpidd.conf file and change the following key:

   [root@controller etc]# **vi /etc/qpidd**
   ---
   auth=no

3. Start the Qpid Message Broker service and configure to start when the system boots.

   [root@controller etc]# **service qpidd start**
   [root@controller etc]# **chkconfig qpidd on**

Install Identity service (Keystone)

The Identity service (Keystone) is installed on the controller node and performs the following functions:

- User management: Tracks users and their permissions.
- Server catalog: Provides a catalog of available services with their API endpoints. Here you define users, tenants, and roles and create API endpoints. For more information on the concepts, please reference the OpenStack Icehouse Installation Guide.

1. Install Keystone Identity service on the controller.

```
[root@controller etc]#  yum install openstack-keystone python-keystoneclient
```

2. Specify location of MySQL database in the configuration file. Enter a database password.

```
[root@controller etc]#  openstack-config --set /etc/keystone/keystone.conf \
                        database connection mysql://keystone:Password@controller/keystone
```

3. Configure MySQL using a password previously set.

```
[root@controller etc]#  mysql -u root –p

Enter password:  password
        mysql>   CREATE DATABASE keystone;
        mysql>   GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
                 IDENTIFIED BY ' Password';
        mysql>   GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
                 IDENTIFIED BY ' Password';
        mysql>   exit
```

4. Create the database tables for the Identity service

```
[root@controller etc]#  su -s /bin/sh -c "keystone-manage db_sync" keystone
```

5. Define an authorization token to use as a shared secret between the Identity service and other OpenStack services. You can use openssl to generate a random token and store it in the configuration file.

```
[root@controller etc]#  ADMIN_TOKEN=$(openssl rand -hex 10)
[root@controller etc]#  echo $ADMIN_TOKEN
[root@controller etc]#  openstack-config --set /etc/keystone/keystone.conf DEFAULT \ admin_token $ADMIN_TOKEN
```

6. Create signing keys and certificates and restrict access to the generated data.

```
[root@controller etc]#  keystone-manage pki_setup --keystone-user keystone --keystone-group keystone
[root@controller etc]#  chown -R keystone:keystone /etc/keystone/ssl
[root@controller etc]#  chmod -R o-rwx /etc/keystone/ssl
```

7. Start the Identity service and enable it to start when the system boots.

```
[root@controller etc]#  service openstack-keystone start
[root@controller etc]#  chkconfig openstack-keystone on
```

8. Run the following command to purge expired tokens every hour and log the output to /var/log/keystone/keystone- tokenflush.log

```
[root@controller etc]#  (crontab -l -u keystone 2>&1 | grep -q token_flush) || \
                        echo '@hourly /usr/bin/keystone-manage token_flush >/var/log/keystone/
                        keystone-tokenflush.log 2>&1' >> /var/spool/cron/keystone
```

Define user, tenants and roles

After you install the Identity service, you will need to set up users, tenants, and roles to authenticate against and allow access to services and endpoints.

Normally, you would indicate a user and password to authenticate with the Identity service. However, you have not created any users at this point, so you have to use the authorization token previously created in an earlier step.

1. Retrieve the token by running the following command on the controller.

   [root@controller keystone]# **cat keystone.conf | grep admin_token admin_token=28016edb0d838a94d593**

2. Set OS_SERVICE_TOKEN, SERVICE_ENDPOINT to specify where the Identity service is running. :

   [root@controller keystone]# **export OS_SERVICE_TOKEN= 28016edb0d838a94d593**
   [root@controller keystone]# **export OS_SERVICE_ENDPOINT=http://controller:35357/v2.0**

Create an administrative user

The next steps create an administrative user, role and tenant. This account is used for administrative interaction with the OpenStack cloud. By default, the Identity service creates a special _member_ role. The OpenStack Dashboard (Horizon) automatically grants access to users with this role. Give the admin user access to this _member_ role and to the admin role.

1. Create the admin user:

   [root@controller keystone]# **keystone user-create --name=admin --pass=insert your password**

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|   email  |                                  |
|  enabled |              True                |
|    id    | 0e5215dd58ac4656800c20af98f4dbd1 |
|   name   |              admin               |
| username |              admin               |
+----------+----------------------------------+
```

2. Create the admin role:

   [root@controller keystone]# **keystone role-create --name=admin**

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | d1efb1b81cea4240a66daeb683865f35 |
|   name   |              admin               |
+----------+----------------------------------+
```

3. Create the admin tenant:

   [root@controller keystone]# **keystone tenant-create --name=admin --description="Admin Tenant"**

```
+-------------+----------------------------------+
|   Property  |            Value                 |
+-------------+----------------------------------+
| description |         Admin Tenant             |
|   enabled   |             True                 |
|     id      | adf92ed8340f4a488f229b629cc39a86 |
|    name     |             admin                |
+-------------+----------------------------------+
```

4. Link together the admin user, admin role and admin tenant using the user-role-add option.

   [root@controller keystone]# **keystone user-role-add --user=admin --tenant=admin --role=admin**

5. Link the admin user, _member_role, and admin tenant.

   [root@controller keystone]# **keystone user-role-add --user=admin --role=_member_ --tenant=admin**

Create normal users

These users created include user and tenant that are linked to a special _member_role. These accounts are primarily for daily non-administrative interaction with the OpenStack cloud. You can repeat this procedure to create additional cloud users with different usernames and passwords.

1. Create demo user

   [root@controller keystone]# **keystone user-create --name=demo --pass=insert a password**

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |              True                |
|    id    | eb957f654c9b4d969cd796c124660d8d |
|   name   |              demo                |
| username |              demo                |
+----------+----------------------------------+
```

2. Create demo tenant

   [root@controller keystone]# **keystone tenant-create --name=demo --description="Demo Tenant"**

```
+-------------+----------------------------------+
|  Property   |              Value               |
+-------------+----------------------------------+
| description |           Demo Tenant            |
|   enabled   |              True                |
|     id      | ff6b8ae0ff714443956e1baff67c0f84 |
|    name     |              demo                |
+-------------+----------------------------------+
```

   Note: Do not repeat this step when adding additional users.

3. Link the demo user, _member_role, and demo tenant.

   [root@controller keystone]# **keystone user-role-add --user=demo --role=_member_ --tenant=demo**

Create a service tenant

OpenStack services require a username, tenant and role to access other OpenStack services. In a basic installation, OpenStack services usually share a single tenant named service. You will create usernames and roles under this tenant as you install and configure each service.

[root@controller keystone]# **keystone tenant-create --name=service --description="Service Tenant"**

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |              True                |
|     id      | 15b0f87645e8473c81b3060515a21028 |
|    name     |             service              |
+-------------+----------------------------------+
```

Define services and API endpoints

Each service in the OpenStack installation needs to be registered so the Identity service can track which OpenStack services are installed and where they are located on the network. Run the commands listed below to register a service. Since Identity is a service, it must also be registered. Use the OS_SERVICE_TOKEN environment variable, as set previously for authentication.

- Keystone service-create: Describes the service
- Keystone endpoint-create: Associates API endpoints with the service

1. Create a service entry for the Identity service:

```
[root@controller keystone]#  keystone service-create --name=keystone --type=identity \
                              --description="OpenStack Identity"
```

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |        OpenStack Identity        |
|   enabled   |              True                |
|     id      | a9331f8300d54fc1b8957376d5d48645 |
|    name     |             keystone             |
|    type     |             identity             |
+-------------+----------------------------------+
```

Note – The service id is randomly generated and will be different from the one show above.

2. Specify an API endpoint for the Identity service by using the returned service ID. When you specify an endpoint, you provide URLs for the public API, internal API and admin API. This guide uses the controller host name. The Identity service uses a different port for the admin API.

```
[root@controller keystone]#  keystone endpoint-create \
                              --service-id=$(keystone service-list | awk '/ identity / {print $2}') \
                              --publicurl=http://controller:5000/v2.0 \
                              --internalurl=http://controller:5000/v2.0 \
                              --adminurl=http://controller:35357/v2.0
```

```
+-------------+----------------------------------+
|  Property   |              Value               |
+-------------+----------------------------------+
|  adminurl   |   http://controller:35357/v2.0   |
|     id      | 5814857075574b40b7887878d86c5237 |
| internalurl |   http://controller:5000/v2.0    |
|  publicurl  |   http://controller:5000/v2.0    |
|   region    |            regionOne             |
| service_id  | a9331f8300d54fc1b8957376d5d48645 |
+-------------+----------------------------------+
```

You will need to create an additional endpoint for each service added to your OpenStack environment. The installation of each service also includes the endpoint creation step specific to the service.

Verify the Identity service installation

1. To verify that the Identity service is installed and configured correctly, clear the values in the OS_SERVICE_TOKEN and OS_SERVICE_ENDPOINT environment variables.

   [root@controller keystone]# **unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT**

2. Varibles used to bootstrap the administrative user and registrer the Identity service are no long needed. Therefore, you can know use regular name-based authentication. Request an authentication token by using the admin user and the password you selected for that user.

   [root@controller keystone]# **keystone --os-username=admin --os-password=insert your password **
   --os-auth-url=http://controller:35357/v2.0 token-get

```
+-----------+-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
------
|   Property | Value
+-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
---
|   expires  |
2014-10-19T02:00:32Z
|
|   id       | iYZqaBes2cCxjiJJDfdaPUFpo14BFJQXwmoAYmLZixrdqh8l8tUFF1gLwg3xzLewcDHEvJRgyqAD-FYL0kC+FD-
O71vCMbRsQQhbW2jrxysRc+yHCLz-0PG-VMEF41Pl+r5f9jrnGbA==
| tenant_id | adf92ed8340f4a488f229b629cc39a86
|
|   user_id  | 0e5215dd58ac4656800c20af98f4dbd1
```

   The response shown above is a token paired with your user ID. This verifies that the Identity service is running on the expected endpoint and that your user account is established with the expected credentials.

3. Verify that authorization behaves as expected by requesting authorization on a tenant.

   [root@controller keystone]# **keystone --os-username=admin --os-password=insert your password **
   **--os-tenant-name=admin --os-auth-url=http://controller:35357/v2.0 **
   token-get

   In response, you will receive a token that includes the ID of the tenant that you specified. This verifies that your user account has an explicitly defined role on the specified tenant and the tenant exists as expected.

4. Set your --os-* variables in your environment to simplify command line usage. Set up an admin-openrc.sh file with the admin credentials and admin endpoint:

```
export OS_USERNAME=admin
export OS_PASSWORD=PASSWORD
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://controller:35357/v2.0
```

5. Source the admin-openrc.sh file to read in the environment.

   [root@controller keystone]# **source admin-openrc.sh**

6. Verify that your admin-openrc.sh file is configured correctly. Run the same command without the --os-* arguments.

   The keystone token-get command should return with a token and the ID of the specified tenant.

   [root@controller keystone]# **keystone token-get**

   Note – This step may take a few seconds to complete

7. Verify that your admin account has authorization to perform administrative commands.

```
[root@controller keystone]#  keystone user-list
+----------------------------------+---------+---------+-----------------------+
|                id                |   name  | enabled |         email         |
+----------------------------------+---------+---------+-----------------------+
| 0e5215dd58ac4656800c20af98f4dbd1 |  admin  |   True  |                       |
| eb957f654c9b4d969cd796c124660d8d |   demo  |   True  |                       |
+----------------------------------+---------+---------+-----------------------+

[root@controller keystone]#  keystone user-role-list --user admin --tenant admin
+----------------------------------+----------+----------------------------------+----------------------------------+
|                id                |   name   |             user_id              |            tenant_id             |
+----------------------------------+----------+----------------------------------+----------------------------------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ | 0e5215dd58ac4656800c20af98f4dbd1 | adf92ed8340f4a488f229b629cc39a86 |
| d1efb1b81cea4240a66daeb683865f35 |  admin   | 0e5215dd58ac4656800c20af98f4dbd1 | adf92ed8340f4a488f229b629cc39a86 |
+----------------------------------+----------+----------------------------------+----------------------------------+
```

The IDs from the Keystone user-list command match the user_id in the keystone user-role-list command and the admin role is listed for that user for the related tenant. This verifies that your user account has the admin role, which matches the role used in the Identity service policy.json file.

OpenStack client overview

Each client command runs cURL commands that embeds the API request. The OpenStack APIs are RESTful APIs that use the HTTP protocol, including methods, uniform resource identifiers (URIs), media types and response codes.

Python clients run on Linux or Mac OS X systems. Each OpenStack service has its own command-line client. On some client commands, you can specify debug parameters to show the underlying API request for the command. At the time this document was written, an OpenStack common client is in development.

Service	Client	Package	Description
Block Storage	Cinder	python-cinderclient	Create and manage volumes
Compute	Nova	python-novaclient	Create and manage images, instances, and flavors
Database	Trove	python-troveclient	Create and manage databases
Identity	Keystone	python-keystoneclient	Create and manage users, tenants, roles, endpoints, and credentials
Image	Glance	python-glanceclient	Create and manage images
Networking	Neutron	python-neutronclient	Configure networks for guest servers. This client was previously called quantum.
Object Storage	Swift	python-swiftclient	Gather statistics, list items, update metadata, and upload, download, and delete files stored by the Object Storage service. Gain access to an Object Storage installation for ad hoc processing.
Orchestration	Heat	python-heatclient	Launch stacks from templates, view details of running stacks including events and resources, and update and delete stacks.
Telemetry	Ceilometer	Python-ceilometerclient	Create and collect measurements across OpenStack

Table 2. OpenStack services and clients.

Install OpenStack command-line clients

1. Before installing the clients, install the prerequisite software and the Python package for each OpenStack client. Use Python 2.6 or later. (At the time this document was written, Python 3 was not supported.)

   Note – Many Linux distributions provide packages to make setup tools easy to install. Search your package manager for setup tools to find an installation package. If you cannot find one, download the setup tools package directly from http://pypi.python.org/pypi/setuptools.

2. For Red Hat Enterprise Linux, use yum to install the clients. Alternatively, you can use pip to manage client installations, for example:

   ```
   [root@controller keystone]#  yum install python-pip
   [root@controller etc]#  pip install python-novaclient
   ```

3. Install the clients from the package versions available in RDO on the controller node:

   ```
   [root@controller etc]#  yum install python-ceilometerclient
   [root@controller etc]#  yum install python-cinderclient
   [root@controller etc]#  yum install python-glanceclient
   [root@controller etc]#  yum install python-keystoneclient
   [root@controller etc]#  yum install python-neutronclient
   [root@controller etc]#  yum install python-novaclient
   [root@controller etc]#  yum install python-swiftclient
   [root@controller keystone]#  yum install python-troveclient
   ```

4. Using a text editor, create and source an OpenStack RC file named admin-openrc.sh. For example:

   ```
   export OS_USERNAME=admin
   export OS_PASSWORD=insert your password
   export OS_TENANT_NAME=admin
   export OS_AUTH_URL=http://controller:35357/v2.0
   ```

   Set environment variables using the source admin-openrc.sh command:

   ```
   [root@controller etc]#  source admin-openrc.sh
   ```

Override environment variable values

When running OpenStack client commands, you can override some environment variable settings by using the options that are listed at the end of the help output of the various client commands. For example, you can override the OS_PASSWORD setting in the PROJECT-openrc.sh file by specifying a password on a Keystone command, as follows:

```
[root@controller etc]#  keystone --os-password PASSWORD service-list
```

Image service overview

The Image service includes the following components:

- Glance-api: accepts Image API calls for image discovery, retrieval, and storage.
- Glance-registry: stores, processes, and retrieves image, metadata, such as size and type.

  Security note – It is recommended to not expose the registry to users. It is a private internal service meant only for the Image service itself.

- Database: stores image metadata. MySQL was used for this RHEL 6.5 deployment.
- Storage repository for image files: the Image service supports a variety of repositories including normal file systems, Object Storage, RADOS block devices, HTTP, and Amazon S3. Some types of repositories support only read-only usage.

Install the Image service

1. Install the Image service on the controller node:

   [root@controller]# **yum install openstack-glance python-glanceclient**

2. Configure the location of the database:

   [root@controller glance]# **openstack-config --set /etc/glance/glance-api.conf database **
   connection mysql://glance:Password@controller/glance
   [root@controller glance]# **openstack-config --set /etc/glance/glance-registry.conf database **
   connection mysql://glance:Password@controller/glance

3. Create a glance database user:

   [root@controller glance]# **mysql -u root –p**

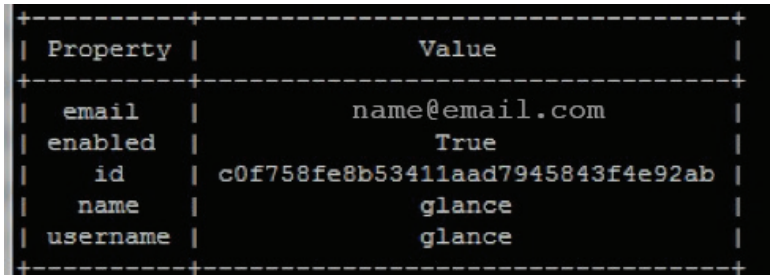
   Enter password: **password**
   mysql> **CREATE DATABASE glance;**
   mysql> **GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' **
   IDENTIFIED BY 'Password';
   mysql> **GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' **
   IDENTIFIED BY 'Password';
   mysql> **exit**

4. Create the database tables for the Image service:

   [root@controller glance]# **su -s /bin/sh -c "glance-manage db_sync" glance**

5. Create a glance user that the Image service can use to authenticate with the Identity service. Choose a password and specify an email address for the glance user. Use the service tenant and give the user the admin role:

   [root@controller gmp-6.0.0]# **keystone user-create --name=glance --pass=insert your password **
   --email=name@company.com



Property	Value
email	name@email.com
enabled	True
id	c0f758fe8b53411aad7945843f4e92ab
name	glance
username	glance

6. Configure the Image service to use the Identity service for authentication:

```
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              auth_uri http://controller:5000
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              > auth_uri http://controller:5000
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              auth_host controller
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              auth_port 35357
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              auth_protocol http
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              admin_user glance
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf keystone_authtoken \
                              admin_password insert your password
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-api.conf paste_deploy \
                              flavor keystone
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              auth_uri http://controller:5000
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              auth_host controller
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              auth_port 35357
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              auth_protocol http
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              admin_tenant_name service
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              admin_user glance
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf keystone_authtoken \
                              admin_password insert your password
[root@controller gmp-6.0.0]#  openstack-config --set /etc/glance/glance-registry.conf paste_deploy \
                              flavor keystone
```

7. Register the Image service with the Identity service so that other OpenStack services can locate it. Register the service and create the endpoint:

```
[root@controller gmp-6.0.0]#  keystone endpoint-create \
                              --service-id=$(keystone service-list | awk '/ image / {print $2}') \
                              --publicurl=http://controller:9292\
                              --internalurl=http://controller:9292 \
                              --adminurl=http://controller:9292
```

8. Start the glance-api service and glance-registry services and configure to automatically start the system boots:

```
[root@controller glance]#  service openstack-glance-api start
[root@controller glance]#  chkconfig openstack-glance-api  on
[root@controller glance]#  service openstack-glance-registry start
[root@controller glance]#  chkconfig openstack-glance-registry  on
```

Verify the Image service installation

To test the Image service installation, download at least one VM image that is known to work with OpenStack. For example, CirrOS is a small test image that is often used for testing OpenStack deployments. For this deployment, the 64-bit CirrOS QCOW2 image was used.

1. Download the image into a dedicated directory (i.e., /usr/tmp) using the wget command.

   [root@controller images]# **wget http://cdn.download.cirros---cloud.net/0.3.2/cirros---0.3.2---x86_64---disk.img**

2. Upload the image to the Image service. In the example below, the qcow2 disk format is used.

   [root@controller] glance]# **glance image-create --name=cirros032 --disk-format qcow2 --container-format=bare --is-public=True < /tmp/images/cirros-0.3.2-x86_64-disk.img**

```
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 64d7c1cd2b6f60c92c14662941cb7913     |
| container_format | bare                                 |
| created_at       | 2014-10-08T08:41:22                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | f19f6af9-1b16-4304-b7d7-755d4e450056 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | cirros032                            |
| owner            | 0ab81199b0e34f65a2ca395c1f33c2fd     |
| protected        | False                                |
| size             | 13167616                             |
| status           | active                               |
| updated_at       | 2014-10-08T08:41:22                  |
| virtual_size     | None                                 |
+------------------+--------------------------------------+
```

3. Verify the format using the file command:

   [root@controller] glance]# **file cirros-0.3.2-x86_64-disk.img**

4. Confirm that the image was uploaded and display its attributes:

   [root@controller glance]# **glance image-list**

```
+--------------------------------------+-----------+-------------+------------------+----------+--------+
| ID                                   | Name      | Disk Format | Container Format | Size     | Status |
+--------------------------------------+-----------+-------------+------------------+----------+--------+
| f19f6af9-1b16-4304-b7d7-755d4e450056 | cirros032 | qcow2       | bare             | 13167616 | active |
+--------------------------------------+-----------+-------------+------------------+----------+--------+
```

5. You can remove the downloaded image from /usr/tmp directory because it is now stored and available through the Image service.

Configure Compute services

The Compute service is a cloud computing fabric controller, which is the main part of an IaaS system. Use it to host and manage cloud computing systems. The main modules are implemented in Python.Compute interacts with the Identity service for authentication, the Image service for images, and the Dashboard service for the user and administrative interface. Access to images is limited by project and by user; quotas are limited per project (for example, the number of instances).The Compute service scales horizontally on standard hardware, and downloads images to launch instances as required. For more information on Compute and nova API services, reference the OpenStack Installation Guide at www.openstack.org.

The Compute service components include:

- Compute core:
  - nova-compute process
  - nova-scheduler process
  - nova-conductor module
- Networking for vms:
  - nova-network worker daemon
  - nova-dhcpbridge script
- Console interface:
  - nova-consoleauth daemon
  - nova-novncproxy daemon
  - nova-xvpnvncoroxy daemon
  - nova-cert daemon
- Image management (EC2 scenario):
  - nova-objectstore daemon
- Command-line clients and other interfaces:
  - nova-client
  - nova-manage client
- Additional components:
  - The queue – RabbitMQ, Qpid, ZeroMQ
  - SQL database – MySQL etc.

Install Compute controller services

Compute is a collection of services that enable you to launch VM instances. These services can be configured to run on separate nodes or on the same node. In this guide, most services run on the controller node and the service that launches each VM runs on a dedicated compute node. This section illustrates how to install and configure these services on the controller node.

1. Install the Compute packages necessary for the controller node:

```
[root@controller etc]# yum install openstack-nova-api openstack-nova-cert openstack-nova-conductor \
openstack-nova-console openstack-nova-novncproxy openstack-nova-scheduler \
python-novaclient
```

2. Configure Compute with the database location and credentials:

```
[root@controller nova]#  openstack-config --set /etc/nova/nova.conf \
                         database connection mysql://nova:Password@controller/nova
```

3. Configure Compute to use the qpid Message Broker:

```
[root@controller nova]#  openstack-config --set /etc/nova/nova.conf \
                         DEFAULT rpc_backend qpid
```

4. Set the my_ip, vncserver_listen, and vncserver_proxyclient_address configurations options to the management interface IP address of the controller node.

```
[root@controller nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT my_ip 10.193.35.1
[root@controller nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 10.193.35.1
[root@controller nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT
                         vncserver_proxyclient_address 10.193.35.1
```

5. Create a nova database user:

```
[root@controller nova]#  mysql -u root –p

Enter password:  password
mysql>  CREATE DATABASE nova;
mysql>  GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
        IDENTIFIED BY 'Password';
mysql>  GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
        IDENTIFIED BY 'Password';
mysql>  exit
```

6. Create the Compute service tables:

```
[root@controller nova]#  su -s /bin/sh -c "nova-manage db sync" nova
```

7. Create a nova user that Compute uses to authenticate with the Identify service. Use the service tenant and give the user the admin role:

```
[root@controller nova]#  keystone user-create --name=nova --pass=insert your password --
                         email=name@company.com
```

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         name@email.com           |
| enabled  |              True                |
|    id    | 0d2ebb1914d5436a90bb3af20150bd9a |
|   name   |              nova                |
| username |              nova                |
+----------+----------------------------------+
```

```
[root@controller nova]#  keystone user-role-add --user=nova --tenant=service --role=admin
```

8. Configure Compute to use these credentials with the Identity service running on the controller:

```
[root@controller nova]# openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy keystone
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_uri
http://controller:5000
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_host controller
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_protocol http
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port 35357
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user nova
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken
admin_tenant_name service
[root@controller nova]# openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_password
insert your password
```

9. Register Compute with the Identity service so that other OpenStack services can locate it. Register the service and specify the endpoint. Output response below shows both properties and values:

```
[root@controller nova]# keystone service-create --name=nova --type=compute \
--description="OpenStack Compute"
[root@controller nova]# keystone endpoint-create \
--service-id=$(keystone service-list | awk '/ compute / {print $2}') \
--publicurl=http://controller:8774/v2/%\(tenant_id\)s \
--internalurl=http://controller:8774/v2/%\(tenant_id\)s \
--adminurl=http://controller:8774/v2/%\(tenant_id\)s
```

```
+-------------+-----------------------------------------+
|   Property  |                Value                    |
+-------------+-----------------------------------------+
|   adminurl  | http://controller:8774/v2/%(tenant_id)s |
|      id     |     91074695f43f4b12ab3def9f7c2eee39    |
| internalurl | http://controller:8774/v2/%(tenant_id)s |
|  publicurl  | http://controller:8774/v2/%(tenant_id)s |
|    region   |                regionOne                |
|  service_id |     9d713475f692438eaea9dcc8fff28f25    |
+-------------+-----------------------------------------+
```

10. Start the Compute services and configure to start automatically with system boots:

```
[root@controller nova]# service openstack-nova-api start
[root@controller nova]# service openstack-nova-cert start
[root@controller nova]# service openstack-nova-consoleauth start
[root@controller nova]# service openstack-nova-scheduler start
[root@controller nova]# service openstack-nova-conductor start
[root@controller nova]# service openstack-nova-novncproxy start
```

11. Verify that the Compute services started:

    [root@controller nova]# **[root@controller nova]# nova-manage service list**

```
[root@controller nova]# nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-consoleauth controller.lab.sj.emulex.com         internal         enabled    :-)   2014-10-19 18:12:51
nova-scheduler   controller.lab.sj.emulex.com         internal         enabled    :-)   2014-10-19 18:12:54
nova-conductor   controller.lab.sj.emulex.com         internal         enabled    :-)   2014-10-19 18:13:00
nova-compute     compute1.lab.sj.emulex.com           nova             enabled    :-)   2014-10-19 18:12:59
nova-cert        controller.lab.sj.emulex.com         internal         enabled    :-)   2014-10-19 18:12:58
```

    Note – A state of :-) indicates Compute services are registered and working properly.

12. Verify the configuration by listing available images:

    [root@controller nova]# **nova image-list**

```
+--------------------------------------+-----------+--------+--------+
| ID                                   | Name      | Status | Server |
+--------------------------------------+-----------+--------+--------+
| 86774f17-3568-4b25-a025-3e3dfb58754a | cirros032 | ACTIVE |        |
+--------------------------------------+-----------+--------+--------+
```

Configure compute node

The Compute service relies on a hypervisor to run VM instances. OpenStack can use various hypervisors, but this guide uses QEMU. The compute node receives requests from the controller node and hosts VM instances. After you configure the Compute service on the controller node, you must configure another system as a compute node. The compute node receives requests from the controller node and hosts VM instances. This makes it easy to scale horizontally by adding additional compute nodes.

1. Install the Compute packages on the compute node. In this example, the compute node is named Compute1;

   [root@compute1 etc]# **yum install openstack-nova-compute**

2. Edit the /etc/nova/nova.conf configuration file;

   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf DEFAULT auth_strategy \ Keystone**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_uri \ http://controller:5000**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken \ auth_host controller**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken \ auth_protocol http**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken auth_port \ 35357**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken admin_user \ nova**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken \ admin_tenant_name service**
   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf keystone_authtoken \ admin_password insert your password**

3. Configure the Compute service to use the Qpid message broker by setting the configuration keys:

   [root@compute1 nova]# **openstack-config --set /etc/nova/nova.conf \DEFAULT rpc_backend qpid**
   [root@compute1 nova]# **openstack-config –set /etc/nova/nova.conf DEFAULT qpid_hostname controller**

4. Configure Compute to provide remote console access to instances:

```
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT my_ip 10.193.35.3
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT vnc_enabled True
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT vncserver_listen 0.0.0.0
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT
                       vncserver_proxyclient_address 10.193.35.3
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf \
                       DEFAULT novncproxy_base_url http://controller:6080/vnc_auto.html
```

5. Specify the host that runs the Image service:

```
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf DEFAULT glance_host controller
```

6. You must determine whether your system's processor and/or hypervisor support hardware acceleration for virtual machines. Run the following command:

```
[root@compute1 nova]#  egrep -c '(vmx|svm)' /proc/cpuinfo "16"
```

Note – If this command returns a value of one or greater, your system supports hardware acceleration which typically requires no additional configuration. However, if this command returns a value of zero, your system does not support hardware acceleration and you must configure libvirt to use QEMU instead of KVM.

If your system is capable of QEMU or KVM, run the following command and enter virt_type:

```
[root@compute1 nova]#  openstack-config --set /etc/nova/nova.conf libvirt virt_type kvm
```

7. Start the Compute service and its dependencies. Configure them to start automatically when the system boots:

```
[root@compute1 nova]#  service libvirtd start
[root@compute1 nova]#  service messagebus start
[root@compute1 nova]#  service openstack-nova-compute start
[root@compute1 nova]#  chkconfig libvirtd on
[root@compute1 nova]#  chkconfig messagebus on
[root@compute1 nova]#  chkconfig openstack-nova-compute on
```

Add Networking service on controller node

Configure OpenStack Networking (Neutron)

OpenStack Networking (Neutron) manages all of the networking facets for the Virtual Networking Infrastructure (VNI) and the access layer aspects of the Physical Networking Infrastructure (PNI) in your OpenStack environment. OpenStack Networking allows tenants to create advanced virtual network topologies including services such as firewalls, load balancers, and virtual private networks (VPNs).

Networking provides the following object abstractions: networks, subnets and routers. Each has functionality that mimics its physical counterpart: networks contain subnets, and routers route traffic between different subnet and networks. For more details on the neutron networking stack, please refer to the OpenStack Installation Guide for Red Hat Enterprise Linux, CentOS, and Fedora (September 19, 2014).

Configure Modular Layer 2 (ML2) plug-in on controller node

As a prerequisite, create Database and Identity service credentials including a user and service. This step is done on the controller node.

1. Create a Neutron database, and grant the proper access to it:

```
[root@controller etc]#  mysql -u root –p

          Enter password:  password
                   mysql>  CREATE DATABASE neutron;
                   mysql>  GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
                           IDENTIFIED BY 'Password';
                   mysql>  GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'%' \
                           IDENTIFIED BY 'Password';
                   mysql>  exit
```

2. Create Identity service credentials for Networking:

   a. Create the Neutron user:

   [root@controller etc]# keystone user-create --name neutron --pass insert a password --email name@company.com

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |          name@email.com          |
| enabled  |               True               |
|    id    | 8c257b1a948f4c59ac478b56d95e749e |
|   name   |             neutron              |
| username |             neutron              |
+----------+----------------------------------+
```

   b. Link the neutron user to the service tenant and admin role:

   [root@controller etc]# keystone user-role-add --user neutron --tenant service --role admin

c. Create the Networking service:

   [root@controller etc]# keystone service-create --name neutron --type network --description "OpenStack Networking"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       OpenStack Networking       |
|   enabled   |               True               |
|      id     | 32a4a9592b2a4d18a94ecacb547c1e82 |
|     name    |             neutron              |
|     type    |             network              |
+-------------+----------------------------------+
```

d. Create the service endpoint:

   [root@controller etc]# keystone endpoint-create \
   --service-id $(keystone service-list | awk '/ network / {print $2}') \
   --publicurl http://controller:9696 \
   --adminurl http://controller:9696 \
   --internalurl http://controller:9696 \

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |      http://controller:9696      |
|      id     | 2ca704f03aff41c39bec910aa578d440 |
| internalurl |      http://controller:9696      |
|  publicurl  |      http://controller:9696      |
|    region   |            regionOne             |
|  service_id | 32a4a9592b2a4d18a94ecacb547c1e82 |
+-------------+----------------------------------+
```

Install the Networking components

1. Install the Networking components on the controller node:

```
[root@controller etc]#  yum install openstack-neutron openstack-neutron-ml2 python-neutronclient
```

2. Configure Networking to use MySQL Database:

```
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf database connection \
                        mysql://neutron:Password@controller/neutron
```

3. Configure Networking to use the Identity service for authentication:

```
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        auth_strategy keystone
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        auth_uri http://controller:5000
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        auth_host controller
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        auth_protocol http
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        auth_port 35357
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        admin_tenant_name service
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        admin_user neutron
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                        admin_password insert your password
```

4. Configure Networking to use the Message Broker:

```
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        rpc_backend neutron.openstack.common.rpc.impl_qpid
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        qpid_hostname controller
```

5. Configure Networking to notify Compute about network topology changes:

```
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        notify_nova_on_port_status_changes True
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        notify_nova_on_port_data_changes True
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        nova_url http://controller:8774/v2
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        nova_admin_username nova
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        nova_admin_tenant_id $(keystone tenant-list | awk '/ service / { print $2 }')
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        nova_admin_password insert your password
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        nova_admin_auth_url http://controller:35357/v2.0
```

6. Configure Networking to use the Modular Layer 2 (ML2) plug-in and associated services:

```
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        core_plugin ml2
[root@controller etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                        service_plugins router
```

Configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the Open vSwitch (OVS) mechanism (agent) to build the virtual networking framework for instances. However, the controller node does not need the OVS agent or service because it does not handle instance network traffic.

```
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
type_drivers gre
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
tunnel_id_ranges 1:1000
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[root@controller etc]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
enable_security_group True
```

Configure Compute to use Networking

Most distributions configure Compute to use legacy networking. You must reconfigure Compute to manage networks through Networking.

1. Run the following commands to configure Compute to use networking:

```
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
network_api_class nova.network.neutronv2.api.API
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_url http://controller:9696
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_auth_strategy keystone
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_tenant_name service
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_username neutron
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_password insert your password
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_auth_url http://controller:35357/v2.0
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
firewall_driver nova.virt.firewall.NoopFirewallDriver
[root@controller neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \ security_group_api neutron
```

Note – By default, Compute uses an internal firewall service. Since Networking includes a firewall service, you must disable the Compute firewall service by using the nova.virt.firewall.NoopFirewallDriver firewall driver.

Finalize the Networking service installation

The Networking service initialization scripts expect a symbolic link /etc/neutron/ plugin.ini pointing to the configuration file associated with your chosen plug-in. Using ML2, for example, the symbolic link must point to /etc/neutron/plugins/ml2/ml2_conf.ini:

1. If the symbolic link doesn't exist, you need to create it.

   [root@controller neutron]# **ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini**

2. Restart the Compute services:

   [root@controller neutron]# **service openstack-nova-api restart**
   [root@controller neutron]# **service openstack-nova-scheduler restart**
   [root@controller neutron]# **service openstack-nova-conductor restart**

```
[root@controller neutron]# service openstack-nova-api restart
Stopping openstack-nova-api:                               [  OK  ]
Starting openstack-nova-api:                               [  OK  ]
[root@controller neutron]# service openstack-nova-scheduler restart
Stopping openstack-nova-scheduler:                         [  OK  ]
Starting openstack-nova-scheduler:                         [  OK  ]
[root@controller neutron]# service openstack-nova-conductor restart
Stopping openstack-nova-conductor:                         [  OK  ]
Starting openstack-nova-conductor:                         [  OK  ]
[root@controller neutron]#
```

3. Start the Networking service and configure it to start when the system boots:

   [root@controller neutron]# **service neutron-server start**

```
[root@controller neutron]# service neutron-server start
Starting neutron:                                          [  OK  ]
```

   [root@controller neutron]# **chkconfig neutron-server on**

   Note – Unlike other services, Networking typically does not require a separate step to populate the database because the neutron-server service populates it automatically. However, the packages for these distributions sometimes require running the neutron-db-manage command listed below prior to starting the neutron-server service. It's highly recommended to attempt to start the service before manually populating the database.

   If the service starts without error, steps 1-3 are not necessary. However, if the neutron-service returns database errors, perform the following operations:

1. Configure Networking service to use long plug-in names:

   # openstack-config --set /etc/neutron/neutron.conf DEFAULT \
   core_plugin neutron.plugins.ml2.plugin.Ml2Plugin

   # openstack-config --set /etc/neutron/neutron.conf DEFAULT \
   service_plugins neutron.services.l3_router. l3_router_plugin.L3RouterPlugin

2. Populate the database:

   # su -s /bin/sh -c "neutron-db-manage --config-file /etc/ neutron/neutron.conf \
   --config-file /etc/neutron/plugin.ini upgrade head" neutron

3. Attempt to start the Neutron-server service again. You can return the core_plugin and service_plugins configuration keys to short plug-in names.

Configure the network node

1. Edit /etc/sysctl.conf to contain the following:

```
[root@network etc]#  vi /etc/systl.conf

net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter=0
net.ipv4.conf.default.rp_filter=0
```

2. Implement the changes:

```
[root@network etc]#  sysctl –p
```

Install and configure Networking common components

1. Use yum to install Networking components on network node.

```
[root@network etc]#  yum install openstack-neutron openstack-neutron-ml2 openstack-neutron- openvswitch \
                     openstack-neutron-openvswitch
```

2. Configure Networking service to use the Identity service for authentication:

```
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                     auth_strategy keystone
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     auth_uri http://controller:5000
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     auth_host controller
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     auth_protocol http
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     auth_port 35357
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     admin_tenant_name service
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     admin_user neutron
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                     admin_password insert your password
```

3. Configure Networking service to use the Message Broker:

```
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                     rpc_backend neutron.openstack.common.rpc.impl_qpid
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                     qpid_hostname controller
```

4. Configure Networking to use the Modular Layer 2 (ML2) plug-in and associated services:

```
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                     core_plugin ml2
[root@network etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                     Service_plugins router
```

Note – It's recommended to add verbose = True to the [DEFAULT] section in /etc/neutron/neutron.conf to assist with troubleshooting.

Configure the Layer-3 (L3) agent

The L3 agent provides routing services for instance virtual networks.

1. Configure the Layer 3 (L3) agent that provides routing services for instance virtual networks.

```
[root@network etc]#  openstack-config --set /etc/neutron/l3_agent.ini DEFAULT \
                     interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
[root@network etc]#  openstack-config --set /etc/neutron/l3_agent.ini DEFAULT \
                     use_namespaces True
```

Note – It's recommended to add verbose = True to the [DEFAULT] section in /etc/neutron/l3_agent.ini to assist with troubleshooting.

Configure the DHCP agent

The DHCP agent provides DHCP services for instance virtual networks.

1. Run the following commands to configure DHCP services:

```
[root@network etc]#  openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT \
                     interface_driver neutron.agent.linux.interface.OVSInterfaceDriver
[root@network etc]#  openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT \
                     dhcp_driver neutron.agent.linux.dhcp.Dnsmasq
[root@network etc]#  openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT \
                     use_namespaces True
```

Note – It's recommended to add verbose = True to the [DEFAULT] section in /etc/neutron/dhcp_agent.ini to assist with troubleshooting.

2. Because tunneling protocols such as generic routing encapsulation (GRE) include additional packet headers that increase overhead and decrease available space for the payload or user data, OpenStack recommends experimentation to determine the proper maximum transition unit (MTU) required for your environment. Please refer to the OpenStack Configuration Reference Guide (Icehouse) for more information related to configuring MTU settings.

   a. Run the following command:

```
[root@network etc]#  openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT \
                     dnsmasq_config_file /etc/neutron/dnsmasq-neutron.conf
```

   b. Create and edit the /etc/neutron/dnsmasq-neutron.conf file and add the following keys:

```
[root@network neutron]#  vi /etc/neutron/dnsmasq-neutron.conf dhcp-option-force=26,1454
```

   c. Kill any existing dnsmasq processes:

```
[root@network neutron]#  killall dnsmasq
```

Configure the metadata agent

The metadata agent provides configuration information such as credentials for remote access to instances.

1. Run the following commands to configure the metadata agent:

```
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
auth_url http://controller:5000/v2.0
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
auth_region regionOne
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
admin_tenant_name service
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
admin_user neutron
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
admin_password insert your password
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
nova_metadata_ip controller
[root@network neutron]# openstack-config --set /etc/neutron/metadata_agent.ini DEFAULT \
metadata_proxy_shared_secret Password
```

Note – Perform the next two steps on the controller node.

2. On the controller node, configure Compute service to use metadata service. Replace METADATA_SECRET with the secret you chose for the metadata proxy:

```
[root@controller]# openstack-config --set /etc/nova/nova.conf DEFAULT \
service_neutron_metadata_proxy true
[root@controller]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_metadata_proxy_shared_secret Password
```

3. On the controller node, restart the Compute API service:

```
[root@controller]# service openstack-nova-api restart
```

```
[root@controller ~]# service openstack-nova-api restart
Stopping openstack-nova-api:                               [  OK  ]
Starting openstack-nova-api:                               [  OK  ]
```

Configure the Modular Layer 2 (ML2) plug-in on the network node

The ML2 plug-in uses the OVS mechanism (agent) to build virtual networking framework for instances. Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your network node. For this setup we use IP address 10.4.5.1 for the instance tunnels network interface on the network node:

```
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
type_drivers gre
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
tunnel_id_ranges 1:1000
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
local_ip 10.4.5.1
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
tunnel_type gre
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
enable_tunneling True
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[root@network ml2]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
enable_security_group True
```

Configure the Open VSwitch (OVS) service

The OVS service provides the underlying virtual networking framework for instances. The integration bridge br-int handles internal instance network traffic within OVS. The external bridge br-ex handles external instance network traffic within OVS. The external bridge requires a port on the physical external network interface to provide instances with external network access.

1. Start the OVS service and configure to start when the system boots:

```
[root@network ml2]# service openvswitch start
[root@network ml2]# chkconfig openvswitch on
```

```
[root@network ml2]# service openvswitch start
/etc/openvswitch/conf.db does not exist ... (warning).
Creating empty database /etc/openvswitch/conf.db           [  OK  ]
Starting ovsdb-server                                      [  OK  ]
Configuring Open vSwitch system IDs                        [  OK  ]
Inserting openvswitch module                               [  OK  ]
Starting ovs-vswitchd                                      [  OK  ]
Enabling remote OVSDB managers                             [  OK  ]
[root@network ml2]# chkconfig openvswitch on
```

2. Add the integration bridge:

```
[root@network openvswitch]# ovs-vsctl add-br br-int
```

3. Add the external bridge:

```
[root@network openvswitch]# ovs-vsctl add-br br-ex
```

4. Add a port to the external bridge that connects to the physical external network interface. Replace INTERFACE_NAME with the actual interface name. For the example in this document, we assigned NIC interface **p6p5** for instance tunnels.

```
[root@network openvswitch]#  ovs-vsctl add-port br-ex p6p5
```

Note – The OpenStack guide recommends disabling Generic Receive Offload (GRO) to achieve suitable throughput between instances and the external network. By default, the GRO is enabled on the Emulex OneConnect OCe14100 adapters. Using ethtool –k command, we see the following settings for OCe14100:

```
[root@network ml2]#  ethtool -k p6p3

rx-checksumming: on
tx-checksumming: on
scatter-gather: on
tcp-segmentation-offload: on
udp-fragmentation-offload: off
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
ntuple-filters: off
receive-hashing: on
```

To disable GRO, run the following command:

```
[root@network ml2]#  ethtool -K p6p3 gro off
```

We recommend that you experiment with GRO in your environment to determine the best option.

Finalize the network node installation

1. The Networking service initialization scripts expect a symbolic link /etc/neutron/plugin.ini pointing to the configuration file associated with your chosen plug-in. Using the ML2 plug-in, for example, the symbolic link must point to /etc/neutron/plugins/ml2/ml2_conf.ini. If this symbolic link does not exist, create it using the following commands:

```
[root@network neutron]#  ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
[root@network neutron]#  cp /etc/init.d/neutron-openvswitch-agent /etc/init.d/neutron-openvswitch- agent.orig
[root@network neutron]#  sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g'
                         /etc/init.d/neutron-openvswitch-agent
```

2. Start the Networking services and configure to start when the system boots:

```
[root@network neutron]#  service neutron-openvswitch-agent start
[root@network neutron]#  service neutron-l3-agent start
[root@network neutron]#  service neutron-dhcp-agent start
[root@network neutron]#  service neutron-metadata-agent start
[root@network neutron]#  chkconfig neutron-openvswitch-agent on
[root@network neutron]#  chkconfig neutron-l3-agent on
[root@network neutron]#  chkconfig neutron-dhcp-agent on
[root@network neutron]#  chkconfig neutron-metadata-agent on
```

```
[root@network neutron]# service neutron-openvswitch-agent start
Starting neutron-openvswitch-agent:                        [  OK  ]
[root@network neutron]# service neutron-l3-agent start
Starting neutron-l3-agent:                                 [  OK  ]
[root@network neutron]# service neutron-dhcp-agent start
Starting neutron-dhcp-agent:                               [  OK  ]
[root@network neutron]# service neutron-metadata-agent start
Starting neutron-metadata-agent:                           [  OK  ]
[root@network neutron]# chkconfig neutron-openvswitch-agent on
[root@network neutron]# chkconfig neutron-l3-agent on
[root@network neutron]# chkconfig neutron-dhcp-agent on
[root@network neutron]# chkconfig neutron-metadata-agent on
```

Configure compute node for Networking

1. Enable Kernel networking functions. Edit /etc/sysctl.conf to contain the following:

   net.ipv4.conf.default.rp_filter = 0 net.ipv4.conf.default.rp_filter=0

2. Implement changes:

   ```
   [root@compute1 etc]#  sysctl -p
   ```

3. Install the Networking components:

   ```
   [root@compute1 etc]#  yum install openstack-neutron-ml2 openstack-neutron-openvswitch
   ```

4. Configure Networking to use the Identity service for authentication:

   ```
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                         auth_strategy keystone
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         auth_uri http://controller:5000
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         auth_host controller
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         auth_protocol http
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         auth_port 35357
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         admin_tenant_name service
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         admin_user neutron
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf keystone_authtoken \
                         admin_password insert your password
   ```

5. Configure Networking to use Message Broker:

   ```
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                         rpc_backend neutron.openstack.common.rpc.impl_qpid
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                         qpid_hostname controller
   ```

6. Configure Networking to use the ML2 plug-in and associated services:

   ```
   [root@compute1 etc]#  openstack-config --set /etc/neutron/neutron.conf DEFAULT \
                         core_plugin ml2
   ```

Configure the Modular Layer 2 (ML2) plug-in

The ML2 plug-in uses the OVS mechanism (agent) to build the virtual networking framework for instances. Replace INSTANCE_TUNNELS_INTERFACE_IP_ADDRESS with the IP address of the instance tunnels network interface on your compute node. This guide uses 10.4.5.2 for the IP address of the instance tunnels network interface (eth3) on the compute node (compute1).

Run the following commands:

```
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
type_drivers gre
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
tenant_network_types gre
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2 \
mechanism_drivers openvswitch
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ml2_type_gre \
tunnel_id_ranges 1:1000
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
local_ip 10.4.5.2
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
tunnel_type gre
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini ovs \
enable_tunneling True
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
firewall_driver neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver
[root@compute1 neutron]# openstack-config --set /etc/neutron/plugins/ml2/ml2_conf.ini securitygroup \
enable_security_group True
```

Configure the OVS service

1. Start the OVS service and configure it to start when the system boots:

```
[root@compute1 neutron]# service openvswitch start
[root@compute1 neutron]# chkconfig openvswitch on
```

2. Add the integration bridge:

```
[root@compute1 neutron]# ovs-vsctl add-br br-int
```

Configure compute node to use Networking

Reconfigure Compute to manage networks through Networking. Run the following commands:

```
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
network_api_class nova.network.neutronv2.api.API
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_url http://controller:9696
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_auth_strategy keystone
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_tenant_name service
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_username neutron
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_password insert your password
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
neutron_admin_auth_url http://controller:35357/v2.0
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
linuxnet_interface_driver nova.network.linux_net.LinuxOVSInterfaceDriver
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
firewall_driver nova.virt.firewall.NoopFirewallDriver
[root@compute1 neutron]# openstack-config --set /etc/nova/nova.conf DEFAULT \
security_group_api neutron
```

Finalize the installation

1. Configure symbolic links to point to /etc/neutron/plugins/ml2/ml2_conf.ini file:

```
[root@compute1 neutron]# ln -s plugins/ml2/ml2_conf.ini /etc/neutron/plugin.ini
```

2. Run the following commands to resolve possible issues with packaging:

```
[root@compute1 neutron]# cp /etc/init.d/neutron-openvswitch-agent /etc/init.d/neutron-openvswitch- agent.orig
[root@compute1 neutron]# sed -i 's,plugins/openvswitch/ovs_neutron_plugin.ini,plugin.ini,g'
/etc/init.d/neutron-openvswitch-agent
```

3. Restart the Compute service:

```
[root@compute1 init.d]# service openstack-nova-compute restart
```

4. Start the OVS agent and configure it to start when the system boots:

```
[root@compute1 init.d]# service neutron-openvswitch-agent start
[root@compute1 init.d]# chkconfig neutron-openvswitch-agent on
```

Create initial networks

To create the external network:

1. Source the admin tenant credentials:

   [root@controller keystone]# **source admin-openrc.sh**

2. Create the network:

   [root@controller]# **neutron net-create ext-net --shared --router:external=True**

```
+---------------------------+--------------------------------------+
|           Field           |                Value                 |
+---------------------------+--------------------------------------+
|       admin_state_up      |                True                  |
|            id             | 246b40ec-fada-4d66-b1a1-a06b5014e81d |
|           name            |               ext-net                |
|   provider:network_type   |                 gre                  |
| provider:physical_network |                                      |
| provider:segmentation_id  |                  1                   |
|      router:external      |                True                  |
|          shared           |                True                  |
|          status           |               ACTIVE                 |
|          subnets          |                                      |
|         tenant_id         |   adf92ed8340f4a488f229b629cc39a86   |
+---------------------------+--------------------------------------+
```

3. Create a subnet on the external network:

   First, specify an exclusive slice of this subnet for router and floating IP addresses to prevent interference with other devices on the external network:

   Floating IP range: 10.193.36.220 – 10.193.36.250
   Gateway: 10.193.36.200 External CIDR: 10.193.36.0/24
   [root@controller neutron]# **neutron subnet-create ext-net --name ext-subnet **
   **--allocation-pool start=10.193.36.220,end=10.193.36.250 **
   --disable-dhcp --gateway 10.193.36.200 10.193.36.0/24

   Created a new subnet:

```
+------------------+-------------------------------------------------------+
|       Field      |                         Value                         |
+------------------+-------------------------------------------------------+
| allocation_pools | {“start”: “10.193.36.220”, “end”: “10.193.36.250”} |
|        cidr      |                    10.193.36.0/24                     |
|  dns_nameservers |                                                       |
|    enable_dhcp   |                         False                         |
|    gateway_ip    |                     10.193.36.200                     |
|    host_routes   |                                                       |
|        id        |         2ff1c66f-9290-460a-9558-59d85134f144          |
|    ip_version    |                           4                           |
|       name       |                       ext-subnet                      |
|    network_id    |         246b40ec-fada-4d66-b1a1-a06b5014e81d          |
|     tenant_id    |           adf92ed8340f4a488f229b629cc39a86            |
+------------------+-------------------------------------------------------+
```

Create the tenant network

1. Source the demo tenant credentials:

   [root@controller]# **source admin-openrc.sh**

2. Create the tenant network:

   [root@controller]# **neutron net-create demo-net**

```
+---------------------------+--------------------------------------+
|           Field           |                Value                 |
+---------------------------+--------------------------------------+
|      admin_state_up       |                True                  |
|            id             | a78f8987-d4f8-4440-951c-f4b6d159bf62 |
|           name            |              demo-net                |
|   provider:network_type   |                 gre                  |
| provider:physical_network |                                      |
|  provider:segmentation_id |                  2                   |
|          shared           |                False                 |
|          status           |                ACTIVE                |
|          subnets          |                                      |
|         tenant_id         |   adf92ed8340f4a488f229b629cc39a86   |
+---------------------------+--------------------------------------+
```

3. Create a subnet on the tenant network:

   Tenant network Gateway: 192.168.1.1 Tenant network CIDR: 192.168.1.0/24

   [root@controller keystone]# **neutron subnet-create demo-net --name demo-subnet **
   --gateway 192.168.1.1 192.168.1.0/24

Created a new subnet:

```
+------------------+-------------------------------------------------+
|       Field      |                      Value                      |
+------------------+-------------------------------------------------+
| allocation_pools | {“start”: “192.168.1.2”, “end”: “192.168.1.254”} |
|       cidr       |                  192.168.1.0/24                 |
|  dns_nameservers |                                                 |
|    enable_dhcp   |                      True                       |
|    gateway_ip    |                   192.168.1.1                   |
|    host_routes   |                                                 |
|        id        |       1decbf16-bc23-4793-801f-981033de0d98      |
|    ip_version    |                        4                        |
|       name       |                   demo-subnet                   |
|    network_id    |       a78f8987-d4f8-4440-951c-f4b6d159bf62      |
|     tenant_id    |         adf92ed8340f4a488f229b629cc39a86        |
+------------------+-------------------------------------------------+
```

Create a router on the tenant network and attach the external and tenant networks

1. Create a router:

   [root@controller]# **neutron router-create demo-router**

   Created a new router:

```
+-----------------------+--------------------------------------+
|          Field        |                Value                 |
+-----------------------+--------------------------------------+
|    admin_state_up     |                True                  |
| external_gateway_info |                                      |
|           id          | 3b308fe5-9ad2-42a0-a90a-db708247c36e |
|          name         |               demo-router            |
|         status        |                 ACTIVE               |
|       tenant_id       |   adf92ed8340f4a488f229b629cc39a86   |
+-----------------------+--------------------------------------+
```

2. Attach the router to the demo tenant subnet:

   [root@controller]# **neutron router-interface-add demo-router demo-subnet**

3. Attach the router to the external network by setting it as the gateway:

   [root@controller]# **neutron router-gateway-set demo-router ext-net**

Verify connectivity

At this point, you should be able to test your external network using the ping command from an available host on your network. The IP addresses created in the allocation pool range start at 10.193.36.220 – 10.193.36.250. The tenant router gateway occupies the lowest IP in the floating IP range. You should be able to ping 10.193.36.220 from any host on the external network.

Add the OpenStack Dashboard (Horizon)

The OpenStack Dashboard (Horizon) is a Web interface that enables cloud administrators and users to manage various OpenStack resource and services. The instructions in this guide show an example deployment configured with an Apache web server. Please refer to the OpenStack Installation Guide (Icehouse) for requirements and further explanation.

1. Install the dashboard on the node that can contact the Identity service as root. In this example we selected the controller node:

   [root@controller]# **yum install memcached python-memcached mod_wsgi openstack-dashboard**

2. Modify the value of CACHES['default']['LOCATION'] in /etc/openstackdashboard/local_settings to match the ones set in /etc/sysconfig/memcached.

   Open /etc/openstack-dashboard/local_settings and look for this line:

```
CACHES = {
'default': {
'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION' : '127.0.0.1:11211'
}
}
```

   Note – You must uncomment the settings above.

3. Update the ALLOWED_HOSTS in local_settings.py to include the address from which you wish to access the dashboard.. Edit /etc/openstack-dashboard/local_settings:

   ALLOWED_HOSTS = ['localhost', 'my-desktop']

4. The Dashboard is run on the controller node in this deployment guide. However, it can also be run on a different platform by changing the appropriate settings in local_settings.py. Edit /etc/openstack-dashboard/local_settings and change OPENSTACK_HOST to the hostname of your Identity service:

   OPENSTACK_HOST = "controller"

5. Ensure SELinux policy of the system is configured to allow network connections to the HTTP server:

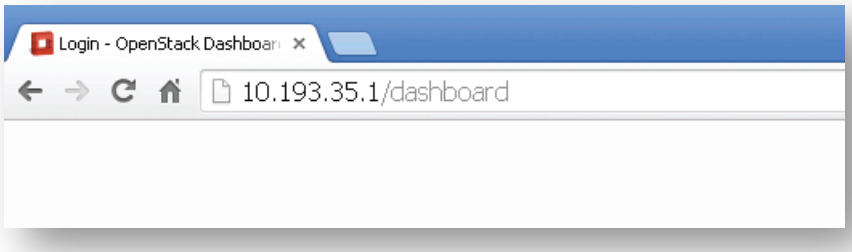
   ```
   [root@controller]#  setsebool -P httpd_can_network_connect on
   ```

6. Start the Apache web server and memcached:

   ```
   [root@controller]#  service httpd start
   [root@controller]#  service memcached start
   [root@controller]#  chkconfig httpd on
   [root@controller]#  chkconfig memcached on
   ```

Launch an instance using Horizon

Launch an instance from the Dashboard (Horizon) from a remote host in your network. From the web browser, enter the URL of the machine where the dashboard is installed. In this example, we installed the Dashboard on the controller node:



Enter a user name and password to log into the OpenStack Dashboard.

Figure 12 shows the OpenStack summary details of the OpenStack configuration that includes the number of instances, vCPUs used, RAM usage, floating IPs and security group information.

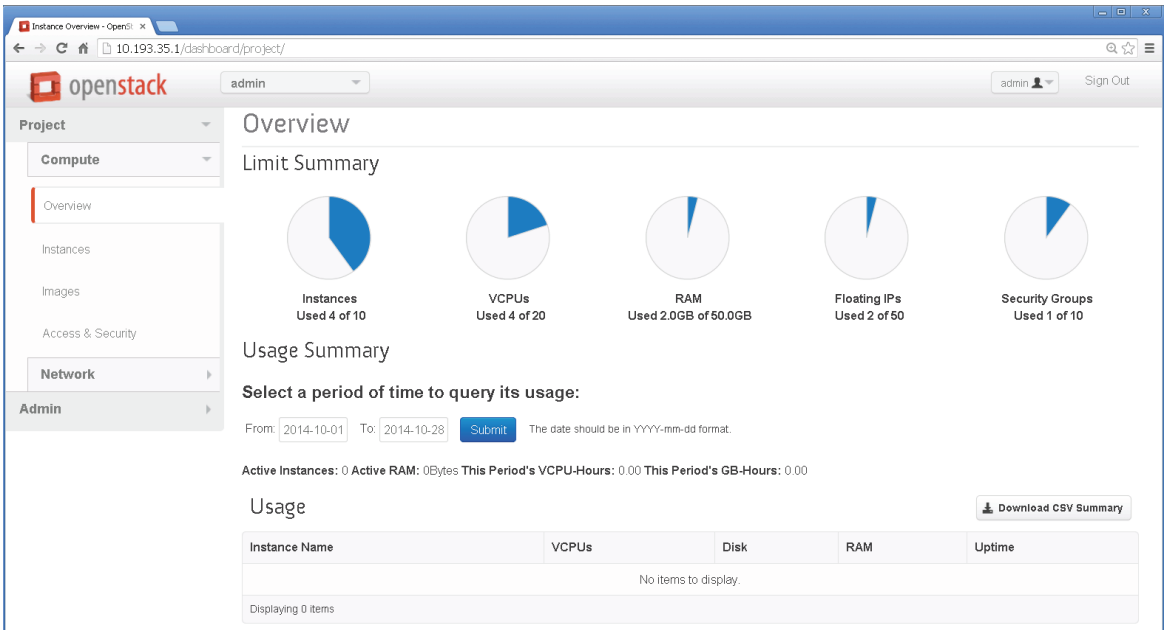


Figure 12. OpenStack Summary Overview.

Figure 13 illustrates four VMs connected to the demo-network 192.168.1.0/24, with access to the external network via internal gateway 192.168.1.1, and floating IP 10.193.36.220 created on the 10.193.36.0/24 network.

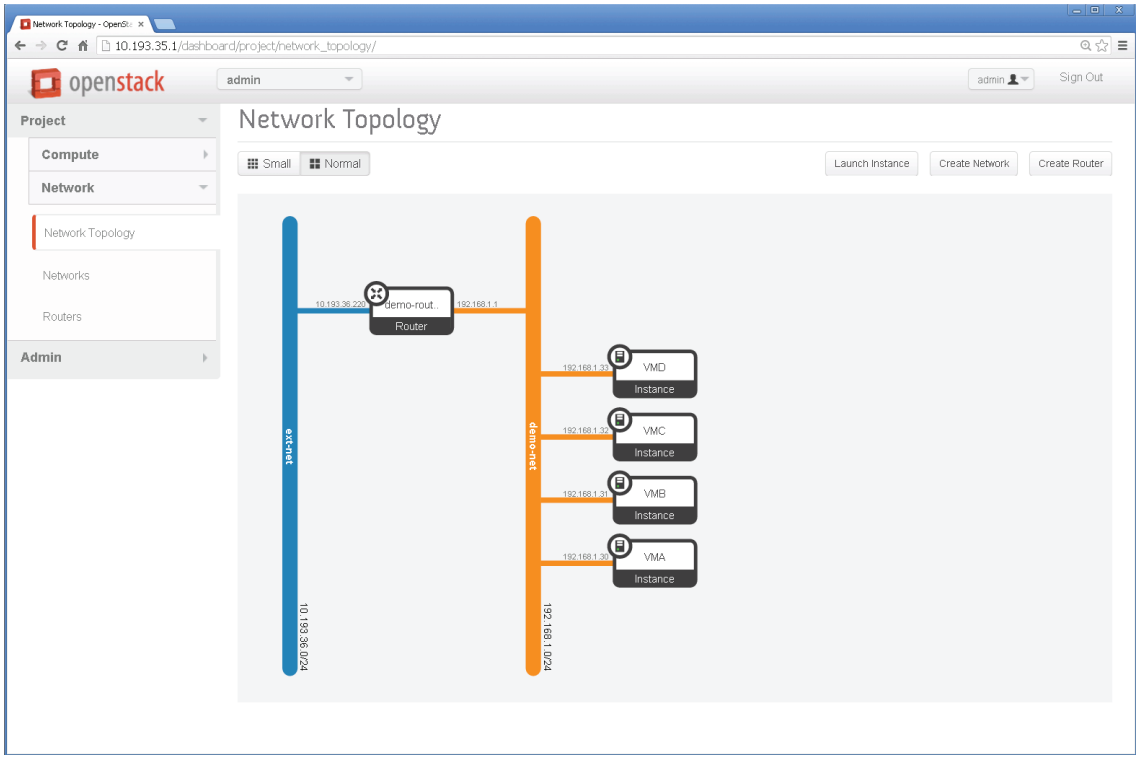


Figure 13. Network topology showing four VM instances.

Conclusion

This paper provides the administrator with steps to configure Emulex OneConnect OCe14000 adapters in a basic three-node OpenStack cloud configuration. Emulex UMC and OneCommand Manager technology provides the underlying networking essentials and tools for building cloud computing environments in high performance 10GbE and40GbE networks, while ensuring Quality of Service, secure and reliable connections. Data center administrators will see significant cost savings for cabling, adapters, switches and power. And finally, since UMC is switch-agnostic, it works with any existing 10GbE or 40GbE switching infrastructure.

References

OpenStack Installation Guide for Red Hat Enterprise Linux, Centos, and Fedora; September 19, 2014 (Icehouse)
http://docs.openstack.org/icehouse/install-guide/install/yum/openstack-install-guide-yum-icehouse.pdf

Red Hat Enterprise Linux OpenStack Platform 5
https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/

Emulex OneCommand Manager Enterprise Application (GUI)
http://www.emulex.com/downloads/emulex/drivers/linux/rhel-6-centos-6/management-and-utilities/

Emulex OneCommand Manager Core Application (CLI)
http://www.emulex.com/downloads/emulex/drivers/linux/rhel-6-centos-6/management-and-utilities/

EMULEX®

World Headquarters 3333 Susan Street, Costa Mesa, CA 92626 +1 714 662 5600
Bangalore, India +91 80 40156789 | **Beijing, China** +86 10 84400221
Dublin, Ireland +35 3 (0) 1 652 1700 | **Munich, Germany** +49 (0) 89 97007 177
Paris, France +33 (0) 158 580 022 | **Tokyo, Japan** +81 3 5325 3261 | **Singapore** +65 6866 3768
Wokingham, United Kingdom +44 (0) 118 977 2929 | **Brazil** +55 11 3443 7735

www.emulex.com