Oracle OpenStack for Oracle Linux Release 1.0 Installation and User's Guide

ORACLE WHITE PAPER | DECEMBER 2014

Introduction

This document guides you through setting up an OpenStack deployment with Oracle OpenStack for Oracle Linux. It demonstrates how to install the product with Oracle VM and Oracle Linux. OpenStack is a very flexible solution that can be configured, deployed, and used in various ways. This document provides guidelines to easily build a multi-node OpenStack deployment using Oracle VM and Oracle Linux.

The guide comprises two main parts. The first part guides you through the deployment and configuration of OpenStack with Xen (Oracle VM) and KVM (Oracle Linux). The second part shows how to test various features of OpenStack, including launching instances and using networking and storage features.

For simplicity, a small configuration was chosen that includes a control node and multiple compute nodes. While it is possible to configure OpenStack in various ways with the tools described here, the discussion is limited to a simplified configuration, allowing you to discover the possibilities of running Oracle VM and Oracle Linux with OpenStack.

Who Should Use this Guide?

This guide is written for readers who would like to get started with OpenStack. No prior knowledge of OpenStack or Oracle VM is required. Therefore, this guide can be used by those taking their first steps into the world of OpenStack. Using this guide, you can quickly put together an OpenStack deployment and test it to see whether Oracle VM or Oracle Linux fits your requirements.

The guide is also useful for vendors who are interested in integrating with Oracle VM or Oracle Linux, or who have customers interested in doing so. Vendors can use this guide to create a deployment on which integration with OpenStack and Oracle VM or Oracle Linux can be tested.

OpenStack Basics

This section gives an introduction to the components of OpenStack.

What Is OpenStack?

OpenStack is open source virtualization management software that allows users to connect various technologies and components from different vendors and expose a unified API, regardless of the underlying technology. With OpenStack, users can manage different types of hypervisors, network devices and services, storage components, and more using a single API that creates a unified data center fabric. OpenStack is, therefore, a pluggable framework that allows vendors to write plug-ins that implement a solution using their own technology, and which allows users to integrate their technology of choice.

OpenStack Services

To achieve this agility, OpenStack is built as a set of distributed services. These services communicate with each other and are responsible for the various functions expected from virtualization/cloud management software. The following are some of the key services of OpenStack:

- **Nova:** A compute service responsible for creating instances and managing their lifecycle, as well as managing the hypervisor of choice. The hypervisors are pluggable to Nova, while the Nova API remains the same, regardless of the underlying hypervisor.
- **Neutron:** A network service responsible for creating network connectivity and network services. It is capable of connecting with vendor network hardware through plug-ins. Neutron comes with a set of default services implemented by common tools. Network vendors can create plug-ins to replace any one of the services with their own implementation, adding value to their users.
- **Cinder:** A storage service responsible for creating and managing external storage, including block devices and NFS. It is capable of connecting to vendor storage hardware through plug-ins. Cinder has several generic plug-ins, which can connect to NFS and iSCSI, for example. Vendors add value by creating dedicated plug-ins for their storage devices.
- **Keystone:** An identity management system responsible for user and service authentication. Keystone is capable of integrating with third-party directory services and LDAP.
- **Glance:** An image service responsible for managing images uploaded by users. Glance is not a storage service, but it is responsible for saving image attributes, making a virtual catalog of the images.
- **Horizon:** A dashboard that creates a GUI for users to control the OpenStack deployment. This is an extensible framework that allows vendors to add features to it. Horizon uses the same APIs exposed to users.

More details are available in the OpenStack documentation at:

http://docs.openstack.org/admin-guide-cloud/content/ch_getting-started-with-openstack.html

OpenStack has many more services that are responsible for various features and capabilities, and the full list can be found on the OpenStack website at:

http://www.openstack.org/

The list presented here is limited to those needed to get started with Oracle VM and Oracle Linux.

OpenStack Instances

OpenStack virtual machines are called *instances*, mostly because they are instances of an image that is created upon request and that is configured when launched. The main difference between OpenStack and traditional virtualization technology is the way state is stored. With traditional virtualization technology, the state of the virtual machine is persistent.

OpenStack can support both persistent and ephemeral models. In the ephemeral model, an instance is launched from an image, the image is copied to the run area, and when the copy is completed, the instance starts running. The size and connectivity of the instance are defined at the time of launching the instance. This ephemeral model is useful for scaling out quickly and maintaining agility for users.

In the persistent model, the instance is launched from a volume. A volume can be any kind of persistent storage, including a file, a block device, an LVM partition, or any other form of persistent storage. In this case, when the instance is terminated, all the changes the user has made are kept and are present next time an instance is launched from the same volume. In the persistent model, the size and connectivity of the instance are also defined at the time the instance launches. In some sense, the persistent model in OpenStack is close to the traditional approach to virtualization.

OpenStack Storage

As already mentioned, the storage used in OpenStack can be either ephemeral or persistent. Ephemeral storage is deleted when an instance is terminated, while persistent storage remains intact. Persistent storage in OpenStack is referred to as a *volume*, regardless of the technology and device it is backed by. Persistent storage can either be used to launch an instance, or it can be connected to an instance as a secondary storage device to retain state. An example for this is a database launched as an ephemeral instance, with a volume connected to it to save the data. When the instance is terminated, the volume retains the data and can be connected to another instance as needed.

The OpenStack Cinder service is responsible for managing the volumes, and it offers a framework for vendors to create plug-ins. If a storage vendor wants to support OpenStack deployment and allow users to create volumes on the device, the vendor must create a Cinder plug-in that allows users to use the standard calls to control the storage device.

OpenStack also supports object storage using the Swift service, but that is not covered in this guide.

OpenStack Networking

This section gives an introduction to networking in OpenStack.

Network Services

The OpenStack networking service, Neutron, offers a complete software-defined networking (SDN) solution, along with various network services, right out of the box. The network services Neutron can support include routing, firewall, DNS, DHCP, load balance, VPN, and more.

Neutron, like Cinder, offers a framework for vendors to write plug-ins for various services. For example, a network vendor might want to offer a custom load balancer instead of the default load balancer provided by Neutron. This gives a user a powerful tool to build sophisticated network topologies using standard APIs.

Network Isolation: Tenant Networks

Tenant networks are the basis for Neutron’s SDN capability. Neutron has full control of layer-2 isolation. This automatic management of layer-2 isolation is completely hidden from the user, providing a convenient abstraction layer required by SDN.

To perform the layer-2 separation, Neutron supports three layer-2 isolation mechanisms: VLANs, VxLANs, and Generic Routing Encapsulation (GRE) tunnels. The user must define which mechanism should be used and set up the physical topology. Neutron is responsible for allocating the resources as needed. For example, a user should configure the VLAN switch, allocate the VLAN range, and configure the VLAN in Neutron. When a user then defines a new network, Neutron automatically allocates a VLAN and takes care of the isolation for the user. The user does not have to manage VLANs, and does not need to be aware of which VLAN was assigned to the network.

Complete Software-Defined Network Solution

OpenStack, using Neutron, presents a complete SDN solution. Users can define isolated networks with any address space, and connect between those networks using virtual routers. Users can define firewall rules without the need to touch or change any element of the physical network topology. Furthermore, there is a complete abstraction between the physical topology and the virtual networks, so that multiple virtual networks can share the same physical resources, without any security or address space concerns.

User Isolation: Multitenancy

Allowing multiple users to share the same physical environment while ensuring complete separation between them is a key feature of OpenStack. OpenStack is designed in a way that many tenants can share the same physical resources without being aware that they do so. OpenStack offer ways to share virtual resources between tenants, but maintains complete separation where needed.

Oracle VM Basics

This section gives an introduction to the components of Oracle VM.

What Is Oracle VM Server?

Oracle VM Server is a component of the Oracle VM product. Oracle VM Server is based on the Xen hypervisor. Oracle VM Server can be managed using Oracle VM Manager or it can be managed as a standalone product with OpenStack. To better understand how Oracle VM Server integrates with OpenStack, it is necessary to understand how Xen works.

Xen Hypervisor

Xen is a bare-metal (type 1) hypervisor. The user can control the hypervisor from a privileged environment (which is also itself a virtual machine) called *Domain0*, or *Dom0*. Dom0 controls the physical devices on the system, and connects them to the other virtual machines. Dom0 is also responsible for launching virtual machines called *DomU(s)* using tools run from the user space. When launching a virtual machine, Xen connects DomU to storage and network resources.

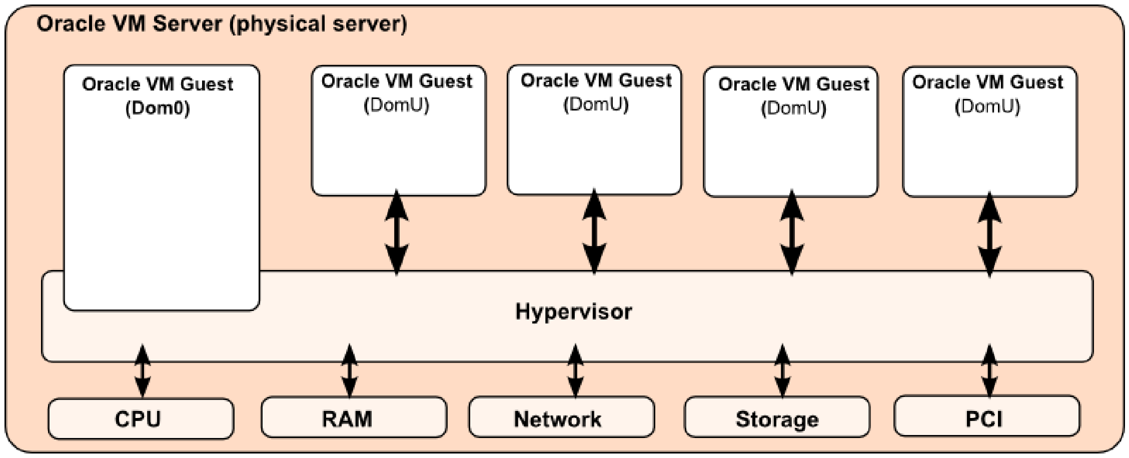


Figure 1. Oracle VM Server deployment on the Xen hypervisor

Since Dom0 is itself a virtual machine, any memory assigned to it cannot be used for the DomUs. Sufficient Dom0 memory is important for performance and correct operation, so it is important to adhere to the minimum memory requirements.

Managing Oracle VM Server with OpenStack

To connect Oracle VM Server to OpenStack, the libvirt driver is used. Xen is fully supported by the libvirt driver, so the integration is very natural. The version of Oracle VM Server used in this deployment is Oracle VM Server Release 3.3, which uses Oracle's Unbreakable Enterprise Kernel Release 3 (also known as, *UEK3*), a 3.8 upstream Linux kernel hardened and shipped by Oracle. As a result, Oracle VM Server uses the latest versions of the hardware drivers and Open vSwitch (for more details on Open vSwitch refer to http://openvswitch.org).

Deployment Options

To integrate Oracle VM Server with OpenStack, use the latest Oracle VM Release 3.3. For Oracle Linux, simply use the latest Oracle Linux 6. OpenStack supports various flexible deployment models, where each service can be deployed separately on a different node or where services can be installed together with other services. A user can set up any number of compute and control nodes to test the OpenStack environment.

OpenStack supports the following deployment models:

» **All-in-one node:** A complete installation of all the OpenStack services on an Oracle Linux node. This deployment model is commonly used to get started with OpenStack or for development purposes. In this model, the user has fewer options to configure, and the deployment does not require more than one node. This deployment model is not supported for production use.

» **One control node and one or more compute nodes:** This is a common deployment across multiple servers. In this case, all the control services are installed on Oracle Linux, while separate compute nodes are set up to run Oracle VM Server or Oracle Linux for the sole purpose of running virtual machines.

» **One control node, one network node, and one or more compute nodes:** Another common deployment configuration is when the network node is required to be separate from the rest of the services. This can be due to compliance or performance requirements. In this scenario, the network node is installed on Oracle Linux, and the rest of the management services are installed on a separate controller node. Compute nodes can be installed as required, as in all other cases.

» **Multiple control nodes with different services and one or more compute nodes:** As mentioned, OpenStack is very flexible, and there is no technical limitation that stops users from experimenting with more sophisticated deployment models. However, using one of the supported configurations reduces complexity.

To get started, Oracle recommends you use either the all-in-one model or the model with one control node and one or more compute nodes.

Setting Up the Environment

This section describes how to install the Oracle OpenStack for Oracle Linux environment.

Installing Oracle Linux or Oracle VM Server on a Compute Node

A compute node is a system running Oracle Linux using KVM, or Oracle VM Server Release 3.3. You can download installation ISOs of the latest version of Oracle Linux 6, or Oracle VM Server Release 3.3, from the Oracle Software Delivery Cloud at:

https://edelivery.oracle.com/linux

For instructions on installing Oracle Linux 6, see the *Oracle Linux Installation Guide for Release 6* at:

http://docs.oracle.com/cd/E37670_01/E41137/html/

For Oracle VM Server, the installation process is very similar to Oracle Linux, and can be done using kickstart or the interactive installer. For information on installing Oracle VM Server, see the *Oracle VM Installation and Upgrade Guide for Release 3.3* at:

http://docs.oracle.com/cd/E50245_01/E50247/html

After successfully installing the compute node, you might need to change the memory size of Dom0. As mentioned previously, Dom0 is the control domain of the Xen server. You can use this domain to control the rest of the virtual machines, as well as to manage hardware and additional software. Dom0 is where the OpenStack Nova compute components will be installed, and it should be configured with at least 4 GB of RAM.

You can check and change the amount of RAM by editing the `grub.conf` file in the boot directory. For example, edit the `/boot/grub/grub.conf` file to the following:

```
kernel /xen.gz . . . . dom0_mem=4096M
```

The following additional configuration should be performed on each compute note:

» **Local storage**: The default installation creates a 50 GB root partition, which is good for running a few small virtual machines. On the control node, Glance also needs space to store the images, so it is recommended that you mount another disk, partition, or NFS share at `/var/lib/glance/images/`.

  On the compute node, this space limitation prevents the running of virtual machines with larger disk space requirements. Therefore, it is recommended that an additional disk, partition, or NFS share, also be mounted at `/var/lib/nova/instances` where the images will run. A different partition table can be defined during a kickstart install, if required.

» **NTP:** It is recommended that you set NTP on the servers to point to your default NTP server or, alternatively, you can set NTP servers on all hosts during the deployment phase with the `packstack` option `–ntp-servers`.

» **Proxy:** If your installation repository is behind a proxy server, make sure you update the `/etc/yum.conf` file with the proxy server address. All nodes will access the installation repository, so it is important to make sure that yum can access the repositories through the proxy server on all the nodes.

Installing OpenStack Services on a Control Node

A *control node* is where most of the OpenStack services are installed. The term *control node* is used to discuss nodes that do *not* run virtual machines. The control nodes may have all the non-compute services or some only of them. In OpenStack, you can choose how to distribute the control services: one node for all the control services; or a node for Neutron, another for Keystone, another for Glance, and so on. The only exception to this is the all-in-one configuration, where all the services, including the compute services, are installed on the same node. The all-in-one model is often used for a demonstration or development environment, but it is not recommended for a production deployment.

In any configuration you use, the control node should be installed only on an Oracle Linux system, and not inside the Dom0 on Oracle VM Server.

Setting Up the Network

Network configuration tends to be the most complex area in OpenStack. Mistakes in network configuration can lead to complicated problems, so it is important to understand how OpenStack networking works. In this release of Oracle OpenStack for Oracle Linux and Oracle VM, Neutron is supported with Open vSwitch. With this configuration, all the services OpenStack provides can be used without external dependency on third-party components. This setup supports the physical separation of management and virtual machine networks. This is particularly important if the management network has less bandwidth. The management and virtual machine networks can share the same physical connection and be separated with VLANs.

All-in-One Model

The all-in-one configuration can be installed only on Oracle Linux, not on Oracle VM. For an all-in-one deployment, two physical network interface cards are required. The first network interface card must be configured with an IP address for managing the server and accessing the API and dashboard. The second card is used to allow instances to access the public network. The second network card will not have an IP address configured. If there are no plans to allow instances external connectivity, there is no need to have the second network interface card:

TABLE 1. ALL-IN-ONE ETHERNET PORTS

Ethernet Port	IP Address	Purpose
eth0	Yes	Connected to the management or public network to allow access to the OpenStack API
eth1	No	Connected to the public network and used by OpenStack to connect instances to the public network

One Control Node and Multiple Compute Nodes Model

When deploying the control and compute nodes separately, the control node should be configured as shown in the following table.

TABLE 2. CONTROL NODE ETHERNET PORTS

Ethernet Port	IP Address	Purpose
eth0	Yes	Connected to the management or public network to allow access to the OpenStack API
eth1	Yes	Connected to a private management network that connects all the compute nodes and the control node
eth2	No	Connected to a private VLAN network that allows instances running on the compute nodes to communicate with each other and with the public network through the control node
eth3	No	Connected to the public network and used to connect instances to the public network

The compute nodes should then be configured with three network cards as shown in the following table.

TABLE 3. COMPUTE NODES ETHERNET PORTS

Ethernet Port	IP Address	Purpose
eth0	Yes	Connected to the public network to allow: » Access to public repositories for installation » SSH connections to the compute node to perform monitoring and maintenance operations
eth1	Yes	Connected to a private management network that connects all the compute nodes and the management node
eth2	No	Connected to the public network and used to connect private VLANs to the public network

Deploying OpenStack Services

This section gives the steps to deploy the OpenStack services using both Oracle VM and Oracle Linux as compute nodes.

The required steps are as follows:

1. Install Oracle VM Server and Oracle Linux on the compute and control nodes.
2. Configure the network.
3. Download the repository file to point to the correct yum repository.
4. Run the installer.

The following section demonstrates how to perform this for Oracle VM compute nodes and for Oracle Linux compute nodes using KVM.

Deploying OpenStack with Oracle VM Server

For this example, the following configuration is used:

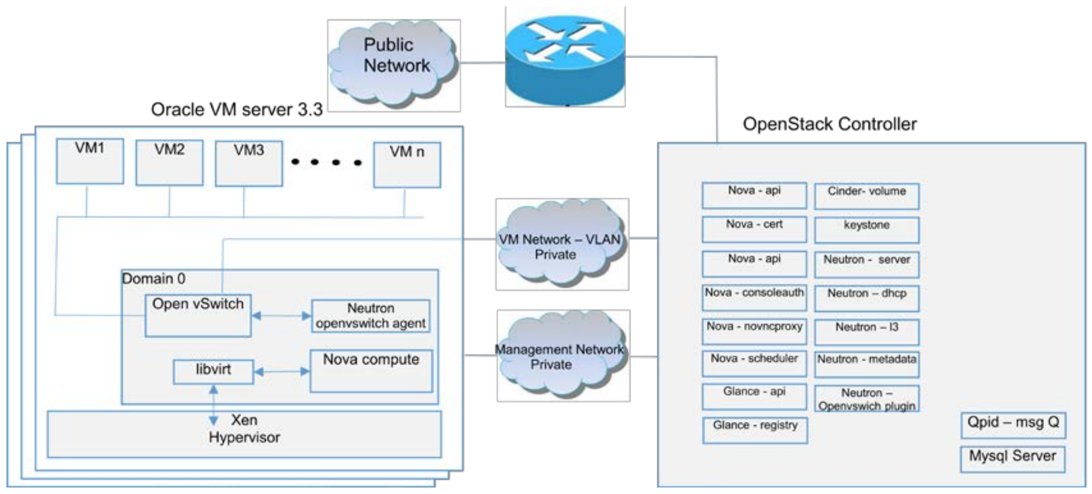


Figure 2. Deployment model to test Oracle VM using OpenStack

Figure 2 shows an example deployment model to test Oracle VM. The deployment includes an OpenStack controller (Oracle Linux), which has all the services installed, and any number of compute nodes (Oracle VM Servers).

To install the Oracle VM Server, Oracle Linux, and OpenStack components on the nodes, perform the following steps:

1. Install and configure Oracle VM Server as described in “Environment Setup” above. Make sure the network is configured according to the instructions in “Network Setup” above.
    a. For Oracle VM Servers, adjust the Dom0 memory to at least 2 GB. Restart the server for the change to take effect.
    b. For Oracle Linux, disable SELINUX and set `SELINUX=disabled` in `the /etc/selinux/config` file.

2. For all nodes in the environment, download the `public-yum-openstack-ol6.repo` file from public-yum.oracle.com, and place it in the `/etc/yum.repos.d/` directory.

3. Install the `packstack` tool on any host running Oracle Linux 6. It does not need to be installed on a compute node.

```
# yum install -y openstack-packstack
```

4. If you are setting up a control node, run the `packstack` command:

```
# packstack –-install-hosts=192.168.0.10,192.168.0.1,192.168.0.2 –ntp-servers=10.162.82.1 --
neutron-ovs-tenant-network-type=vlan --neutron-ovs-vlan-ranges=default:1000:2000 --neutron-ovs-
bridge-mappings=default:br-eth2 --neutron-ovs-bridge-interfaces=br-eth2:eth2 --novavncproxy-
hosts=<PUBLIC IP OF CONTROL NODE>
```

The example shows how the command might be run for a three-node installation (one control node and two compute nodes). The `packstack` parameters used in the example are explained in the table below.

TABLE 4. PACKSTACK PARAMETERS

Parameter	Description
`install-hosts=192.168.0.10,192.168.0.1,192.168.0.2`	In this case there are three nodes specified by network IP address: » 192.168.0.10: the control node » 192.168.0.1,2: the two compute nodes This tells `packstack` the location of the nodes it needs to connect to in order to perform the installation.
`ntp-servers=10.162.82.1`	Defines the NTP setup for all nodes to use particular servers, for example, 10.162.82.1.
`neutron-ovs-tenant-network-type=vlan`	Configures Neutron to use VLAN as the tenant network separation mechanism.
`neutron-ovs-vlan-ranges=default:1000:2000`	Sets the VLAN range available for Neutron to isolate networks to 1000-2000. Any range can be chosen.
`neutron-ovs-bridge-mappings=default:br-eth2`	Defines the bridge on the Open vSwitch to be used for guest traffic.
`neutron-ovs-bridge-interfaces=br-eth2:eth2`	Defines the physical port for virtual machine traffic.
`novavncproxy-hosts`	Defines the IP address of the control node that serves as a VNC proxy to send the console traffic from the servers to the user.

All done!

Example `packstack` Run

The following example output shows the complete `packstack` installation described in the previous section. The example shows the installation of a control node, two compute nodes, and a network node.

```
# packstack --install-hosts=192.168.0.10,192.168.0.1,192.168.0.2 -–ntp-servers=10.162.82.1 --neutron-
ovs-tenant-network-type=vlan --neutron-ovs-vlan-ranges=default:1000:2000 --neutron-ovs-bridge-
mappings=default:br-eth2 --neutron-ovs-bridge-interfaces=br-eth2:eth2 --novavncproxy-hosts=<PUBLIC IP OF
CONTROL NODE>

Welcome to Installer setup utility
Packstack changed given value to required value /root/.ssh/id_rsa.pub

Installing:
Clean Up...                                            [ DONE ]
Setting up ssh keys...root@192.168.0.2's password:
root@192.168.0.1's password:
root@192.168.0.10's password:
                                 [ DONE ]
Discovering hosts' details...                          [ DONE ]
Disabling NetworkManager...                            [ DONE ]
Adding pre install manifest entries...                 [ DONE ]
Adding MySQL manifest entries...                       [ DONE ]
Adding QPID manifest entries...                        [ DONE ]
Adding Keystone manifest entries...                    [ DONE ]
Adding Glance Keystone manifest entries...             [ DONE ]
Adding Glance manifest entries...                      [ DONE ]
Installing dependencies for Cinder...                  [ DONE ]
Adding Cinder Keystone manifest entries...             [ DONE ]
Adding Cinder manifest entries...                      [ DONE ]
Checking if the Cinder server has a cinder-volumes vg...[ DONE ]
Adding Nova API manifest entries...                    [ DONE ]
Adding Nova Keystone manifest entries...               [ DONE ]
Adding Nova Cert manifest entries...                   [ DONE ]
Adding Nova Conductor manifest entries...              [ DONE ]
Adding Nova Compute manifest entries...                [ DONE ]
Adding Nova Scheduler manifest entries...              [ DONE ]
Adding Nova VNC Proxy manifest entries...              [ DONE ]
Adding Nova Common manifest entries...                 [ DONE ]
Adding Openstack Network-related Nova manifest entries...[ DONE ]
Adding Neutron API manifest entries...                 [ DONE ]
Adding Neutron Keystone manifest entries...            [ DONE ]
Adding Neutron L3 manifest entries...                  [ DONE ]
Adding Neutron L2 Agent manifest entries...            [ DONE ]
Adding Neutron DHCP Agent manifest entries...          [ DONE ]
Adding Neutron Metadata Agent manifest entries...      [ DONE ]
Adding OpenStack Client manifest entries...            [ DONE ]
Adding Horizon manifest entries...                     [ DONE ]
Adding Ceilometer manifest entries...                  [ DONE ]
Adding Ceilometer Keystone manifest entries...         [ DONE ]
Adding post install manifest entries...                [ DONE ]
Preparing servers...                                   [ DONE ]
Installing Dependencies...                             [ DONE ]
Copying Puppet modules and manifests...                [ DONE ]
Applying Puppet manifests...
Applying 192.168.0.2_prescript.pp
Applying 192.168.0.1_prescript.pp
Applying 192.168.0.10_prescript.pp
Applying xxxxxxxx_prescript.pp
192.168.0.1_prescript.pp :                   [ DONE ]
xxxxxxxx_prescript.pp :                   [ DONE ]
192.168.0.10_prescript.pp :                 [ DONE ]
192.168.0.2_prescript.pp :                   [ DONE ]
Applying 192.168.0.10_mysql.pp
Applying 192.168.0.10_qpid.pp
192.168.0.10_mysql.pp :                         [ DONE ]
192.168.0.10_qpid.pp :                           [ DONE ]
Applying 192.168.0.10_keystone.pp
Applying 192.168.0.10_glance.pp
Applying 192.168.0.10_cinder.pp
192.168.0.10_keystone.pp :                   [ DONE ]
192.168.0.10_glance.pp :                       [ DONE ]
192.168.0.10_cinder.pp :                       [ DONE ]
Applying 192.168.0.10_api_nova.pp
```

```
192.168.0.10_api_nova.pp :                    [ DONE ]
Applying 192.168.0.10_nova.pp
Applying 192.168.0.2_nova.pp
Applying 192.168.0.1_nova.pp
Applying xxxxxxxx_nova.pp
192.168.0.10_nova.pp :                            [ DONE ]
xxxxxxxx_nova.pp :                              [ DONE ]
192.168.0.1_nova.pp :                              [ DONE ]
192.168.0.2_nova.pp :                              [ DONE ]
Applying 192.168.0.2_neutron.pp
Applying 192.168.0.1_neutron.pp
Applying 192.168.0.10_neutron.pp
192.168.0.10_neutron.pp :                      [ DONE ]
192.168.0.1_neutron.pp :                        [ DONE ]
192.168.0.2_neutron.pp :                        [ DONE ]
Applying 192.168.0.10_osclient.pp
Applying 192.168.0.10_horizon.pp
Applying 192.168.0.10_ceilometer.pp
192.168.0.10_osclient.pp :                    [ DONE ]
192.168.0.10_horizon.pp :                      [ DONE ]
192.168.0.10_ceilometer.pp :                [ DONE ]
Applying 192.168.0.2_postscript.pp
Applying 192.168.0.1_postscript.pp
Applying 192.168.0.10_postscript.pp
Applying xxxxxxxx_postscript.pp
192.168.0.2_postscript.pp :                 [ DONE ]
xxxxxxxx_postscript.pp :                 [ DONE ]
192.168.0.1_postscript.pp :                 [ DONE ]
192.168.0.10_postscript.pp :                [ DONE ]
                           [ DONE ]
Finalizing...                                          [ DONE ]

 **** Installation completed successfully ******


Additional information:
 * A new answerfile was created in: /root/packstack-answers-20140504-032716.txt
 * Time synchronization installation was skipped. Please note that unsynchronized time on server
instances might be problem for some OpenStack components.
 * File /root/keystonerc_admin has been created on OpenStack client host 192.168.0.10. To use the
command line tools you need to source the file.
 * To access the OpenStack Dashboard browse to http://192.168.0.10/dashboard.
Please, find your login credentials stored in the keystonerc_admin in your home directory.
 * Because of the kernel update the host 192.168.0.2 requires reboot.
 * Because of the kernel update the host 192.168.0.1 requires reboot.
 * Because of the kernel update the host 192.168.0.10 requires reboot.
 * Because of the kernel update the host xxxxxxxx requires reboot.
 * The installation log file is available at: /var/tmp/packstack/20140504-032716-R2G8be/openstack-
setup.log
 * The generated manifests are available at: /var/tmp/packstack/20140504-032716-R2G8be/manifests
```

.

All done!

Note that on the OpenStack dashboard, the hypervisor type will show *QEMU* and not *KVM*.

Post-Installation Steps

After the installation process completes, the deployment is ready. There are several things you need to know about the installation:

» **Dashboard:** The OpenStack dashboard is available at:

  http://<public IP address of management node or its FQDN>/dashboard

» **`keystonerc_admin` file:** At the end of the installation, `packstack` automatically creates a file with this name in the home directory of the installing user (`root` in the example above). The following is an example of the contents of the file:

```
export OS_USERNAME=admin
export OS_TENANT_NAME=admin
export OS_PASSWORD=318ea4fd2e324171
export OS_AUTH_URL=http://192.168.0.10:35357/v2.0/
export PS1='[\u@\h \W(keystone_admin)]\$ '
```

  Sourcing this file allows you to use the command line without the need to deal with a password for every command. Also, the password here is the password to be used when logging in to the OpenStack dashboard.

» **Logging:** The logs for the OpenStack services are located at `/var/log/<service>/`. Debug mode generates more debug information in the log file, which might make the logs difficult to read. Debug mode is disabled by default in the `*.conf` file (for example, `/etc/nova/nova.conf` is set to `false`).

» **Connecting to an external network:** To connect instances to the public network, connect a network interface from the `br-ex` bridge on the Open vSwitch to an Ethernet port which is connected to the public network. To do this, use the following command on the control node:

```
# ovs-vsctl add-port br-ex eth3
```

  In this example, `eth3` is connected to the external network and has no IP address configured.

» **Enabling metadata service** (optional): OpenStack provides a metadata service that allows instances to receive identity and configuration information after they are instantiated. To enable the metadata service, run the following commands on the control node:

```
# openstack-config --set /etc/neutron/dhcp_agent.ini DEFAULT enable_isolated_metadata true
# service neutron-dhcp-agent restart
```

Testing OpenStack Features

This section shows you how to test the OpenStack features. In this example, Oracle VM Server is used as a compute node.

Creating an Instance

This section shows the steps to create the first instance.

1. **Import an image into OpenStack:** An image can be described as a virtual disk containing an installed operating system. To create one, you can use any platform or tool that can generate a virtual disk in raw format. For this example, a virtual disk from an Oracle VM environment created by Oracle VM Manager, called `ol6_pvm.img`, was used. This is an Oracle VM guest installed with Oracle Linux 6. It is recommended that you make sure this image is configured for DHCP networking so it can consume the IP address assigned automatically by Neutron.

To upload the image, use the following command:

```
~(keystone_admin)]#  glance image-create --name ol65 --disk-format=raw --container-format=bare <
ol65-pvm.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | f037b34db632c3b603802951edf1ca83     |
| container_format | bare                                 |
| created_at       | 2014-09-15T00:53:21                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | raw                                  |
| id               | e9b5d544-9818-4d72-8fc9-63be7f723e54 |
| is_public        | False                                |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | ol65                                 |
| owner            | 6e933bb96b0f443791dbeb2fc2dfc299     |
| protected        | False                                |
| size             | 1395864320                           |
| status           | active                               |
| updated_at       | 2014-09-15T00:53:55                  |
| virtual_size     | None                                 |
+------------------+--------------------------------------+
```

Remember, you need to first source `keystonerc_admin`.

If the image is of an HVM (hardware virtual machine) guest, you need to add a property designating it as such:

```
# glance image-update <image ID> --property vm_mode=hvm
```

2. **Create a network:** The network is discussed in detail below. To launch an instance, at least one network must be created. To create a network called `net1` with a subnet 10.10.10.0/24, run the following commands:

```
# export tenant_id=$(keystone tenant-list | grep admin| awk ' {print $2} ')
# neutron net-create --tenant-id=$tenant_id net1
# neutron subnet-create --tenant-id=$tenant_id net1 10.10.10.0/24
```

3. **Add rules to security groups to allow `ssh` and `ping`** (optional but recommended): To make sure that it is possible to communicate with the instance when it is up and running, adding the following rules to the default security groups is recommended:

```
# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
# nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
```

4. **Launch the instance:** This can be done from either the command line or the OpenStack dashboard. The following example shows using the command line:

```
# nova boot --image ol65 --flavor 1 ol65 --nic net-id=$net_id
```

The value of `$net-id` can be obtained by running the following command:

```
~(keystone_admin)]# neutron net-list
+--------------------------------------+------+-------------------------------------- ----------
| id                                   | name | subnets
|
+--------------------------------------+------+-------------------------------------- ----------
| c61c4043-5b2b-46ce-8316-e35e7daee378 | net1 | 0236def9-5e74-45c1-8edb-c6c32d2c8363
  10.10.10.0/24 |
+--------------------------------------+------+--------------------------------------
```

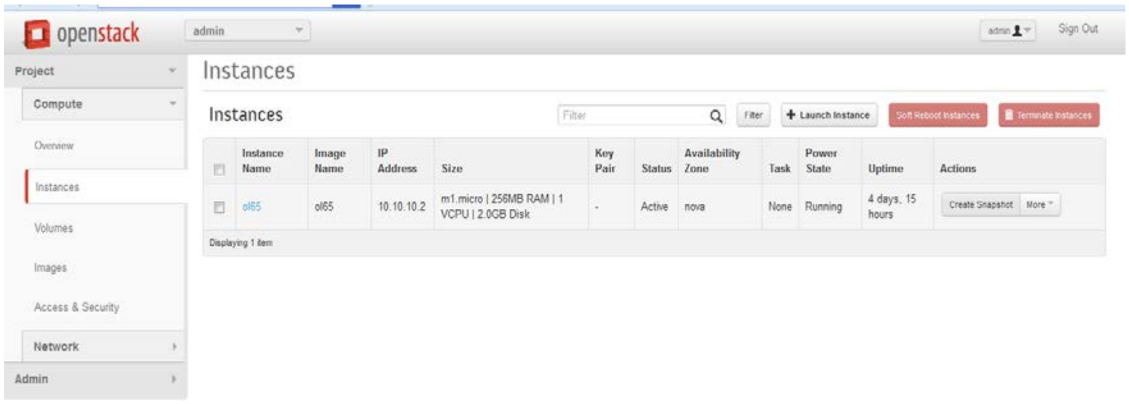
The `flavor` parameter, used in the command above, is the way OpenStack describes the size of the virtual machine. In the example `--flavor 1` was used, which maps onto the values for the flavor with an ID equivalent to 1, as obtained from the following command:

```
~(keystone_admin)# nova flavor-list
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.micro  | 256       | 2    | 0         |      | 1     | 1.0         | True      |
| 2  | m1.tiny   | 512       | 10   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 4  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 5  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 6  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+
```

After launching the instance, the image is copied to the run area. While it is copying, the instance is in BUILD status:

```
~(keystone_admin)]# nova list
+--------------------------------------+------+--------+-----------+------------+-------------
| ID                                   | Name | Status | Task State| Power State| Networks
|
+--------------------------------------+------+--------+-----------+------------+-------------
| 8947001c-348b-4487-9c7c-a1d358f50eb4 | ol65 | ACTIVE | -         | Running    | net1=10.10.10.5
|+--------------------------------------+------+--------+-----------+------------+-------------
```

From the Dashboard, you can monitor the instance build from the **Instances** tab:



Instances

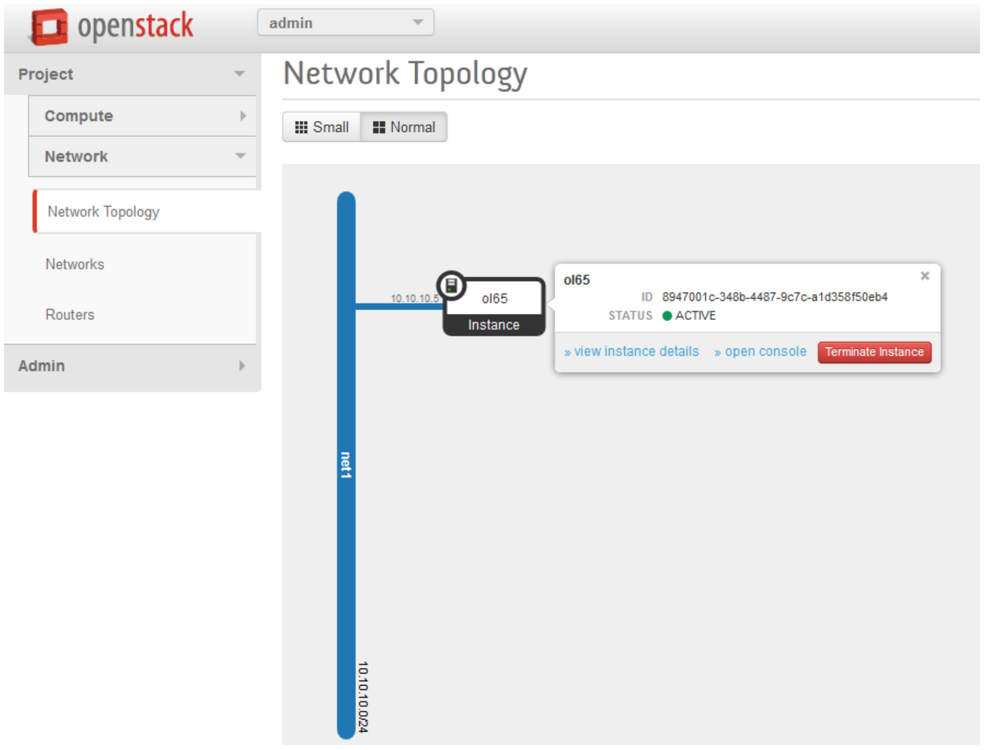
Instances

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
ol65	ol65	10.10.10.2	m1.micro | 256MB RAM | 1 VCPU | 2.0GB Disk	-	Active	nova	None	Running	4 days, 15 hours	Create Snapshot More

Displaying 1 item

Figure 3. OpenStack Instances tab

Or you can monitor it from the **Network Topology** view:



ol65
ID 8947001c-348b-4487-9c7c-a1d358f50eb4
STATUS ● ACTIVE

Figure 4. OpenStack Network Topology view

Note that the instance is automatically assigned an IP address from `net1`'s subnet.

At this point, you can open a console from either the **Instances** tab or the **Network Topology** view:

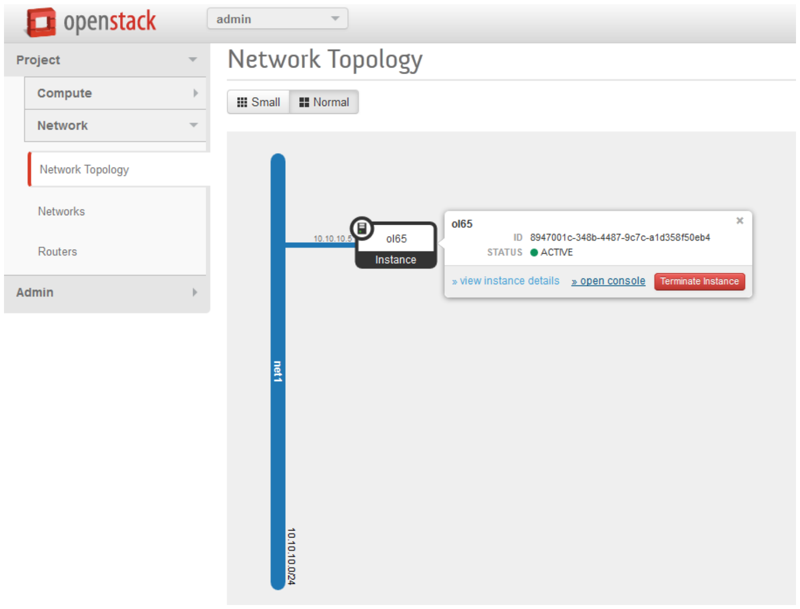
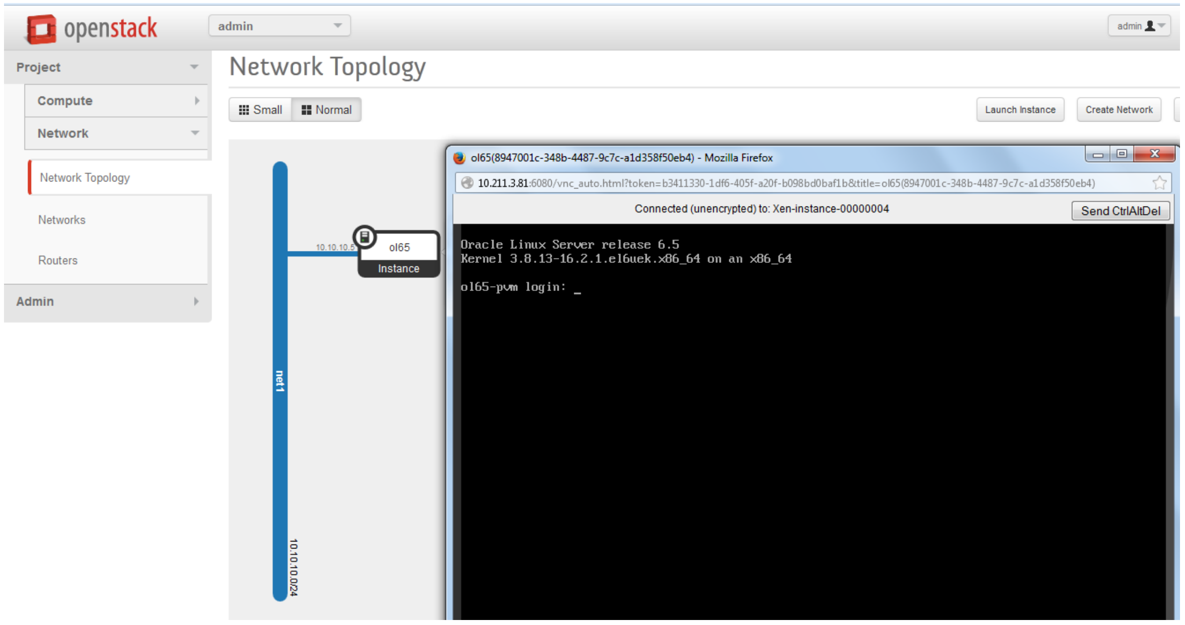


Figure 5. Opening a console from the OpenStack Network Topology view

Click **open console** to show the console.


openstack
admin
admin
Project
Compute
Network
Network Topology
Networks
Routers
Admin
Network Topology
Small
Normal
Launch Instance
Create Network
net1
10.10.10.0/24
10.10.10.5
ol65
Instance
ol65(8947001c-348b-4487-9c7c-a1d358f50eb4) - Mozilla Firefox
10.211.3.81:6080/vnc_auto.html?token=b3411330-1df6-405f-a20f-b098bd0baf1b&title=ol65(8947001c-348b-4487-9c7c-a1d358f50eb4)
Connected (unencrypted) to: Xen-instance-00000004
Send CtrlAltDel

```
Oracle Linux Server release 6.5
Kernel 3.8.13-16.2.1.el6uek.x86_64 on an x86_64

ol65-pvm login: _
```

Figure 6. Console displayed from the OpenStack Network Topology view

Exploring Network Features

As described earlier, OpenStack has a large set of networking capabilities that comprehensively cater to the requirements of most environments. This section explores some of those features and illustrates how to test them with Oracle VM Server.

OpenStack allows users to add software-defined routers to route traffic between isolated networks. The following steps describe how to create a router and test connectivity between two virtual machines placed on two different subnets.

1. Using the steps described in "Creating an Instance," create a new virtual machine on a separate network called `net2`. Use a different subnet, for example, 20.20.20.0/24. The **Network Topology** tab should look like Figure 7:

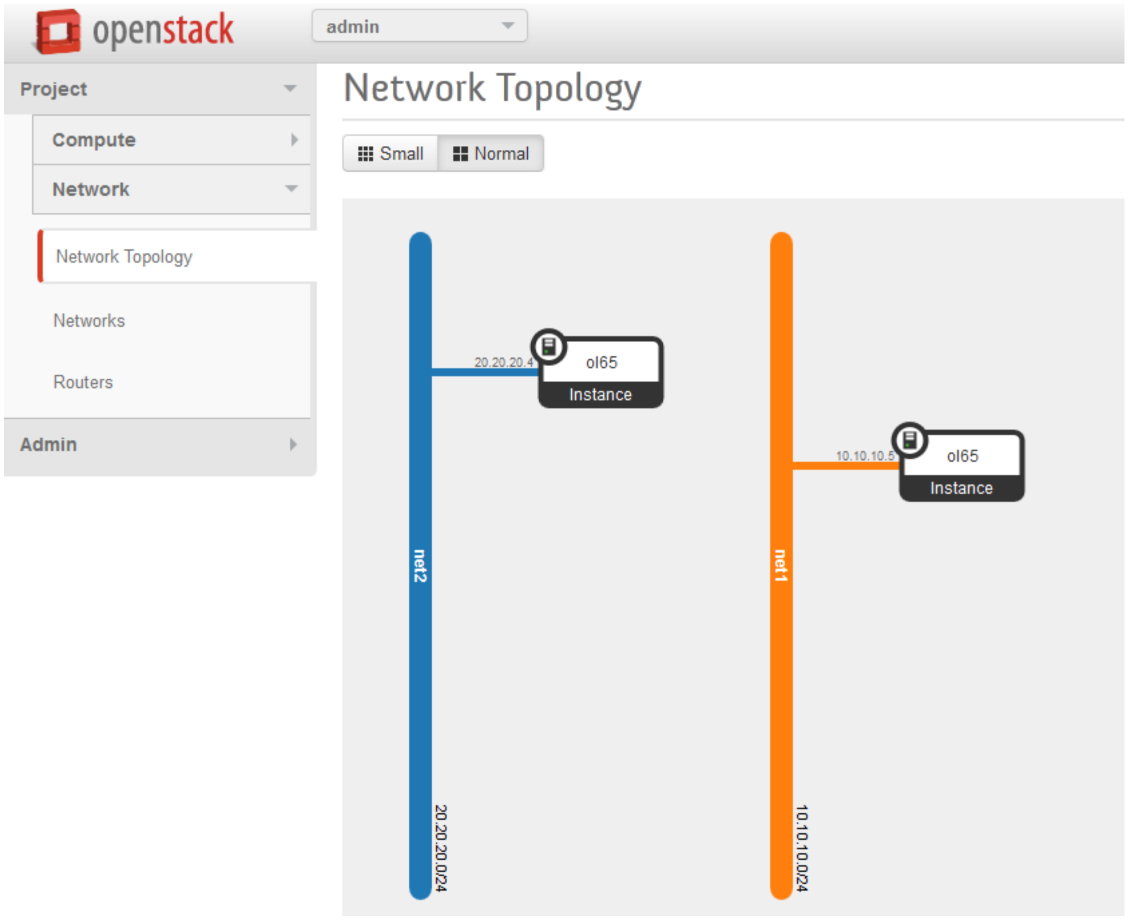


Figure 7. OpenStack Network Topology view with two subnets

2. Now create a router using the **router** button. After creating a router, go to the **Routers** tab and add interfaces from both subnets. The result looks like Figure 8:

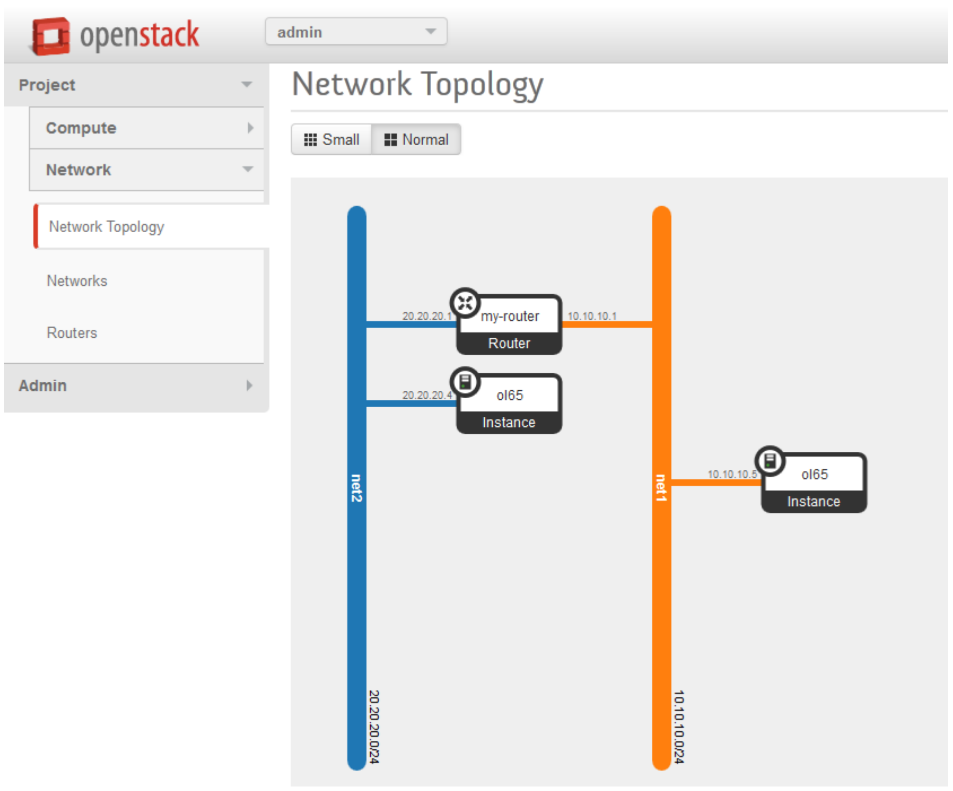
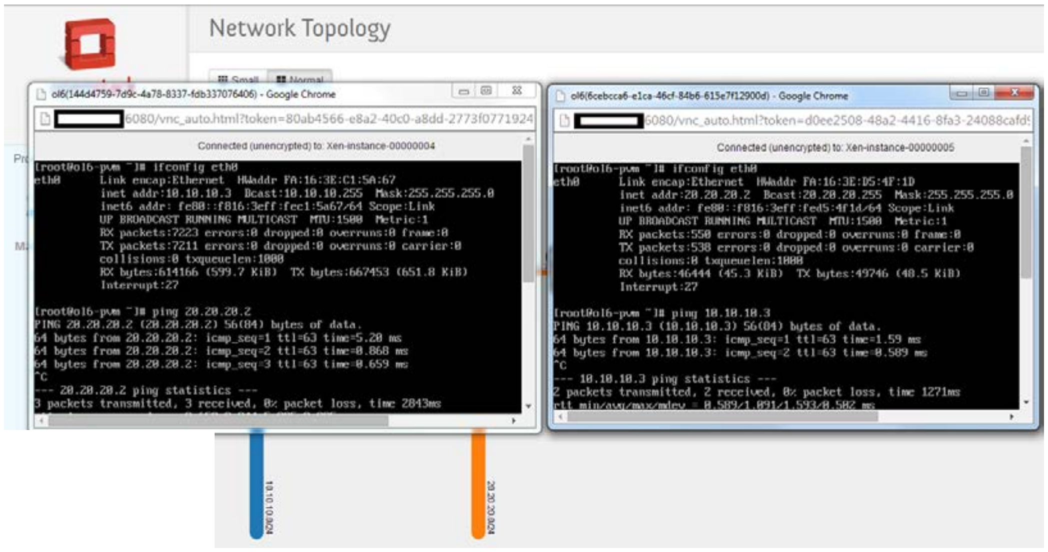


Figure 8. OpenStack Network Topology view after adding a router

Figure 8 shows the router with two interfaces. Near the router you can see the gateway IP addresses, which were defined when creating `net1` and `net2` (10.10.10.1 and 20.20.20.1).

3. Now that the two subnets are connected with a router, you can run the `ping` and `ssh` commands from one instance to the other, as shown in Figure 9.


Network Topology
Small
Normal
ol6(144d4759-7d9c-4a78-8337-fdb337076406) - Google Chrome
[redacted]6080/vnc_auto.html?token=80ab4566-e8a2-40c0-a8dd-2773f0771924
Connected (unencrypted) to: Xen-instance-00000004
ol6(6cebcca6-e1ca-46cf-84b6-615e7f12900d) - Google Chrome
[redacted]6080/vnc_auto.html?token=d0ee2508-48a2-4416-8fa3-24088cafd
Connected (unencrypted) to: Xen-instance-00000005
10.10.10.0/24
20.20.20.0/24

```
[root@ol6-pvm ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:C1:5A:67
          inet addr:10.10.10.3  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fec1:5a67/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7223 errors:0 dropped:0 overruns:0 frame:0
          TX packets:7211 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:614166 (599.7 KiB)  TX bytes:667453 (651.8 KiB)
          Interrupt:27

[root@ol6-pvm ~]# ping 20.20.20.2
PING 20.20.20.2 (20.20.20.2) 56(84) bytes of data.
64 bytes from 20.20.20.2: icmp_seq=1 ttl=63 time=5.20 ms
64 bytes from 20.20.20.2: icmp_seq=2 ttl=63 time=0.868 ms
64 bytes from 20.20.20.2: icmp_seq=3 ttl=63 time=0.659 ms
^C
--- 20.20.20.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2843ms
```

```
[root@ol6-pvm ~]# ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:D5:4F:1D
          inet addr:20.20.20.2  Bcast:20.20.20.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fed5:4f1d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:550 errors:0 dropped:0 overruns:0 frame:0
          TX packets:538 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:46444 (45.3 KiB)  TX bytes:49746 (48.5 KiB)
          Interrupt:27

[root@ol6-pvm ~]# ping 10.10.10.3
PING 10.10.10.3 (10.10.10.3) 56(84) bytes of data.
64 bytes from 10.10.10.3: icmp_seq=1 ttl=63 time=1.59 ms
64 bytes from 10.10.10.3: icmp_seq=2 ttl=63 time=0.589 ms
^C
--- 10.10.10.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1271ms
rtt min/avg/max/mdev = 0.589/1.091/1.593/0.502 ms
```

Figure 9. Consoles showing two subnets communicating with each other

4. Next, create a floating IP address and associate it with an instance. This allows you to connect to the instance from outside the OpenStack deployment.

    a. The first step is to create a public network from the command line. On the control node, run the following command:

```
(keystone_admin)]#  neutron net-create public --router:external=True
Created a new network:
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | True                                 |
| id                        | 3aef4506-c027-465e-9a31-8d21a4b95853 |
| name                      | public                               |
| provider:network_type     | local                                |
| provider:physical_network |                                      |
| provider:segmentation_id  |                                      |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | 6e933bb96b0f443791dbeb2fc2dfc299     |
+---------------------------+--------------------------------------+
```

    b. Next, create a subnet, which has an IP addresses on the public network. To do so, use the following command, which allocates a range of IP addresses to use from a public network:

```
# neutron subnet-create public xx.xx.xx.xx/xx --name public_subnet --enable_dhcp=False --
allocation-pool start=<First IP in the range>,end=<Last IP in the range> --gateway=<gateway of the
public network>
```

c. Now you need to go to the router to set up a gateway. You can do this from the dashboard. Neutron selects the first available IP address in the range to use as a gateway for the router:

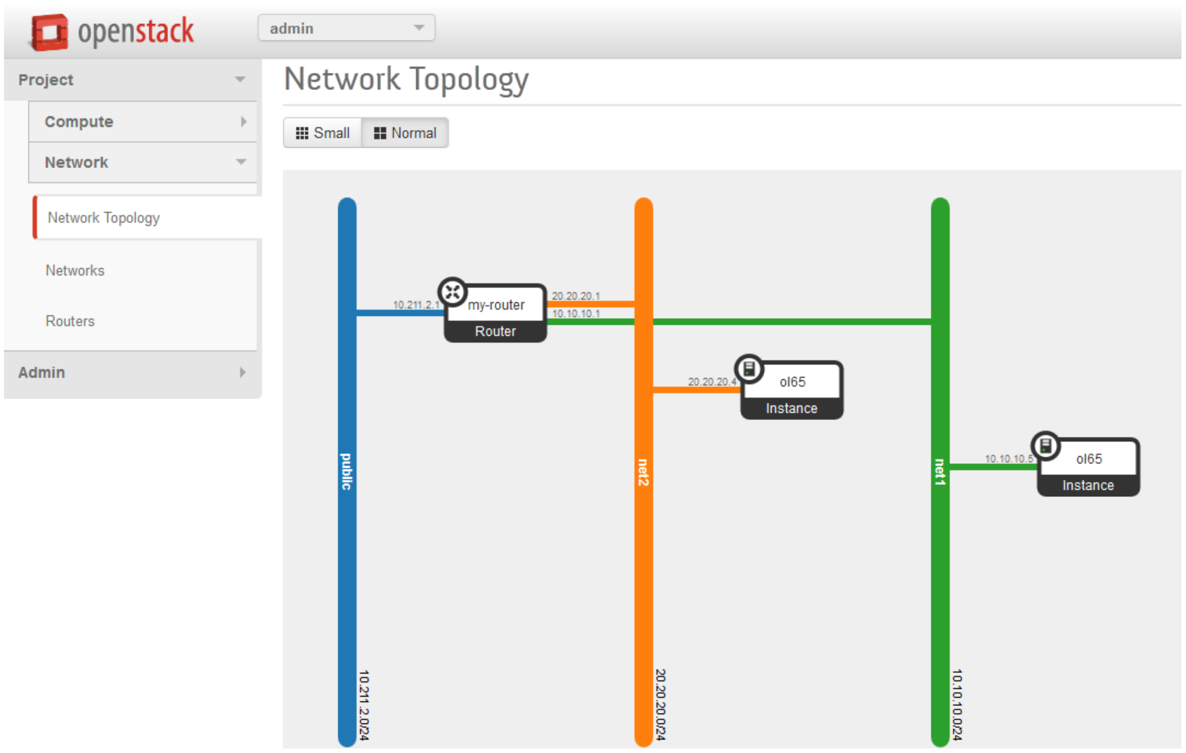


Figure 10. OpenStack Network Topology view showing the gateway configured

d. The next step is to generate a floating IP address. This is done from the command line:

```
~(keystone_admin)# neutron floatingip-create public
Created a new floatingip:
+---------------------+--------------------------------------+
| Field               | Value                                |
+---------------------+--------------------------------------+
| fixed_ip_address    |                                      |
| floating_ip_address | xxxxxxxx.58                          |
| floating_network_id | 4296fb7f-64f0-4297-ba9d-2f1fa45551ef |
| id                  | 4431db13-ed42-46b3-af53-4a455e9052ea |
| port_id             |                                      |
| router_id           |                                      |
| tenant_id           | ceb1d97df33642f78e8678149ce32903     |
+---------------------+--------------------------------------+
```

e. From the dashboard, you can now associate the floating IP address with the virtual machine. Go to the **Instances** tab, and choose the instance with which to associate the floating IP address. Use the **More** menu to associate a floating IP address with the instance.

5. Then try to log into the instance from an external source to confirm you can reach the instance.

Exploring Storage Features

OpenStack supports many storage devices. The following example shows how to use standard NFS as a persistent storage solution. The service that handles persistent storage is called Cinder. Like the rest of the OpenStack services, Cinder is also a pluggable framework, allowing vendors to create plug-ins to control their storage devices.

To use NFS, use the standard NFS driver. This driver is not geared towards a specific vendor, so it can be used for any NFS storage. Because it is a generic driver without specific knowledge of the back-end storage device, it cannot perform actions that are specific to the array, such as creating a snapshot, cloning, and performing a backup.

For this exercise, you need to have some NFS shares that can be mounted by the Oracle VM Servers.

1. The first step is to configure Cinder to use NFS and tell it where the NFS shares are located:

```
# openstack-config --set /etc/cinder/cinder.conf DEFAULT volume_driver
cinder.volume.drivers.nfs.NfsDriver
# openstack-config --set /etc/cinder/cinder.conf DEFAULT nfs_shares_config /etc/cinder/shares.conf
```

   In this example, Cinder is configured to look at `/etc/cinder/shares.conf` to find the list of available shares such as the following:

```
#cat /etc/cinder/shares.conf
192.168.0.10:/export/shares1
192.168.0.10:/export/shares2
```

   The next step is to restart the Cinder service for the changes to take effect:

```
# service openstack-cinder-volume restart
```

2. Creating a volume is a quick and easy operation that can be performed from the command line as follows:

```
~(keystone_admin)]# cinder create --display-name my-new-volume 5
+---------------------+--------------------------------------+
|       Property      |                Value                 |
+---------------------+--------------------------------------+
|     attachments     |                  []                  |
|  availability_zone  |                 nova                 |
|       bootable      |                false                 |
|      created_at     |      2014-09-15T17:30:26.097772      |
| display_description |                 None                 |
|     display_name    |            my-new-volume             |
|      encrypted      |                False                 |
|          id         | fa9cc808-8428-430b-a18e-e3b7703e99d9 |
|       metadata      |                  {}                  |
|         size        |                  5                   |
|     snapshot_id     |                 None                 |
|     source_volid    |                 None                 |
|        status       |               creating               |
|     volume_type     |                 None                 |
+---------------------+--------------------------------------+
```

   After the volume is created, it can be viewed either in the dashboard or by using the `cinder list` command:

```
~(keystone_admin)]# cinder list
+--------------------------------------+-----------+---------------+------+-------------+---------
-+-------------+
|                  ID                  |   Status  |  Display Name | Size | Volume Type | Bootable
| Attached to |
+--------------------------------------+-----------+---------------+------+-------------+---------
-+-------------+
| 82277594-ec5c-454b-864f-546640bae77b | available | my-new-volume |  5   |     None    |  false
|             |
+--------------------------------------+-----------+---------------+------+-------------+---------
-+-------------+
```

3. Now the volume must be connected to an instance. To do this, use the `nova` command, which takes the instance ID, volume ID, and device to attach the volume to the instance. You can use `auto` to let Nova choose which device should be used.

   a. To obtain a list of instance IDs, run the following command:

```
~(keystone_admin)]# nova list
+--------------------------------------+------+--------+------------+-------------+---------------+
| ID                                   | Name | Status | Task State | Power State | Networks
|
+--------------------------------------+------+--------+------------+-------------+---------------+
| 8947001c-348b-4487-9c7c-a1d358f50eb4 | ol65 | ACTIVE | -          | Running     | net1=10.10.10.2
|
| 753fe279-d5a9-4b69-90cc-06f6f64ca840 | ol65 | ACTIVE | -          | Running     | net2=20.20.20.2
+--------------------------------------+------+--------+------------+-------------+---------------+
```

   b. After selecting the ID of the instance, run the following command to connect the instance. This will also change the status of the volume from `available` to `in-use`.

```
# nova volume-attach 753fe279-d5a9-4b69-90cc-06f6f64ca840 82277594-ec5c-454b-864f-546640bae77b auto
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/sdb                             |
| id       | 82277594-ec5c-454b-864f-546640bae77b |
| serverId | 753fe279-d5a9-4b69-90cc-06f6f64ca840 |
| volumeId | 82277594-ec5c-454b-864f-546640bae77b |
+----------+--------------------------------------+
```

4. From within the instance, the volume is accessible as `/dev/xvdb`, and it can be formatted and mounted as a normal device, for example:

```
# ls /dev/xvdb
/dev/xvdb

# mkfs -t ext3 /dev/xvdb

# mount /dev/xvdb my-drive/
```

5. Another interesting feature is the capability of booting from a volume. This allows you to create an instance that remains persistent after it is terminated. The difference between persistent and ephemeral storage is that with persistent storage, the boot process does not require copying an image, making it significantly faster than the boot process for ephemeral storage. To create a bootable volume, run the following command:

```
# cinder create --display-name my-new-bootable-volume --image-id e9b5d544-9818-4d72-8fc9-
63be7f723e54 4
```

   The command finishes immediately, and then the image is copied. The volume is ready to use when the status is `available`:

```
~(keystone_admin)]# cinder list
+--------------------------------------+-----------+------------------------+------+-------------+-
---------+--------------------------------------+
|                  ID                  |   Status  |      Display Name      | Size | Volume Type |
Bootable |             Attached to              |
+--------------------------------------+-----------+------------------------+------+-------------+-
---------+--------------------------------------+
| 2875856e-7595-4acb-b70c-0f6b3d4b19bf | available | my-new-bootable-volume |  4   |     None    |
false    |                                      |
| 82277594-ec5c-454b-864f-546640bae77b |   in-use  |     my-new-volume      |  5   |     None    |
false    | 753fe279-d5a9-4b69-90cc-06f6f64ca840 |
+--------------------------------------+-----------+------------------------+------+-------------+-
---------+--------------------------------------+
```

6. Now an instance can be easily launched from this volume.

   a. First, find out which network ID to use to attach the instance to one of your defined networks:

```
~(keystone_admin)]# nova net-list
+--------------------------------------+--------+------+
| ID                                   | Label  | CIDR |
+--------------------------------------+--------+------+
| af29b504-a6a6-41a3-b91d-535c82d41dff | public | -    |
| c61c4043-5b2b-46ce-8316-e35e7daee378 | net1   | -    |
| c8b7b482-aab2-4356-9678-284e10137435 | net2   | -    |
+--------------------------------------+--------+------+
```

   b. When you have obtained the network ID, run the following command to start the instance:

```
(keystone_admin)]# nova boot --boot-volume 2875856e-7595-4acb-b70c-0f6b3d4b19bf --flavor 1 ol6 --
nic net-id=c61c4043-5b2b-46ce-8316-e35e7daee378
+--------------------------------------+--------------------------------------------------+
| Property                             | Value                                            |
+--------------------------------------+--------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                           |
| OS-EXT-AZ:availability_zone          | nova                                             |
| OS-EXT-SRV-ATTR:host                 | -                                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000003                                |
| OS-EXT-STS:power_state               | 0                                                |
| OS-EXT-STS:task_state                | scheduling                                       |
| OS-EXT-STS:vm_state                  | building                                         |
| OS-SRV-USG:launched_at               | -                                                |
| OS-SRV-USG:terminated_at             | -                                                |
| accessIPv4                           |                                                  |
| accessIPv6                           |                                                  |
| adminPass                            | fEnjDZj77s33                                     |
| config_drive                         |                                                  |
| created                              | 2014-09-19T20:45:08Z                             |
| flavor                               | m1.tiny (1)                                      |
| hostId                               |                                                  |
| id                                   | 3d438eab-4a8b-46e2-a689-a94112f8dc51             |
| image                                | Attempt to boot from volume - no image supplied  |
| key_name                             | -                                                |
| metadata                             | {}                                               |
| name                                 | ol6                                              |
| os-extended-volumes:volumes_attached | [{"id": "2875856e-7595-4acb-b70c-0f6b3d4b19bf"}] |
| progress                             | 0                                                |
| security_groups                      | default                                          |
| status                               | BUILD                                            |
| tenant_id                            | 6e933bb96b0f443791dbeb2fc2dfc299                 |
| updated                              | 2014-09-19T20:45:08Z                             |
| user_id                              | 58a6e616182a4915b0dd3783f4169d33                 |
+--------------------------------------+--------------------------------------------------+
```

The boot-from volume is very fast and usually the instance is up and running in a matter of seconds.

Summary

This document guided you through different options of deploying OpenStack on Oracle VM and on Oracle Linux. It started by describing some basic features of OpenStack and Oracle VM and describing how to set them up for an OpenStack deployment. It continued by providing the installation steps and demonstrating how to use several features on the network and storage stacks.

OpenStack has a rich feature set and various services. After going through this guide, you should have a good foundation from which to learn more about the different OpenStack features and experiment with them in the environment you created. OpenStack will continue to evolve by adding new services and further developing existing services. To find out more about existing and new OpenStack features, see the OpenStack documentation available at:

http://docs.openstack.org/

As you might have seen, networking seems to be the most challenging and sophisticated area in OpenStack. It is recommended that you spend some time learning more about the network stack and understanding how networking is structured.