An Oracle White Paper
August 2014

Getting Started with Oracle VM, Oracle Linux and OpenStack

Technology Preview

Contents

Introduction 1
- Who Should Use this Whitepaper? 1

OpenStack Basics 1
- What is OpenStack? 1
- OpenStack Services 2
- Instances 2
- Storage in OpenStack 3
- Networking in OpenStack 3
- User Isolation Multi Tenancy 4

Oracle VM Basics 4
- What is Oracle VM Server? 4
- Xen Hypervisor 4
- Managing Oracle VM with OpenStack 5

Supported Configuration 5

Environment Setup 6
- Installing a Compute Node 6
- Installing a Control Node 6
- Network Set Up 7
- All-in-One 7
- Control and Multiple Compute Nodes 7

Deploying OpenStack Services 8
- Deploying OpenStack using Oracle VM Compute Nodes 8
- Deploying OpenStack using Oracle Linux Compute Nodes 11

Post Installation 12

Testing OpenStack Features 12
- Creating the First Instance 13
- Exploring Network Features 16
- Exploring Storage Features 19

Summary 21

Introduction

This whitepaper guides readers through the set up of an OpenStack deployment with Oracle VM and Oracle Linux. OpenStack is a very flexible system that can be configured, deployed, and used in many ways. This whitepaper is not a comprehensive resource for every possible option and feature in OpenStack, but provides guidelines to easily build a multi-node OpenStack deployment using Oracle VM and Oracle Linux.

This whitepaper is comprised of two main parts. The first part guides the reader through the deployment and configuration of OpenStack with Oracle VM.. The second part shows how to test various features of OpenStack, including launching instances and using networking and storage features. For the purpose of this technology preview, we have chosen a simple configuration that includes a control node and multiple compute nodes. While it is possible to configure OpenStack in various different ways with the tools we describe here, we chose to limit the discussion to a simplified configuration, allowing you to discover the possibilities of running Oracle VM and Oracle Linux with OpenStack.

Who Should Use this Whitepaper?

This whitepaper is written for users who would like to get started with OpenStack, and have little to no knowledge about it. In this whitepaper we do not assume prior knowledge of OpenStack, or Oracle VM. Therefore, this guide can also be used by anyone taking first steps into the world of OpenStack.

The whitepaper is recommended to customers who are interested in introducing OpenStack into their environment. Using this whitepaper, the reader is able to put together a deployment in a short time, and test it to see whether Oracle VM or Oracle Linux fits their requirements. While customers can deploy compute nodes using Oracle Linux and Oracle VM, Oracle strongly recommends using Oracle VM.

The whitepaper 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 whitepaper 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, which allows users to connect various technologies and components from different vendors and expose a unified API, regardless of the underlying technology. With OpenStack, you can manage different kinds 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 that can communicate with each other, and which are responsible for the different functions required from a virtualization/cloud management system. Some of the key services of OpenStack are:

- **Nova** – A compute service, responsible for creating instances and 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. Capable of connecting with vendor network gear 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. Capable of connecting to vendor storage gear 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 the images uploaded by the users. Glance is not a storage service but is responsible for saving image attributes.
- **Horizon** – A Dashboard, creates a GUI for users to be able 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:

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

OpenStack has **many** more services which are responsible for various features and capabilities, and the full list can be found on the OpenStack web site at:

http://www.openstack.org/

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

Instances

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

OpenStack can support both a persistent and ephemeral models. In the ephemeral model, an instance is launched from an image, the image is copied to the run area and once the copy completes, 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 to be able to scale out quickly, and maintain 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, block device, 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 case, 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.

Storage in OpenStack

As already mentioned, 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 *volume*, regardless of the technology and device it is backed by. Persistent storage can either be used to launch an instance or 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. Once 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 offering 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 this is not covered in this whitepaper.

Networking in OpenStack

This section gives an introduction to networking in OpenStack.

Network Services

Networking in OpenStack is one of the most powerful and sophisticated feature sets. The OpenStack networking service, Neutron, offers a complete SDN solution along with various network services, 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 different services. A network vendor that would like to offer a custom load balancer, instead of the default load balancer provided by Neutron, can do so. This gives a user a powerful tool to build sophisticated network topologies with standard APIs.

Network Isolation – Tenant Networks

The 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 GRE tunnels. The user is asked to define which one should be used, and sets up the physical topology. Neutron is responsible for allocating the resources as needed.

Using VLANs, for example, the user is required to allocate a VLAN range and hand it off to Neutron. The user also sets up the nodes so that all of them have an interface connected to a VLAN trunk port on a switch. The trunk port needs to be configured to the same VLAN range allocated to Neutron.

Once such configuration is done whenever a user 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 which was just created.

A 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 with 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 Multi Tenancy

Allowing multiple users to share the same physical environment with complete separation between them is a key feature in 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 Oracle VM components.

What is Oracle VM Server?

Oracle VM Server is based on the Xen hypervisor. Oracle VM Server can be managed using Oracle VM Manager, or 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 is the control domain for the Xen hypervisor. It 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 user space tools. When launching a virtual machine, Xen connects the DomU to storage and network resources.

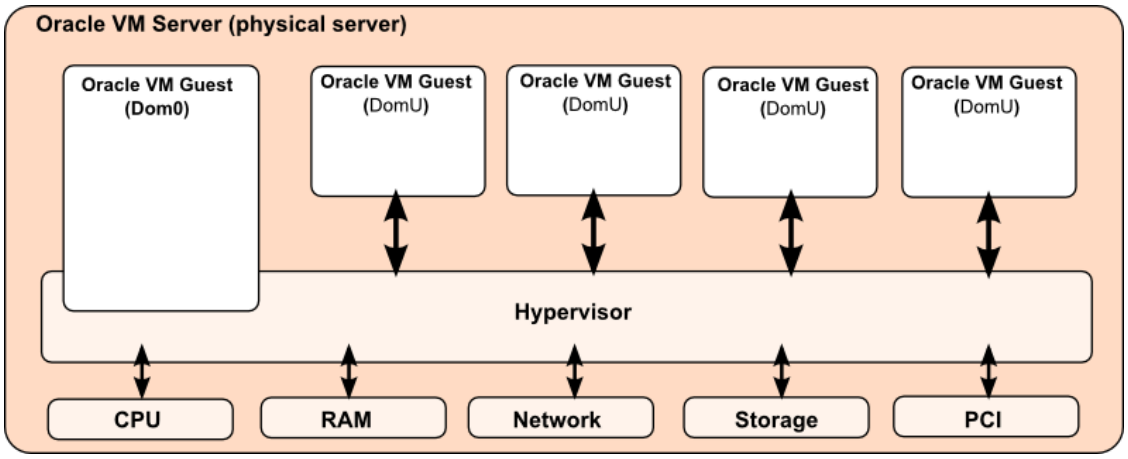

Figure 1. Oracle VM Server deployment on the Xen hypervisor

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 follow the directions about the minimum requirement for it.

Managing Oracle VM with OpenStack

To connect Oracle VM to OpenStack, we currently use the libvirt driver. Xen is fully supported by the libvirt driver, so the integration is very natural. The version of Oracle VM used in the technology preview uses the Unbreakable Enterprise Kernel Release 3 (also known as, *UEK3*), which is a new 3.8 upstream kernel. As a result, Oracle VM uses the latest version of the hardware drivers and Open vSwitch.

Supported Configuration

To integrate Oracle VM with OpenStack, use the latest Oracle VM Release 3.3. For Oracle Linux, simply use the latest Oracle Linux. OpenStack supports various flexible deployment models, where each service can be deployed separately on a different node, or installed together with other services. A user can setup any number of compute and control nodes and test OpenStack in an environment. To lower the complexity, two configurations are supported:

- **All-in-one** – 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.
- **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.

As mentioned, OpenStack is very flexible, and there is no technical limitation that stops the user from experimenting with more sophisticated deployment models. We recommend one of the supported configurations to reduce complexity.

Environment Setup

This section shows you how to install the environment.

Installing a Compute Node

A compute node is simply a system running Oracle Linux or Oracle VM Server Release 3.3. Installation ISOs are available from the following location:

http://edelivery.oracle.com/linux

The installation process is very similar to Linux, and can be done using kickstart, or the interactive installer. For more details on installing Oracle VM Server, see the *Oracle VM Installation and Upgrade Guide for Release 3.3*, available from the link above.

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

The user can check and change the amount of RAM by editing grub.conf in the boot directory:

```
/boot/grub/grub.conf:
kernel /xen.gz . . . . dom0_mem=4096M
```

In the case of the all-in-one deployment, it is recommended to use at least 8GB for Dom0. Dom0's memory comes at the expense of available guest memory, and so the user must make the correct balance.

Local Storage – The default install creates a 50GB root partition, this is good to run a few small VMs. On the control node, Glance also needs space to store the images, so it is recommended to mount another disk, partition or NFS share at /var/lib/glance/images/.

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

NTP – It is recommended that you set the NTP on the servers to point to your default NTP server.

Proxy – If your installation repository is behind a proxy server, make sure you update /etc/yum.conf with this proxy server address. All nodes will access the installation repository and therefore it is important to make sure that yum can access the repositories through the proxy on all the servers.

Installing a Control Node

A *control node* is where most of the OpenStack services are installed. We refer to control node when we discuss nodes which do not run virtual machines. Those nodes can have all the non-compute services, or some of them. In OpenStack, the user 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 compute services, are installed on the same node. All-in-one is often used for a demonstration or development environment, but is not recommended for a production deployment.

In any configuration we choose to use, the control node should only be installed on a Linux system, and not inside a Dom0 of Oracle VM. You can obtain a copy of the latest version of Oracle Linux 6 from:

https://edelivery.oracle.com/linux

Network Set Up

Network configuration tends to be the most sophisticated area in OpenStack. Mistakes in network configuration can lead to complicated problems, and, therefore, it is important to understand how OpenStack networking works. In this technology preview, 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 set up supports 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

All-in-one can only be installed on Oracle Linux, and not on Oracle VM. For an all-in-one deployment, two physical network interface cards are required. The first network interface care 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:

- **eth0** – IP'ed port, connected to the management or public network to allow users to access OpenStack API.
- **eth1** – Non IP'ed, connected to the public network and is used by OpenStack to connect instances to the public network.

Control and Multiple Compute Nodes

When deploying control and compute separately, the control node must be configured as follows (example):

- **eth0** – IP'ed port connected to the public network to allow users to access the OpenStack Dashboard/API and manage the server itself.
- **eth1** – IP'ed port connected to a private management network that connects all the compute nodes and the management node.
- **eth2** – Non-IP'ed port connected to a private VLAN network. This is the port on which the instances, running on the compute nodes, communicate with each other and with the public through the control node.
- **eth3** – Non-IP'ed port connected to the public network. This port is the gateway for the instances to access the public network.

The compute node must have three network cards that must be configured as follows:

- **eth0** – IP'ed port connected to the public network to allow the installer to access public repositories for installation. Also used for users to ssh to the Oracle VM Server and perform monitoring and maintenance operations.

- **eth1** – IP'ed port connected to the private management network that connects all compute and control nodes.
- **eth2** – Non-IP'ed port connected to the private VLAN network. This port is used for virtual machine traffic.

Deploying OpenStack Services

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

Deploying OpenStack using Oracle VM Compute Nodes

For this example, we use the following configuration:

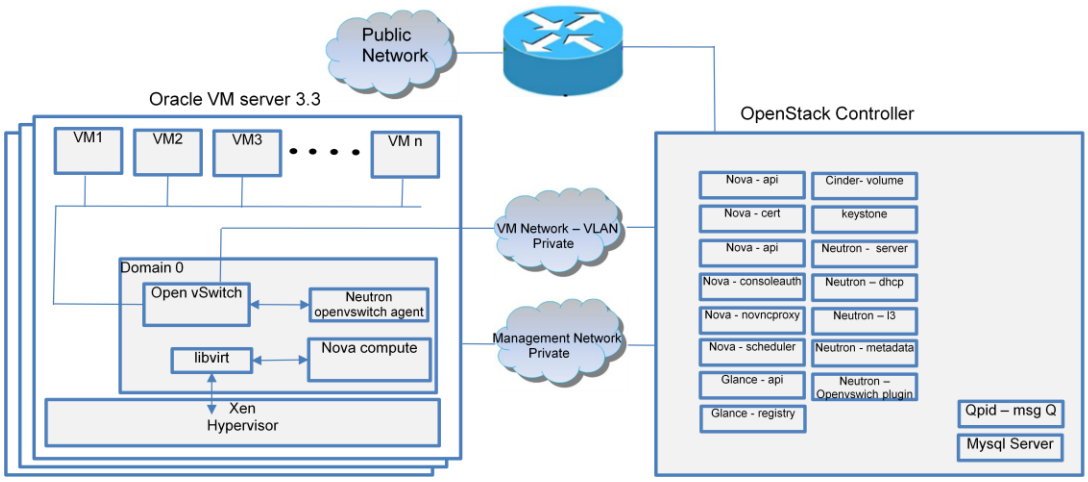

Figure 2. Deployment model to test Oracle VM using OpenStack

In the figure, the deployment model to test Oracle VM is displayed. The deployment includes an OpenStack controller, which has all the services installed, and any number of compute nodes (Oracle VM Servers).

To install the components:

1. Install and configure Oracle VM Server as described above.
    1.1. Make sure the network is configured according to the instructions.
    1.2. Adjust Dom0 memory to at least 4GB or more, and reboot for the change to take effect.
2. Download the **Oracle VM** yum repo files according to the instructions in the Oracle OpenStack channel at:

    http://public-yum.oracle.com/beta/
3. Install the oraclevm-openstack-preinstall rpm available at the repository above by downloading it from the OpenStack repository and installing it manually or simply running:

    ```
    # yum install oraclevm-openstack-preinstall
    ```

    Perform this operation on every compute node. The installation configures the server so that it is ready for the installation process. There is no need to run any script.
4. Install the packstack tool only on the control node:

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

5. Run the packstack command on the control node. The example shows how the command might be run for a three node installation (one control and two compute):

```
# packstack --install-hosts=192.168.0.10,192.168.0.1,192.168.0.2 --use-epel=n --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 packstack parameters used in the example are explained in the table below.

PARAMETER	DESCRIPTION
install-hosts=192.168.0.10,192.168.0.1,192.168.0.	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.
use-epel=n	epel is an external repository that packstack accesses. Since all the packages are in a single location, this is the directive to tell packstack not to access an epel repository.
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 of the control node that serves as a VNC proxy to send the console traffic from the servers to the user.

Table 1. packstack parameters

6. After packstack successfully completes, run the following commands on the compute node:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT use_cow_images false
# openstack-config --set /etc/nova/nova.conf DEFAULT libvirt_type xen
# service openstack-nova-compute restart
Stopping openstack-nova-compute:                           [  OK  ]
Starting openstack-nova-compute:                           [  OK  ]
```

For the **Icehouse** release we need to add two more configuration parameters:

```
# openstack-config --set /etc/nova/nova.conf libvirt cpu_mode none
# openstack-config --set /etc/nova/nova.conf libvirt virt_type xen
# service openstack-nova-compute restart
Stopping openstack-nova-compute:                           [  OK  ]
Starting openstack-nova-compute:                           [  OK  ]
```

All done!

Example of a Packstack Run

The following example output shows the complete packstack install described in the previous sections. 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 --use-epel=n --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
```

Deploying OpenStack using Oracle Linux Compute Nodes

To install the components:

1. Install and configure Oracle Linux as described in the section titled *Environment Setup*. Make sure to disable selinux:

```
# setenforce 0
Also set SELINUX=disabled in /etc/selinux/config
```

2. Download the **Oracle Linux** yum repo files according to the instructions in the Oracle OpenStack channel at:

   http://public-yum.oracle.com/beta/

3. Install the packstack tool only on the control node:

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

4. Run the packstack command on the control node. The example shows how the command might be run for a three node installation (one control and two compute):

```
# packstack --install-hosts=192.168.0.10,192.168.0.1,192.168.0.2 --use-epel=n --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 packstack parameters used in the example are explained in the table in the section titled *Deploying OpenStack using Oracle VM Compute Nodes*.

5. After packstack successfully completes, run the following commands on the compute node:

```
# openstack-config --set /etc/nova/nova.conf DEFAULT libvirt_type kvm
# service openstack-nova-compute restart
Stopping openstack-nova-compute:                           [  OK  ]
Starting openstack-nova-compute:                           [  OK  ]
```

For the **Icehouse** release we need to add two more configuration parameters:

```
# openstack-config --set /etc/nova/nova.conf libvirt cpu_mode none
# openstack-config --set /etc/nova/nova.conf libvirt virt_type kvm
# reboot
```

All done!

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

Post Installation

After the install process completes, the script is run, and the servers are rebooted, the deployment is ready. There are several things you needs to know about the installation:

- **Dashboard** – The OpenStack dashboard is available at:

  http://<public IP 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 the user to use command line without the need to deal with password for every command. Also the password here is the password to be used when logging into the OpenStack dashboard.

- **Logging** – The logs for the OpenStack services are located at /var/log/<service>/. By default, the deployment is set to debug mode. Debug mode generates a lot of debug information in the log file which may make the logs difficult to read. To disable the debug mode, change the `debug` parameter in /etc/nova/nova.conf to `false`, and restart the nova compute service using the following command on all compute nodes:

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

- **Connect to external network –** to connect instances to the public network we will need to connect a network interface from the br-ex bridge on the Open vSwitch to an Ethernet port which is connected to the public network, perform the following command on the control node:

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

- **Enable metadata service** (optional) – OpenStack provides a metadata service which 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, we use Oracle VM Server as a compute node.

Creating the First Instance

This section shows the steps to create the first instance.

1. **Import an image to 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 we use a virtual disk from an Oracle VM environment created by Oracle VM Manager, called ol6_pvm.img. This is an Oracle VM guest installed with Oracle Linux 6. It is also recommended to make sure this image is configured for DHCP networking so it can consume the IP assigned automatically by Neutron.

   To upload the image, use the following command:

```
~(keystone_admin)# glance image-create --name ol6 --disk-format=raw --container-format=bare <
ol6-pvm.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 7c937ba332a45e91ffc29391b1827b28     |
| container_format | bare                                 |
| created at       | 2014-05-05T03:51:44                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | raw                                  |
| id               | 9cdc3491-8885-4702-816d-23364eceb634 |
| is_public        | False                                |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | ol6                                  |
| owner            | ceb1d97df33642f78e8678149ce32903     |
| protected        | False                                |
| size             | 3221225472                           |
| status           | active                               |
| updated_at       | 2014-05-05T03:54:29                  |
+------------------+--------------------------------------+
```

   Remember, you need to first source keystonerc_admin.

2. **Create a network** – 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:

```
# 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. **Launching the instance** – This can be done from either the command line, or the Dashboard. An example from the command line:

```
# nova boot --image ol6 --flavor 1 ol6 --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                                              |
|
+--------------------------------------+------+------------------------------------------------------+
| bf18cef3-74df-434e-8497-b06a8ec6a026 | net1 | 4e4e99a2-aeaa-4a74-8f56-387204de9bc5
10.10.10.0/24 |
+--------------------------------------+------+------------------------------------------------------+
```

The **flavor** parameter, used in the above command, 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 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.tiny   | 512       | 10   | 0         |      | 1     | 1.0         | True      |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | 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
|+-------------------------------------+-----+------+----------+-------------+-----------------+
| 144d4759-7d9c-4a78-8337-fdb33707640  | ol6 |BUILD |spawning  | NOSTATE     | net1=10.10.10.3
|+-------------------------------------+-----+------+----------+-------------+-----------------+
```

From the Dashboard, the instance build can be monitored from the **Instances** tab:

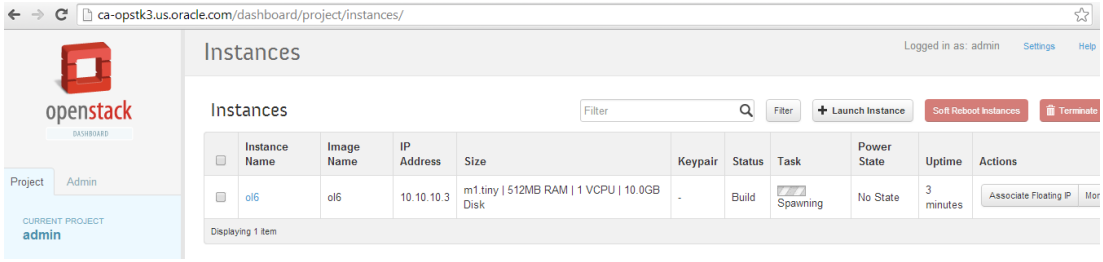

Figure 3. OpenStack Instances view

Or from the **Network Topology** view:

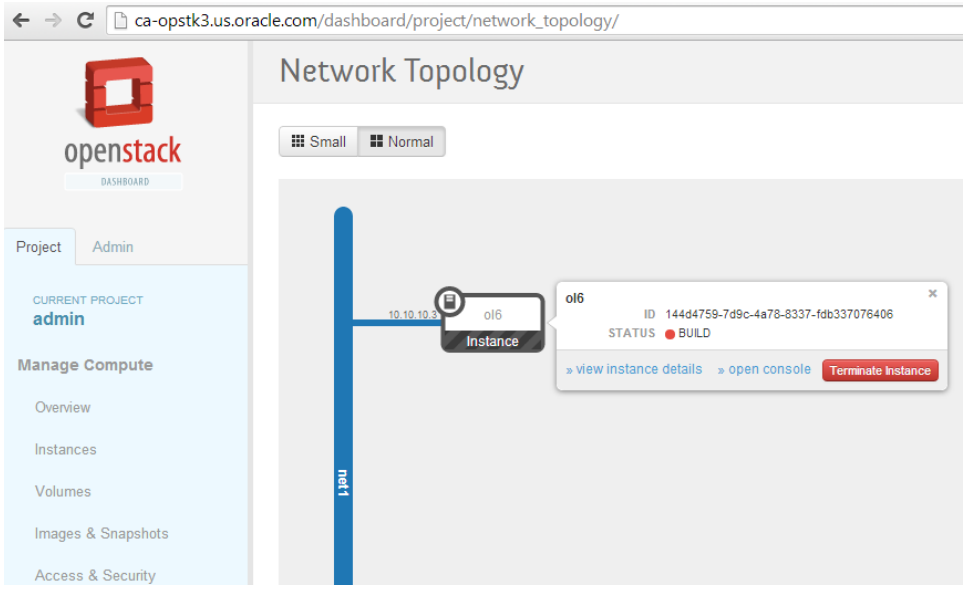

Figure 4. OpenStack Network Topology view

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

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

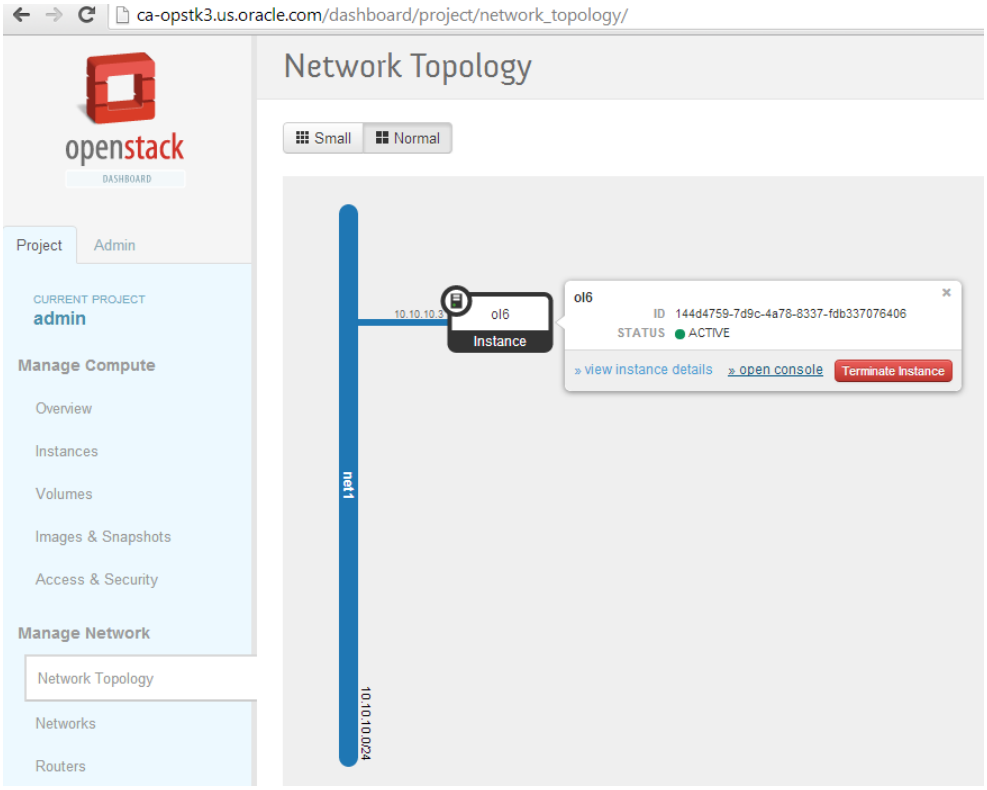

Figure 5. Opening a Console from the OpenStack Network Topology view

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

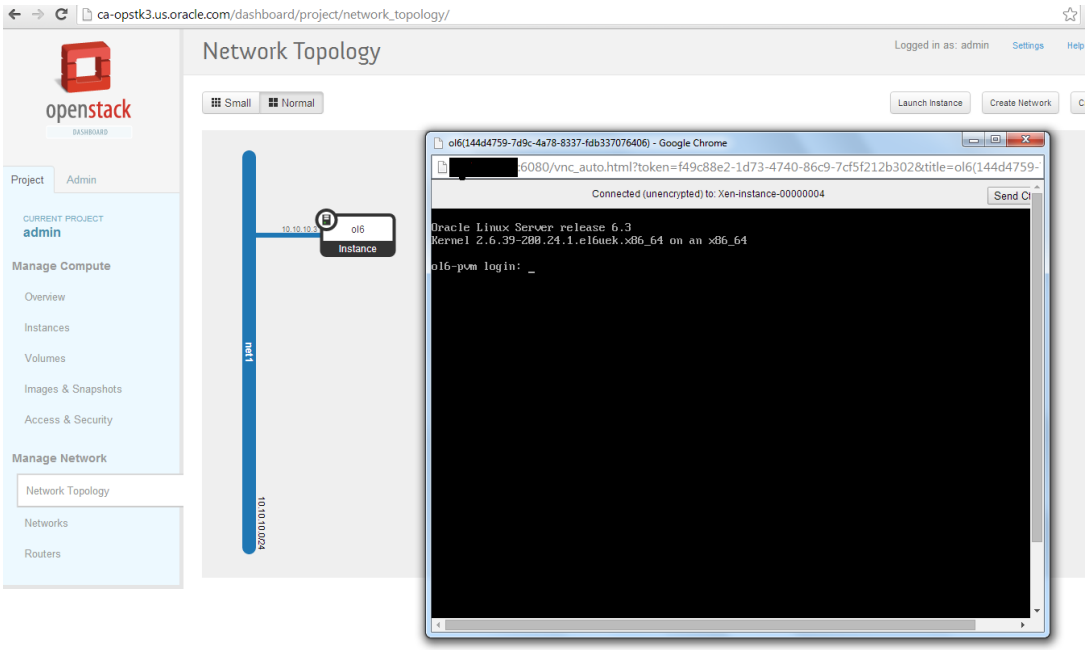

Figure 6. Console Displayed from the OpenStack Network Topology view

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.

OpenStack allows users to add software-defined routers to route 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 the previous section, a new virtual machine is created on a separate network called net2. It is given a different subnet of 20.20.20.0/24. The **Network Topology** tab should look like this:

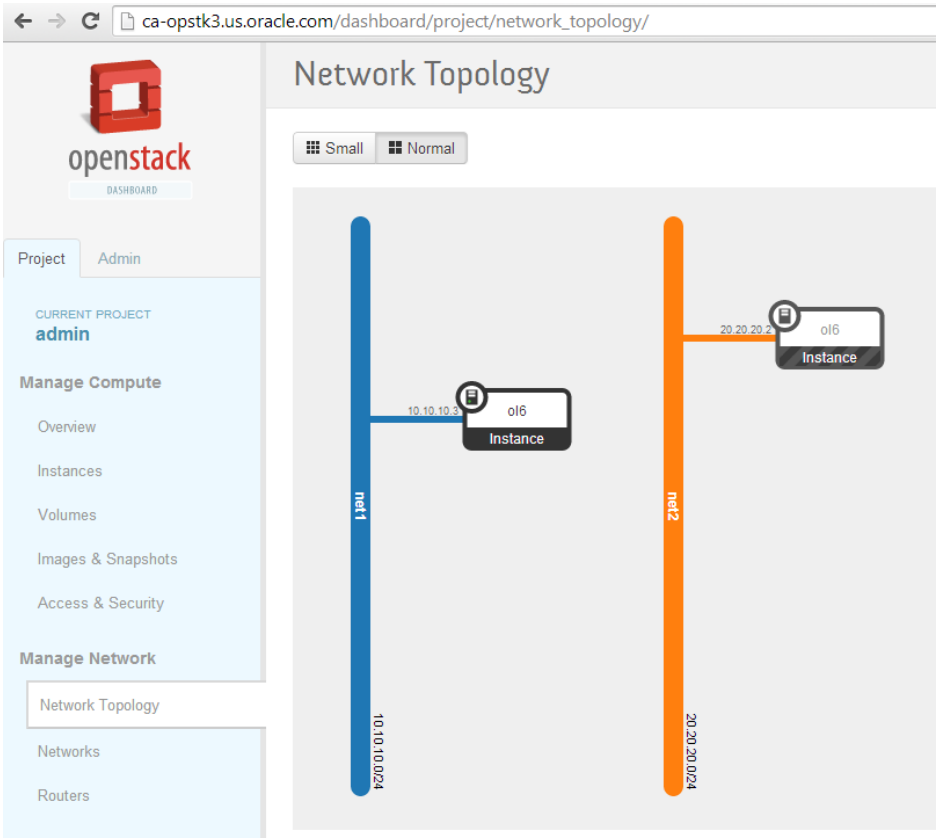

Figure 7. OpenStack Network Topology view

2. Now create a router using the router button on the upper right hand side of the screen. After creating a router, go to the **Routers** tab and add interfaces from both subnets. The result looks like this:

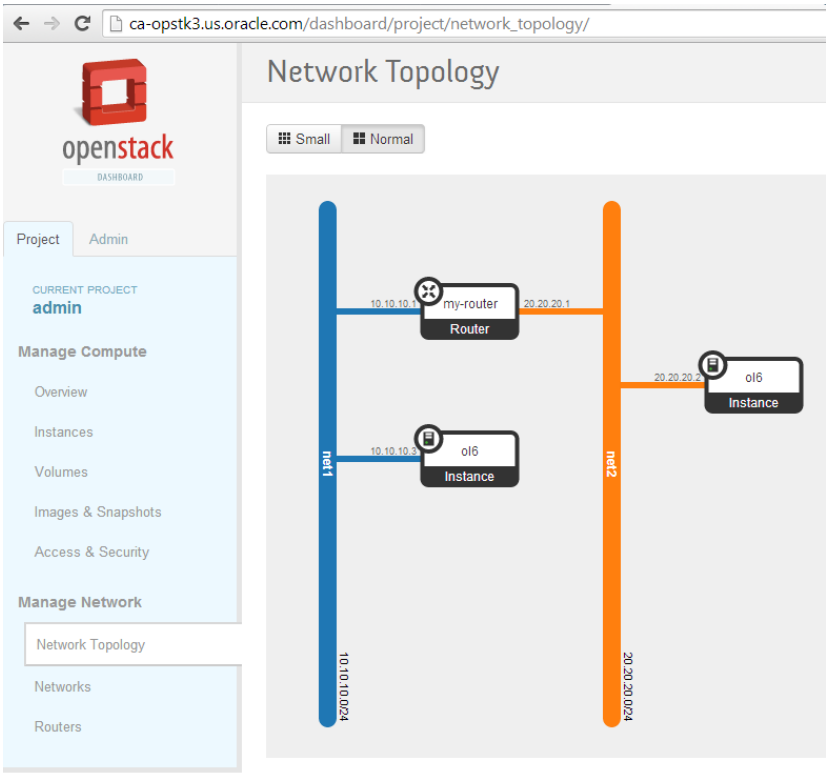

Figure 8. OpenStack Network Topology view

The above figure shows the router with two interfaces. Near the router you can see the gateway IP, which was 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 ping and ssh from one instance to the other.

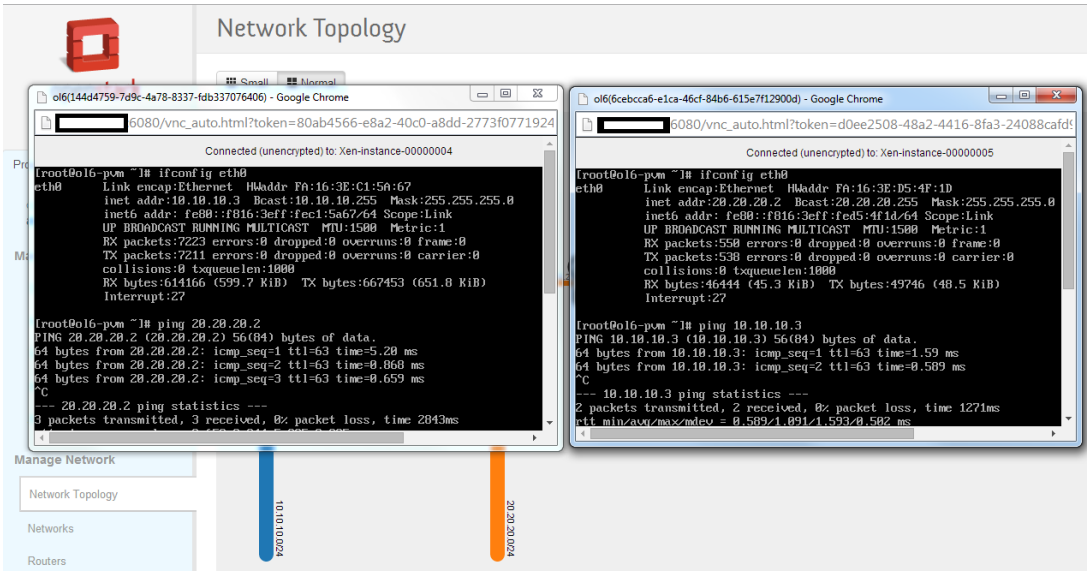

Figure 9. Consoles communicating with each other

4. Next, create a floating IP and associate it with an instance. This allows the user to connect to the instance from outside the OpenStack deployment. 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                        | 4296fb7f-64f0-4297-ba9d-2f1fa45551ef |
| name                      | public                               |
| provider:network_type     | vlan                                 |
| provider:physical_network | default                              |
| provider:segmentation_id  | 1002                                 |
| router:external           | True                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   |                                      |
| tenant_id                 | ceb1d97df33642f78e8678149ce32903     |
+---------------------------+--------------------------------------+
```

5. Next, create a subnet, which has an IP addresses on the public network. To do so, use the following command:

```
# 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>
```

Here you allocate a range of IP addresses to use from a public network.

6. 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 in the range to use as a gateway for the router:

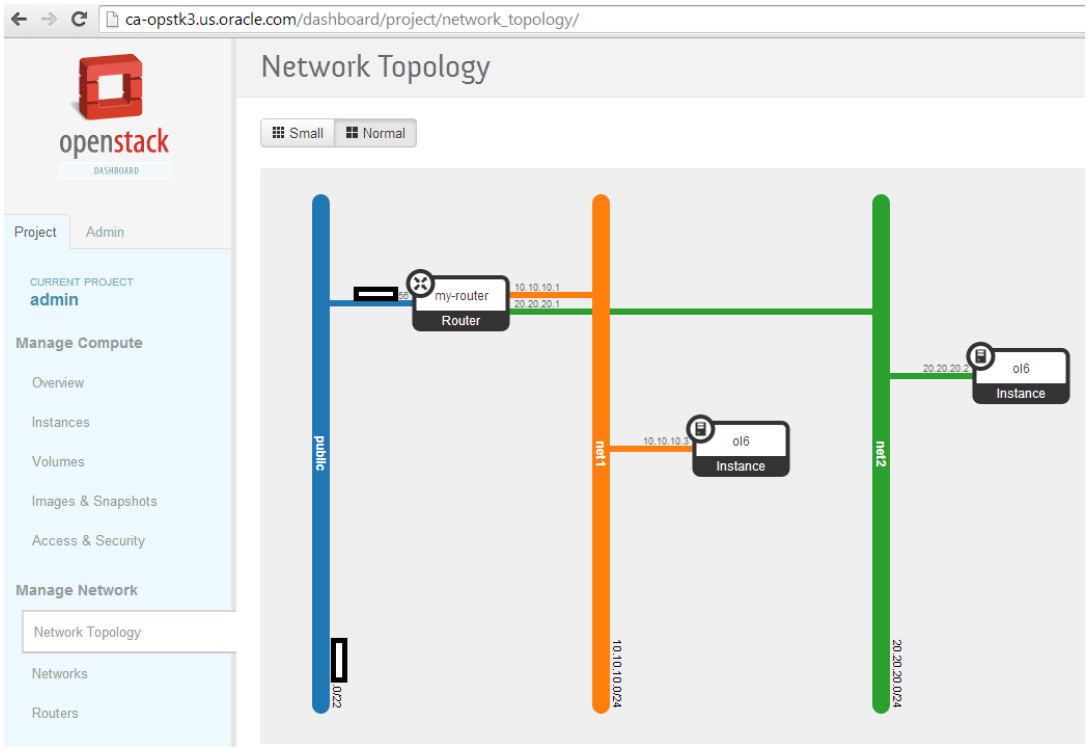

Figure 10. Gateway configured

7. 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     |
+---------------------+--------------------------------------+
```

8. From the Dashboard, you can now associate the floating IP with the VM. Go to the **Instances** tab, and choose the instance with which to associate the floating IP. Use the **More** menu to associate a floating IP with the instance. 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. This 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, and can be used for any NFS storage. Since this is a generic driver without specific knowledge of the backend storage device, it cannot perform actions which are specific to the array, such as snapshot, clone, and backup.

For this exercise, you need to have some NFS shares which 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. 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 on 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     |      2014-05-05T22:51:19.648992      |
| display_description |                 None                 |
|     display_name    |            my-new-volume             |
|          id         | 759d2736-a21f-493c-ad21-5ae976cf92fe |
|       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 using the **cinder list** command:

```
~(keystone_admin)# cinder list
+--------------------------------------+-----------+---------------+------+-------------+--------
--+-------------+
|                  ID                  |   Status  |  Display Name | Size | Volume Type |
Bootable | Attached to |
+--------------------------------------+-----------+---------------+------+-------------+--------
--+-------------+
| 759d2736-a21f-493c-ad21-5ae976cf92fe | 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 it to the instance. Here you can use **auto** to let Nova choose which device should be used. To obtain a list of instance IDs, first run

```
~(keystone admin)# nova list
+--------------------------------------+------+--------+------------+-------------+--------------
----------------+
| ID                                   | Name | Status | Task State | Power State | Networks
|
+--------------------------------------+------+--------+------------+-------------+--------------
----------------+
| 144d4759-7d9c-4a78-8337-fdb337076406 | ol6  | ACTIVE | None       | Running     |
net1=10.10.10.3              |
| 6cebcca6-e1ca-46cf-84b6-615e7f12900d | ol6  | ACTIVE | None       | Running     |
net2=20.20.20.2, xxxxxx.57 |
+--------------------------------------+------+--------+------------+-------------+--------------
----------------+
```

After selecting the ID of the instance, run the following command to connect the instance:

```
# nova volume-attach 6cebcca6-e1ca-46cf-84b6-615e7f12900d 759d2736-a21f-493c-ad21-5ae976cf92fe
auto
+----------+--------------------------------------+
| Property | Value                                |
+----------+--------------------------------------+
| device   | /dev/xvdb                            |
| serverId | 6cebcca6-e1ca-46cf-84b6-615e7f12900d |
| id       | 759d2736-a21f-493c-ad21-5ae976cf92fe |
| volumeId | 759d2736-a21f-493c-ad21-5ae976cf92fe |
+----------+--------------------------------------+
```

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 which remains persistent after it is terminated. The difference from ephemeral storage is that the boot process does not require copying an image, making it significantly faster than the ephemeral boot. To create a bootable volume, run the following command:

```
# cinder create --display-name my-new-bootable-volume  --image-id 9cdc3491-8885-4702-816d-
23364eceb634 4
```

The command finishes immediately, but 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              |
+--------------------------------------+-----------+-----------------------+------+-------------
+----------+--------------------------------------+
| 759d2736-a21f-493c-ad21-5ae976cf92fe |   in-use  |     my-new-volume     |  5   |     None
|  false   | 6cebcca6-e1ca-46cf-84b6-615e7f12900d |
| 9966ae2a-0135-46d9-b115-cd3a41c56b24 | available | my-new-bootable-volume |  4   |     None
|   true   |                                      |
+--------------------------------------+-----------+-----------------------+------+-------------
+----------+--------------------------------------+
```

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

```
+--------------------------------------+--------+------+
| 4296fb7f-64f0-4297-ba9d-2f1fa45551ef | public | None |
| bf18cef3-74df-434e-8497-b06a8ec6a026 | net1   | None |
| e86928ee-6587-4658-bcaa-30e4e8cc5642 | net2   | None |
+--------------------------------------+--------+------+
```

When you have obtained the network id, run the following command to start the instance:

```
~(keystone_admin)# nova boot --boot-volume 9966ae2a-0135-46d9-b115-cd3a41c56b24 --flavor 1 ol6
--nic net-id=bf18cef3-74df-434e-8497-b06a8ec6a026
+--------------------------------------+------------------------------------------------------+
| Property                             | Value                                                |
+--------------------------------------+------------------------------------------------------+
| OS-EXT-STS:task_state                | scheduling                                           |
| image                                | Attempt to boot from volume - no image supplied      |
| OS-EXT-STS:vm_state                  | building                                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000006                                    |
| OS-SRV-USG:launched_at               | None                                                 |
| flavor                               | m1.tiny                                              |
| id                                   | d852918a-86d0-4bd7-b05f-89cc408b6e56                 |
| security_groups                      | [{u'name': u'default'}]                              |
| user_id                              | eff1d8bc7ce24ab091aff893638a387c                     |
| OS-DCF:diskConfig                    | MANUAL                                               |
| accessIPv4                           |                                                      |
| accessIPv6                           |                                                      |
| progress                             | 0                                                    |
| OS-EXT-STS:power_state               | 0                                                    |
| OS-EXT-AZ:availability_zone          | nova                                                 |
| config_drive                         |                                                      |
| status                               | BUILD                                                |
| updated                              | 2014-05-05T23:28:01Z                                 |
| hostId                               |                                                      |
| OS-EXT-SRV-ATTR:host                 | None                                                 |
| OS-SRV-USG:terminated_at             | None                                                 |
| key_name                             | None                                                 |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                                 |
| name                                 | ol6                                                  |
| adminPass                            | p8JVCBuumXWy                                         |
| tenant_id                            | ceb1d97df33642f78e8678149ce32903                     |
| created                              | 2014-05-05T23:28:00Z                                 |
| os-extended-volumes:volumes_attached | [{u'id': u'9966ae2a-0135-46d9-b115-cd3a41c56b24'}]   |
| metadata                             | {}                                                   |
+--------------------------------------+------------------------------------------------------+
```

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

Summary

This whitepaper took the reader through different options of deploying OpenStack on Oracle VM and Oracle Linux. We started by going through some basic concepts about OpenStack and Oracle VM and describing how to set them up for the OpenStack deployment. We continued by going over the installation steps, and demonstrating how to use several features on the network and storage stacks.

As you will come to recognize, OpenStack has a rich feature set and various different services. After going through this whitepaper, the reader should have a good starting point to learn more about the different features in OpenStack, and experiment with them on the environment we have created. OpenStack will continue to evolve, adding new services, 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 may have seen, networking seems to be the most challenging and sophisticated area in OpenStack. We recommend spending time learning more about the network stack, and understanding how networking is structured.

We hope you found this whitepaper useful, and wish you successful deployment.

Getting Started with Oracle VM, Oracle Linux and OpenStack: Technology Preview
May 2014
Author: Ronen Kofman

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Oracle is committed to developing practices and products that help protect the environment

Hardware and Software, Engineered to Work Together