OpenStack Havana

Joseph Callen

December 12, 2013

Acknowledgements

I have read many articles from various sources I am sure I have missed to source them some along the way.

Contents

List of Figures

List of Tables

Listings

Executive Summary

OpenStack can be difficult software package to install and configure for a novice. I have created this document to guide someone through the install that I performed in the lab. Hopefully with a "real world" example it should be easier to understand the various modules and concepts required for an OpenStack deployment. This document was created with LaTeX.

Chapter 1

Introduction

1.1 Physical Hosts

In order to test the installation and operational aspects of a true OpenStack environment I needed additional nodes. Utilizing old laptop hardware I was able to create four additional virtual machines to run OpenStack services on. The compute node remains physical. I am still using the RDO Havana community edition on Fedora 19 x64.

Table 1.1: Physical Compute

Hardware		**Dev**	**Model**	**Network**	**Usage**
AMD Athlon Dual Core	1	eth0	r8169	Managment	Nova, Glance &
8 GB RAM, 2 TB HD	2	eth1	e1000	Instance uplink	Cinder
Levono T61	1	eth0	e1000e	Management &	oVirt Node
Intel®Core2^{TM}Duo CPU T7100, 4 GB RAM				Instance Uplink	
Dell Latitude D620	1	eth0	tg3	Management &	oVirt Node
Intel®Core2^{TM}Duo CPU T5500, 4 GB RAM				Instance Uplink	
Dell PE2850	1	em1	e1000	Management	NFS
Intel®Xeon^{TM}, 2 GB RAM, 3x146GB 10K				Instance Uplink	

1.2 oVirt

oVirt is a management and host node virtualization package similar to vSphere. It uses Linux KVM as the hypervisor with either Fedora or CentOS as the base OS. The management console is web based.

1.2.1 oVirt Engine Install

Before installation DNS needs to be configured, including an A record of the oVirt engine and nodes.

Listing 1.1: Install oVirt repo and package

```
yum localinstall http://ovirt.org/releases/ovirt-release-fedora.noarch.
    rpm
yum install -y ovirt-engine
engine-setup
```

1.2.2 PXE

To install an oVirt node the easiest method I have found is use PXE. First lets convert the ISO to tftpboot. **Note:** This is an older release of ovirt-node, the current version has a bug with installation.[1]

Listing 1.2: Download and convert oVirt Node

```
wget http://resources.ovirt.org/releases/3.3/iso/ovirt-node-iso
    -3.0.1-1.0.2.vdsm.fc19.iso
livecd-iso-to-pxeboot ovirt-node-iso-3.0.1-1.0.2.vdsm.fc19.iso
```

Listing 1.3: Copy tftproot to root, restore SELinux context, change permissions

```
cp -R /tftpboot/ /
restorecon -Rv /tftpboot/
chmod -R 755 /tftpboot/
```

Listing 1.4: Enable TFTP

```
sed -i 's/disable.*=\ yes/disable\t\t\t= no/' /etc/xinetd.d/tftp
# or
vi /etc/xinetd.d/tftp
service tftp
{
        ...
        disable = no #change from yes to no
        ...
}
systemctl restart xinetd
```

[1] Red Hat Bugzilla Bug #1032228

1.2.3 DHCP

The DHCP server needs to be configured to provide addresses and PXE related configuration elements.

Listing 1.5: Install ISC DHCP server

```
yum install dhcp -y
```

Below is a partial example of my DHCPd configuration file. The most important section is "ovirt-server1". Options filename and next-server must be listed for PXE boot to function correctly. The option filename will not change and should be listed as "pxelinux.0".

Listing 1.6: Example DHCPd configuration

```
authoritative;
ddns-update-style none;

ddns-domainname         "virtomation.com";
option domain-name      "virtomation.com";
option domain-name-servers      10.53.252.123, 10.53.252.246;
option ntp-servers              10.53.252.123;
default-lease-time      86400;
option ip-forwarding    off;
...
subnet 10.53.253.0 netmask 255.255.255.0 {
        range 10.53.253.130 10.53.253.140;
        option routers 10.53.253.1;
        option subnet-mask      255.255.255.0;
}

# HOST - RHEV
host ovirt-server1  {
        hardware ethernet 00:15:C5:59:AE:36; # MAC from machine that
            will boot via PXE
        fixed-address 10.53.253.50; # Fixed IP address
        filename "pxelinux.0";
        next-server 10.53.252.50;  # DHCP Server
        option host-name "virkvmpaw001.virtomation.com";
}
...
```

After the configuration has been saved remember to restart dhcpd.

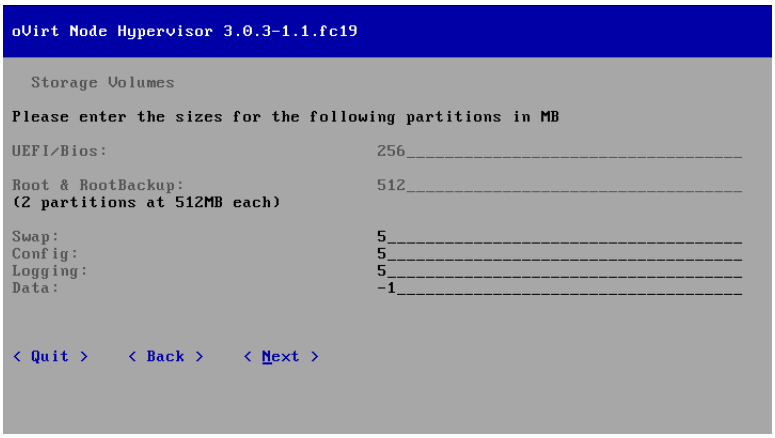
Listing 1.7: Restart DHCPd

```
systemctl reload-or-try-restart dhcpd
```

1.2.4 oVirt Node Install

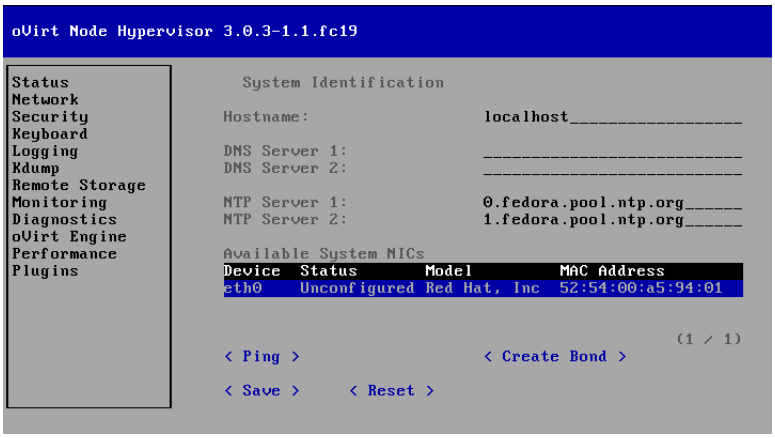
The installation of the oVirt node is generally as easy as ESXi. I am only displaying figures that need extra explanation. Figure 1.1 shows the system volumes and sizes. If you are using a USB flash drive set the log and swap to 5 MB.

Figure 1.1: System Volumes



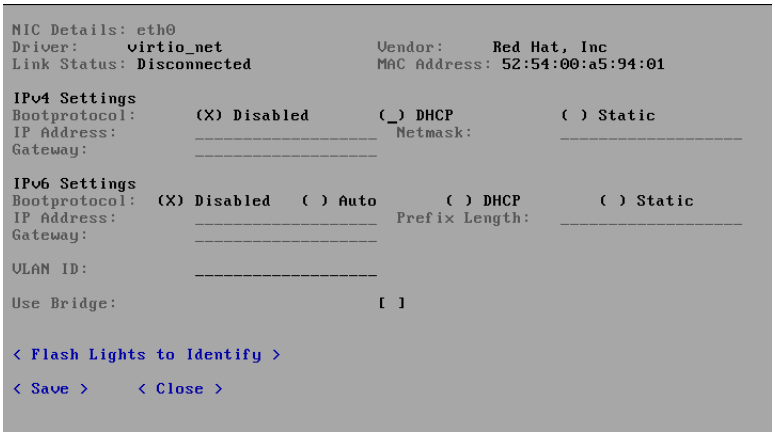
The interfaces are not automatically configured. Figure 1.2 shows the available interfaces, select an interface and hit enter. In the next screen listed in Figure 1.3 configure the IP settings of the selected interface.

Figure 1.2: Network Configuration



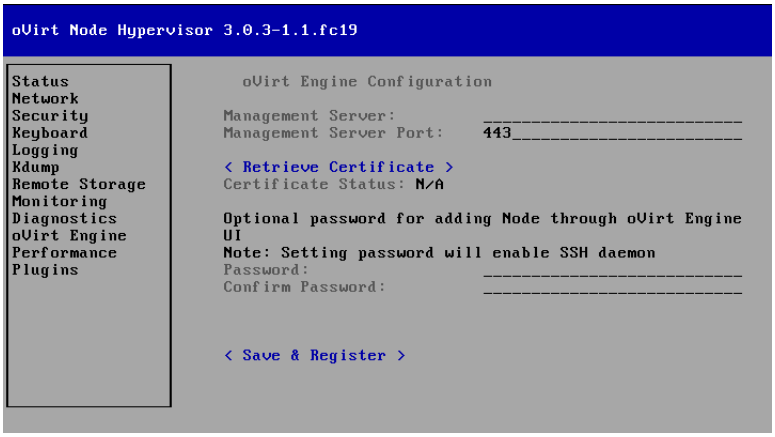
Finally once the oVirt is available on the network you can configure it with the oVirt engine. As displayed in Figure 1.4 configure the management server using either FQDN or IP and retrieve the certificate. You can also set the optional password, which I recommend

Figure 1.3: IP Configuration



for a lab environment since it will also start ssh and set the root password.

Figure 1.4: oVirt Engine Configuration



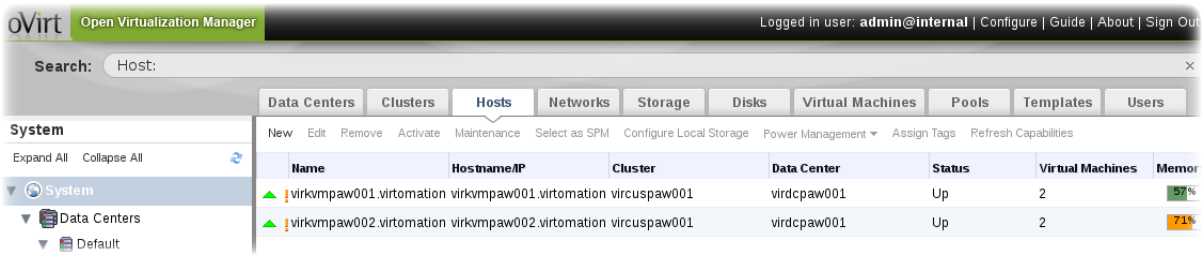
Note: In my testing SELinux needed to be disabled before for any virtual machines would start. To disable SELinux use the command listed below. Unfortunately I haven't found a way to persist the changes after a reboot. I would guess that re-mounting the filesystem read-write and changing setting in /etc/selinux/config file would work but I have yet to try it.

Listing 1.8: Disable SELinux

```
1 setenforce 0
```

Below is a screenshot of the oVirt Administration UI with the available hosts listed.

Figure 1.5: Hosts in oVirt Engine



Chapter 2

All-In-One install with Neutron using VLANs

2.1 Physical Networking

I have three VLAN 98 — 100 configured on a dot1q trunk to interface p9p1 on a Fedora 19 host. VLAN 100 has a gateway IP configured (10.53.100.1) on the Cisco Catalyst 4006. VLAN 98 and 99 are non-routed vlans, at least as configured on the catalyst, so they have no gateway IP addresses configured, that will be provided by the neutron router.

Listing 2.1: VLAN Configuration

```
interface Vlan98
 no ip address
end
interface Vlan99
 no ip address
end
interface Vlan100
 ip address 10.53.100.1 255.255.255.0
 ip helper-address 10.53.252.50
end
```

Listing 2.2: Instance uplink interface; gi2/5; #1

```
interface GigabitEthernet2/5
 switchport trunk encapsulation dot1q
 switchport trunk allowed vlan 2-4,12,98-100,250-253
 switchport mode trunk
end
```

Listing 2.3: Management interface; gi3/13; #2

Figure 2.1: Compute Node

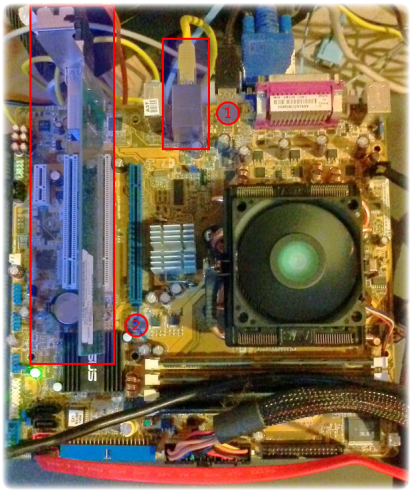
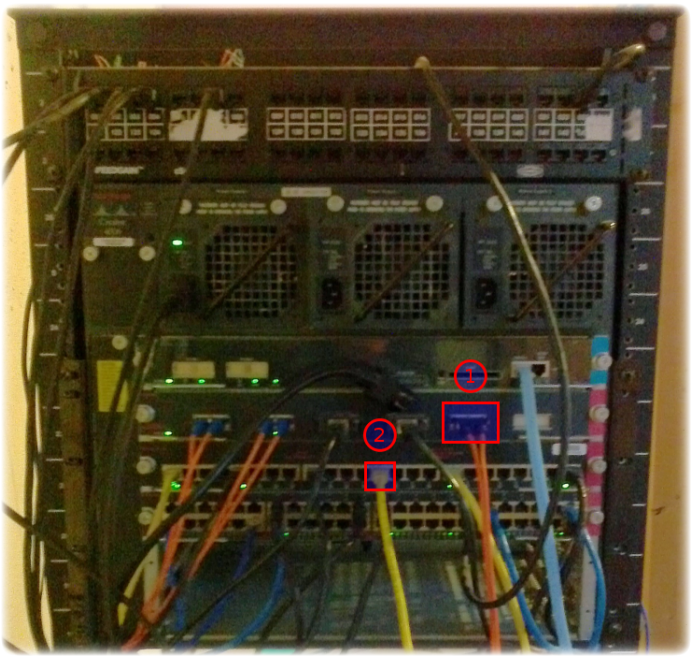


Figure 2.2: Cisco Catayst 4006



```
WS-C4006-SUPV#sh run int gi3/13
Building configuration...

Current configuration : 65 bytes
!
interface GigabitEthernet3/13
 switchport access vlan 253
end
```

2.2 Installation

2.2.1 Packstack

Listing 2.4: Configure Packstack

```
DATE=`date +"%Y_%m_%d_%H_%M_%S"`
packstack --gen-answer-file=packstack-${DATE}
vi packstack-${DATE}
```

Listing 2.5: Answer file changes

```
CONFIG_NEUTRON_INSTALL=y
CONFIG_CINDER_VOLUMES_CREATE=n
CONFIG_NEUTRON_L3_EXT_BRIDGE=provider
CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=vlan
CONFIG_NEUTRON_OVS_VLAN_RANGES=inter-vlan:98:99
CONFIG_NEUTRON_OVS_BRIDGE_MAPPINGS=inter-vlan:br-inst
CONFIG_NEUTRON_OVS_BRIDGE_IFACES=br-inst:p9p1
```

Listing 2.6: Execute Packstack

```
packstack --answer-file=packstack-${DATE}
```

Note: Monitor the progress in detail via /var/tmp/packstack/

2.3 Neutron Networking

2.3.1 External

Now we will configure VLAN 100 to be our external network. First we need to source the keystone file created by packstack.

Listing 2.7: Create network

```
neutron net-create ext_vlan100 --provider:network_type vlan --provider:
    physical_network inter-vlan --provider:segmentation_id 100 --router:
    external=True
```

Create a subnet for ext_vlan100, this will be used for floating addresses.

Listing 2.8: Create subnet

```
neutron subnet-create ext_vlan100 --gateway 10.53.100.1 --name
    ext_vlan100_subnet 10.53.100.0/24 --allocation-pool start
    =10.53.100.10,end=10.53.100.253 --enable_dhcp=False
```

Listing 2.9: Create router

```
neutron router-create ext_vlan100_router
```

Listing 2.10: Set gateway

```
neutron router-gateway-set ext_vlan100_router ext_vlan100
```

2.3.2 Tenant

Listing 2.11: Create network

```
TENANT_ID=$(keystone tenant-list|awk '/admin/ {print $2}')
neutron net-create --tenant-id ${TENANT_ID} net01
```

Listing 2.12: Create subnet

```
neutron subnet-create net01 --name net01_subnet 192.168.98.0/24
```

Listing 2.13: Add interface

```
neutron router-interface-add ext_vlan100_router net01_subnet
```

The ext_vlan_router than becomes the default gateway for the vlan with an IP of 192.168.98.1. And the result as displayed in Horizon.

2.3.3 Troubleshooting

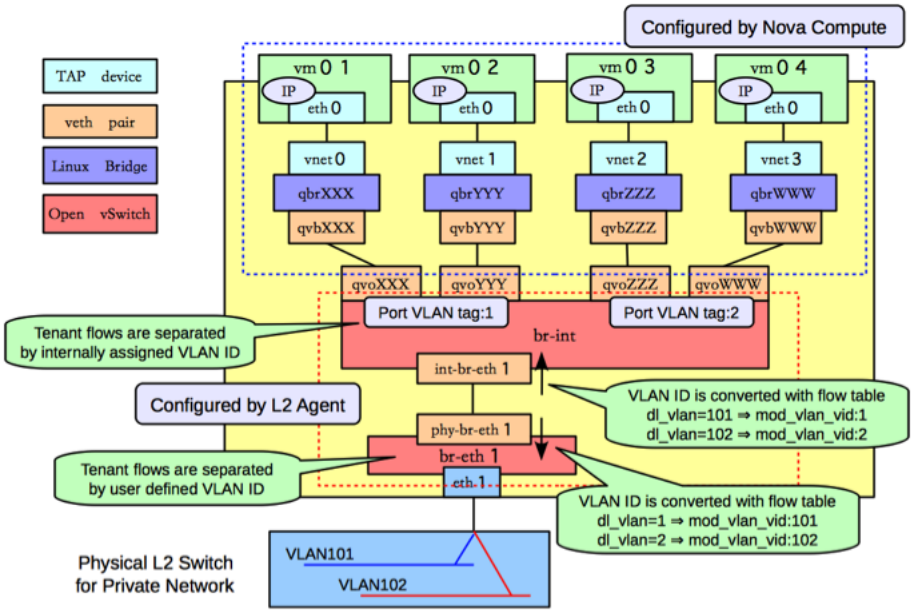
I believe OpenStack networking with Neutron is the most difficult part of the stack to understand. The problem being just to have isolated tenant networks requires linux bridge, open vswitch bridge, and virtual Ethernet devices. I have created a python script to better understand the relationship between a virtual machine and those devices.

The table below shows the interfaces between the vm and int-br-eth (as shown in Figure 2.3), where a majority of the devices are configured.

Listing 2.14: map_router.py script

```
#Sources:
# http://effbot.org/zone/element-xpath.htm
```

Figure 2.3: VLAN Scenario




```
3  # http://eli.thegreenplace.net/2012/03/15/processing-xml-in-python-with
       -elementtree/
4  # http://wiki.libvirt.org/page/SSHSetup
5  # http://docs.python.org/2/tutorial/datastructures.html#dictionaries
6  # http://libvirt.org/git/?p=libvirt.git;a=blob;f=tools/virsh-domain-
       monitor.c
7  # http://www.linuxproblem.org/art_9.html
8  # http://j2labs.tumblr.com/post/4477180133/ssh-with-pythons-paramiko
9
10 __author__ = 'jcallen'
11 import os
12 import libvirt
13 import sys
14 import prettytable
15 import novaclient.v1_1.client as nvclient
16 from neutronclient.v2_0 import client
17 import keystoneclient.v2_0.client as ksclient
18 import xml.etree.ElementTree as ET
19 import paramiko
20 import re
21 import argparse
22
23
24 def keystone_connect(hostname, username, password, tenant):
25     auth_url = "http://%s:35357/v2.0" % hostname
```

```
26
27      keystone = ksclient.Client(auth_url=auth_url,
28                                 username=username,
29                                 password=password,
30                                 tenant_name=tenant)
31      if keystone is None:
32          print 'Failed to open connection to OpenStack instance'
33          sys.exit(1)
34
35      return keystone
36
37
38  def neutron_connect(hostname, keystone):
39      os_url = "http://%s:9696/" % hostname
40      neutron_conn = client.Client(endpoint_url=os_url, token=keystone.
            auth_token)
41      if neutron_conn is None:
42          print 'Failed to open connection to OpenStack instance'
43          sys.exit(1)
44
45      return neutron_conn
46
47
48  def nova_connect(hostname, username, password, tenant):
49      authurl = "http://%s:5000/v2.0" % hostname
50      nova_conn = nvclient.Client(username, password, tenant, authurl,
            service_type="compute")
51      if nova_conn is None:
52          print 'Failed to open connection to OpenStack instance'
53          sys.exit(1)
54
55      return nova_conn
56
57
58  def qemu_connect(hostname, username):
59      qemu = "qemu+ssh://%s@%s/system" % (username, hostname)
60      qemu_conn = libvirt.openReadOnly(qemu)
61      if qemu_conn is None:
62          print 'Failed to open connection libvirtd'
63          sys.exit(1)
64
65      return qemu_conn
66
67
68  def ssh_connect(host, username, private_key, port=22):
69      """Helper function to initiate an ssh connection to a host."""
70      transport = paramiko.Transport((host, port))
71
72      if os.path.exists(private_key):
73          rsa_key = paramiko.RSAKey.from_private_key_file(private_key)
```

```
74          transport.connect(username=username, pkey=rsa_key)
75      else:
76          raise TypeError("Incorrect private key path")
77
78      return transport
79
80
81  def exec_cmd(transport, command):
82      """Executes a command on the same server as the provided
83      transport
84      """
85      try:
86          channel = transport.open_session()
87          channel.exec_command(command)
88          if channel.recv_exit_status() == 0:
89              output = channel.makefile('rb', -1).readlines()
90              return output
91          else:
92              stderr_output = channel.makefile_stderr('rb', -1).readlines
                    ()
93              print stderr_output
94              print "error: " + channel.recv_exit_status()
95              return ""
96      except:
97          print sys.exc_info()
98
99
100 def show_brctl_veth(device, transport):
101     BRCTL = "/usr/sbin/brctl"
102     command = BRCTL + " show " + device
103     output = exec_cmd(transport, command)
104     if output != "":
105         ifline = output[1]
106         match = re.findall("(qvb[\w-]+$)", ifline)
107         veth = match[0]
108         return veth
109
110
111 def ethtool_adapter_stats(device, transport):
112     ETHTOOL = "/usr/sbin/ethtool"
113     command = ETHTOOL + " -S " + device
114     output = exec_cmd(transport, command)
115     if output != "":
116         ifline = output[1]
117         match = re.findall("peer_ifindex: ([\d]+)$", ifline)
118         peer_ifindex = match[0]
119         return peer_ifindex
120
121
122 def ip_link(peer_ifindex, transport):
```

```
123     ETHTOOL = "/usr/sbin/ip"
124     command = ETHTOOL + " link "
125     output = exec_cmd(transport, command)
126     if output != "":
127         for interfaces in output:
128             ifline = interfaces
129             matchstring = "^%s:\s([\w-]+)" % peer_ifindex
130             match = re.findall(matchstring, ifline)
131             if len(match) != 0:
132                 veth_pair2 = match[0]
133                 return veth_pair2
134 
135 
136 def ovs_ofctl(action, device, transport):
137     """
138     Executes ovs-ofctl via paramiko ssh client
139 
140     Requires a sudoers entry:
141     Defaults !requiretty
142     user ALL = NOPASSWD: /usr/bin/ovs-ofctl
143     """
144     OFCTL="sudo /usr/bin/ovs-ofctl"
145     command = OFCTL + " " + action + " " + device
146     output = exec_cmd(transport, command)
147     return output
148 
149 
150 def ovs_ofctl_dump_flows(device, tag, transport):
151     output = ovs_ofctl("dump-flows", device, transport)
152     if output != "":
153         for line in output:
154             matchstring = "mod_vlan_vid:%s" % tag
155             if re.search(matchstring, line) is not None:
156                 matchstring = "dl_vlan=(\d+)"
157                 match = re.findall(matchstring, line)
158                 if len(match) != 0:
159                     #print match
160                     vlan = match[0]
161                     return vlan
162 
163 
164 def ovs_vsctl(action, device, transport):
165     """
166     Executes ovs-vsctl via paramiko ssh client
167 
168     Requires a sudoers entry:
169     Defaults !requiretty
170     user ALL = NOPASSWD: /usr/bin/ovs-vsctl
171     """
172 
```

```
173     VSCTL="sudo /usr/bin/ovs-vsctl"
174     command = VSCTL + " " + action + " " + device
175     output = exec_cmd(transport, command)
176     return output
177
178
179 def ovs_vsctl_list_port(device, transport):
180     output = ovs_vsctl("list port", device, transport)
181     if output != "":
182         for line in output:
183             matchstring = "^tag[\s:]*(\d+)"
184             match = re.findall(matchstring, line)
185             if len(match) != 0:
186                 tag = match[0]
187                 return tag
188
189
190 def ovs_vsctl_port_to_br(device, transport):
191     output = ovs_vsctl("port-to-br", device, transport)
192     bridge = output[0].split("\n")[0]
193     return bridge
194
195
196 def get_router_id(neutron_conn, net_id):
197     """
198     Yes, I know there is a bug here, will resolve later!
199     """
200     net = neutron_conn.show_network(net_id)
201
202     for subnet in net['network']['subnets']:
203         ports = neutron_conn.list_ports(retrieve_all=True, subnet_id=
                subnet)
204         for port in ports['ports']:
205             if port['device_owner'] == 'network:router_interface':
206                 return port['device_id']
207
208
209 def domiflist(domxml):
210     tree = ET.ElementTree(ET.fromstring(domxml))
211     elements = tree.iterfind("./devices/interface")
212
213     for ele in elements:
214         interface_type = ele.attrib['type']
215         if interface_type == "bridge":
216             bridge = ele.find("source[@bridge]").attrib["bridge"]
217             tap = ele.find("target[@dev]").attrib["dev"]
218             mac = ele.find("mac[@address]").attrib["address"]
219             return bridge, tap, mac
220         else:
221             print "Error: Currently will only work with bridged
```

```
                interfaces."
222
223
224 def args():
225     parser = argparse.ArgumentParser()
226     parser.add_argument('--hostname', help='Hostname of OpenStack
            instance', required=True)
227     parser.add_argument('--os-username', help='OpenStack Username',
            required=True)
228     parser.add_argument('--os-password', help='OpenStack Password',
            required=True)
229     parser.add_argument('--os-tenant-name', help='OpenStack Tenant',
            required=True)
230     parser.add_argument('--username', help='Username of OpenStack host'
            , required=True)
231     parser.add_argument('--priv-key', help='Location of private SSH Key
            ', required=True)
232     arguments = parser.parse_args()
233     return arguments
234
235
236 def main():
237
238     args_namespace = args()
239
240     nova_conn = nova_connect(args_namespace.hostname,
241                              args_namespace.os_username,
242                              args_namespace.os_password,
243                              args_namespace.os_tenant_name)
244     qemu_conn = qemu_connect(args_namespace.hostname, args_namespace.
            username)
245     keystone_conn = keystone_connect(args_namespace.hostname,
246                                      args_namespace.os_username,
247                                      args_namespace.os_password,
248                                      args_namespace.os_tenant_name)
249     neutron_conn = neutron_connect(args_namespace.hostname,
            keystone_conn)
250
251     try:
252         pt = prettytable.PrettyTable(["Property", "Value"])
253         pt.align = "l"
254         servers = nova_conn.servers.list(detailed=True)
255         for srv in servers:
256             interface_list = srv.interface_list()
257             for interface in interface_list:
258                 router_id = get_router_id(neutron_conn, interface.
                        net_id)
259                 print ("\nTroubleshooting commands:")
260                 print ("ip netns exec qrouter-%s ip a" % router_id)
261                 print ("ip netns exec qrouter-%s ip r" % router_id)
```

```
262                 print ("ip netns exec qrouter-%s iptables -t nat -L -nv
                        " % router_id)
263                 print ("ip netns exec qrouter-%s iptables -S -t nat" %
                        router_id)
264                 print ("ip netns exec qrouter-%s ping 8.8.8.8" %
                        router_id)
265
266             dom = qemu_conn.lookupByUUIDString(srv.id)  # get domain
                    from UUID
267             xml = dom.XMLDesc()
268             bridge, tap, mac = domiflist(xml)
269
270             transport = ssh_connect(args_namespace.hostname,
                    args_namespace.username, args_namespace.priv_key)
271             veth_pair1 = show_brctl_veth(bridge, transport)
272             peer_ifindex = ethtool_adapter_stats(veth_pair1, transport)
273             veth_pair2 = ip_link(peer_ifindex, transport)
274             ovsbridge = ovs_vsctl_port_to_br(veth_pair2, transport)
275             tag = ovs_vsctl_list_port(veth_pair2, transport)
276             vlan = ovs_ofctl_dump_flows(ovsbridge, tag, transport)
277
278             pt.add_row(["OpenStack Name", srv.human_id])
279             pt.add_row(["QEMU Name", dom.name()])
280             pt.add_row(["UUID", dom.UUIDString()])
281             pt.add_row(["tap", tap])
282             pt.add_row(["linuxbridge", bridge])
283             pt.add_row(["MAC Address", mac])
284             pt.add_row(["VETH Pair #1", veth_pair1])
285             pt.add_row(["VETH Pair #2", veth_pair2])
286             pt.add_row(["ovsbridge", ovsbridge])
287             pt.add_row(["TAG", tag])
288             pt.add_row(["VLAN", vlan])
289
290             print(pt)
291             pt.clear_rows()
292
293     except:
294         print sys.exc_info()
295         sys.exit(1)
296
297 if __name__ == '__main__':
298     main()
```

Listing 2.15: map_router output

```
1 Troubleshooting commands:
2 ip netns exec qrouter-2672c623-3b4e-4c28-9f4f-505b17a25542 ip a
3 ip netns exec qrouter-2672c623-3b4e-4c28-9f4f-505b17a25542 ip r
4 ip netns exec qrouter-2672c623-3b4e-4c28-9f4f-505b17a25542 iptables -t
    nat -L -nv
```

```
ip netns exec qrouter-2672c623-3b4e-4c28-9f4f-505b17a25542 iptables -S
    -t nat
ip netns exec qrouter-2672c623-3b4e-4c28-9f4f-505b17a25542 ping 8.8.8.8
+----------------+--------------------------------------+
| Property       | Value                                |
+----------------+--------------------------------------+
| OpenStack Name | ubuntu1204                           |
| QEMU Name      | instance-00000016                    |
| UUID           | e2c0d2b3-05c6-45e9-ad52-6f11647badbb |
| tap            | tap0598ef4e-cd                       |
| linuxbridge    | qbr0598ef4e-cd                       |
| MAC Address    | fa:16:3e:65:a7:1f                    |
| VETH Pair #1   | qvb0598ef4e-cd                       |
| VETH Pair #2   | qvo0598ef4e-cd                       |
| ovsbridge      | br-int                               |
| TAG            | 2                                    |
| VLAN           | 98                                   |
+----------------+--------------------------------------+
```

Chapter 3

Multinode install using Neutron with GRE

3.1 Physical Networking

The configuration of the physical network is significantly reduced when using tunneling. To make configuration easier I only used VLAN 253.

Listing 3.1: Nova Node uplink interface; gi2/5

```
1 interface GigabitEthernet2/5
2  switchport access vlan 253
3  switchport mode access
4 !
```

Listing 3.2: Virtual Machine uplink; gi3/4

```
1 interface GigabitEthernet3/4
2  switchport access vlan 253
3  switchport mode access
4 !
```

Listing 3.3: Virtual Machine uplink; gi3/20

```
1 interface GigabitEthernet3/20
2  switchport access vlan 253
3  switchport mode access
4 !
```

3.2 Installation

First things first, we need to completely remove the OpenStack install on the physical compute node. To accomplish that I used the big hammer method.[1] This would also be useful if Packstack has issues with installation and you would like to start over without removing a snapshot or redeploying.

Listing 3.4: Remove OpenStack from all nodes

```
1 ssh root@viropspaw002 'bash -s' < big_hammer.sh
2 for i in {1..3}; do ssh root@viropspaw01$i 'bash -s' < big_hammer.sh;
    done
```

3.2.1 DNS

My current DNS servers are Windows 2012. To add multiple entries its just easier to use PowerShell, which is listed below.

Listing 3.5: DNS Entries

```
1 Add-DnsServerResourceRecordA -Name "viropspaw002" -ZoneName "
    virtomation.com" -IPv4Address 10.53.253.70 -ComputerName
    10.53.252.123
2 Add-DnsServerResourceRecordA -Name "viropspaw010" -ZoneName "
    virtomation.com" -IPv4Address 10.53.253.90 -ComputerName
    10.53.252.123
3 Add-DnsServerResourceRecordA -Name "viropspaw011" -ZoneName "
    virtomation.com" -IPv4Address 10.53.253.100 -ComputerName
    10.53.252.123
4 Add-DnsServerResourceRecordA -Name "viropspaw012" -ZoneName "
    virtomation.com" -IPv4Address 10.53.253.110 -ComputerName
    10.53.252.123
5 Add-DnsServerResourceRecordA -Name "viropspaw013" -ZoneName "
    virtomation.com" -IPv4Address 10.53.253.120 -ComputerName
    10.53.252.123
```

3.2.2 Virtual Machine Deployment

Just like with vSphere we will create a template to use for virtual machine deployment. First we need to upload the Fedora 19 x64 iso to the NFS domain.

Listing 3.6: Upload ISO

```
1 wget http://download.fedoraproject.org/pub/fedora/linux/releases/19/
    Fedora/x86_64/iso/Fedora-19-x86_64-netinst.iso
2 ovirt-iso-uploader -i ISO_NFS upload Fedora-19-x86_64-netinst.iso
```

[1] Uninstalling RDO

3.2.2.1 Disable firewalld

We need to disable firewalld and replace it with iptables as it is not supported with Havana.[2]

Listing 3.7: Disable firewalld, install iptables

```
yum install iptables -y
systemctl disable firewalld
systemctl enable iptables
```

3.2.2.2 Disable SELinux

The issue with SELinux and OpenStack has been resolved[3], lets disable it anyway.

Listing 3.8: Disable SELinux

```
setenforce 0 # does not persist
sed -i 's/SELINUX=enforcing/SELINUX=disabled/' /etc/selinux/config
# or
vi /etc/selinux/config
...
SELINUX=disabled
...
```

3.2.3 SSH Keys

Listing 3.9: SSH Key Copy

```
echo "StrictHostKeyChecking no" >> /root/.ssh/config
for i in {1..3}
do
        ssh-copy-id -i /root/.ssh/id_rsa.pub viropspaw01$i
done
ssh-copy-id -i /root/.ssh/id_rsa.pub viropspaw002
```

3.2.4 Packstack

Listing 3.10: Packstack remote install after big hammer

```
for i in {0..3}; do ssh viropspaw01$i "yum install openstack-packstack
    -y"; done
```

[2] Red Hat Bugzilla Bug #981583
[3] Red Hat Bugzilla Bug #995780

Listing 3.11: Configure Packstack

```
DATE=`date +"%Y_%m_%d_%H_%M_%S"`
packstack --gen-answer-file=packstack-${DATE}
vi packstack-${DATE}
```

Listing 3.12 is the differences between a newly generated answer file and one that has been configured for GRE. The left side is the original and the right has the changes. Occasionally when the line is too long the change is forced to a new line.

Listing 3.12: Changes in Packstack answer file

```
CONFIG_CEILOMETER_INSTALL=n
CONFIG_NTP_SERVERS=10.53.252.123,10.53.252.246
CONFIG_NAGIOS_INSTALL=y
CONFIG_GLANCE_HOST=10.53.253.70
CONFIG_CINDER_HOST=10.53.253.70
CONFIG_CINDER_VOLUMES_CREATE=n
CONFIG_NOVA_COMPUTE_HOSTS=10.53.253.70
CONFIG_NEUTRON_SERVER_HOST=10.53.253.100
CONFIG_NEUTRON_L3_HOSTS=10.53.253.100
CONFIG_NEUTRON_DHCP_HOSTS=10.53.253.100
CONFIG_NEUTRON_METADATA_HOSTS=10.53.253.100
CONFIG_NEUTRON_OVS_TENANT_NETWORK_TYPE=gre
CONFIG_NEUTRON_OVS_TUNNEL_RANGES=1:1000
CONFIG_NEUTRON_OVS_TUNNEL_IF=eth1
CONFIG_HORIZON_HOST=10.53.253.120
CONFIG_NAGIOS_HOST=10.53.253.120
```

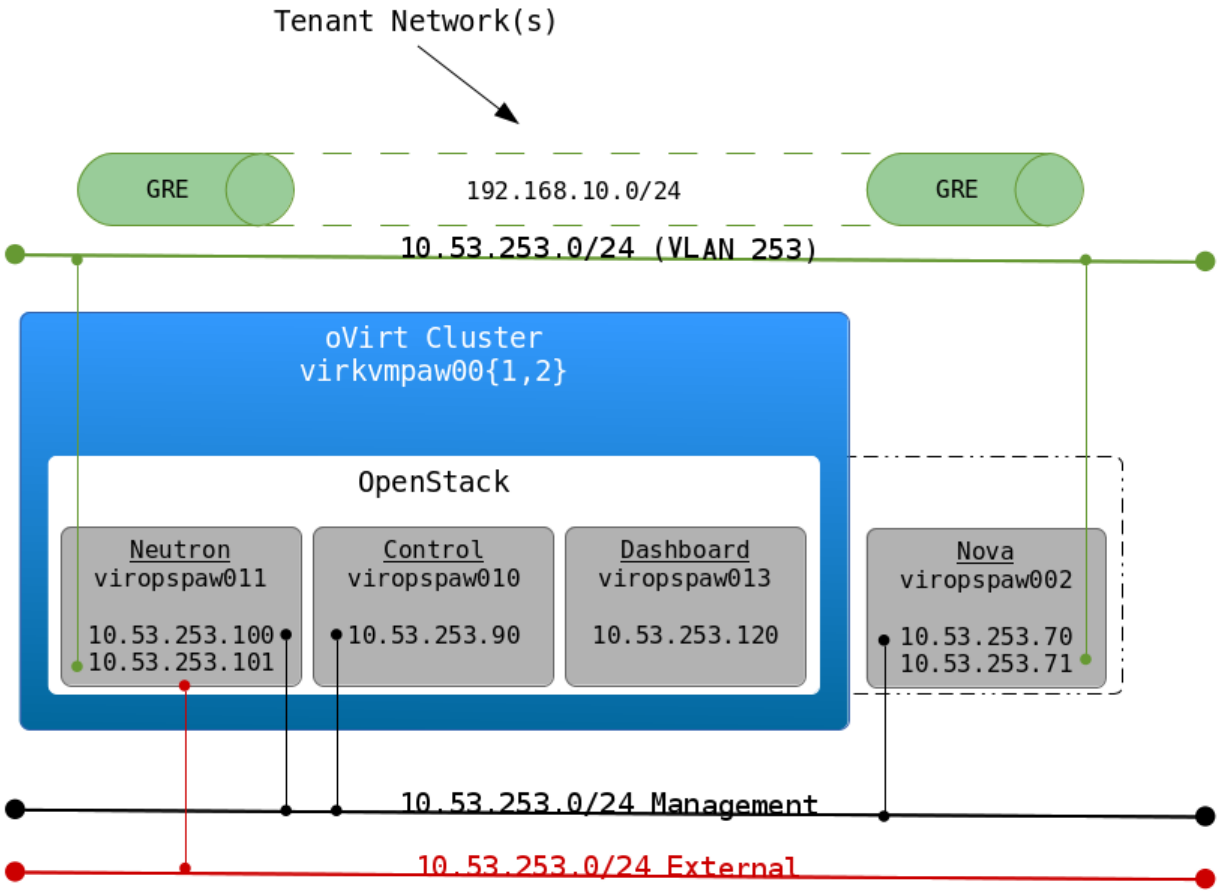
Listing 3.13: Execute Packstack

```
packstack --answer-file=packstack-${DATE}
```

Note: Monitor the progress in detail via /var/tmp/packstack/

3.3 Neutron Networking

Figure 3.1: OpenStack Node Details



Listing 3.14: Nova Node Open vSwtich

```
ovs-vsctl show
95dc7e47-0b2b-49e9-82ae-da4b60dccb68
    Bridge br-tun
        Port br-tun
            Interface br-tun
                type: internal
        Port "gre-1"
            Interface "gre-1"
                type: gre
                options: {in_key=flow, local_ip="10.53.253.71", out_key
                    =flow, remote_ip="10.53.253.101"}
        Port patch-int
            Interface patch-int
                type: patch
                options: {peer=patch-tun}
    Bridge br-int
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
        Port "qvo7ac76cfd-c8"
            tag: 4
            Interface "qvo7ac76cfd-c8"
        Port "qvoec307994-cd"
            tag: 4
            Interface "qvoec307994-cd"
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "1.11.0"
```

Listing 3.15: Neutron Node Open vSwtich

```
9a0a47e6-e002-4494-b217-5143e9419870
    Bridge br-int
        Port "tap5555b73f-79"
            tag: 1
            Interface "tap5555b73f-79"
                type: internal
        Port "qr-25c4c000-4b"
            tag: 1
            Interface "qr-25c4c000-4b"
                type: internal
        Port br-int
            Interface br-int
                type: internal
        Port patch-tun
            Interface patch-tun
                type: patch
                options: {peer=patch-int}
```

```
18      Bridge br-tun
19          Port patch-int
20              Interface patch-int
21                  type: patch
22                  options: {peer=patch-tun}
23          Port br-tun
24              Interface br-tun
25                  type: internal
26          Port "gre-2"
27              Interface "gre-2"
28                  type: gre
29                  options: {in_key=flow, local_ip="10.53.253.101",
                        out_key=flow, remote_ip="10.53.253.71"}
30      Bridge br-ex
31          Port "qg-f7e557b2-b4"
32              Interface "qg-f7e557b2-b4"
33                  type: internal
34          Port "eth2"
35              Interface "eth2"
36          Port br-ex
37              Interface br-ex
38                  type: internal
39      ovs_version: "1.11.0"
```

3.3.1 External

Listing 3.16: Create public network and subnet

```
1 neutron net-create public_network --router:external=True
2 neutron subnet-create public_network 10.53.253.0/24 --name
      public_subnet --enable_dhcp=False --allocation-pool start
      =10.53.253.150,end=10.53.253.160 --gateway=10.53.253.1
```

3.3.2 Tenant

Listing 3.17: Create admin tenant router, network and subnet

```
1 neutron router-create admin_router
2 neutron net-create admin_network
3 neutron subnet-create admin_network 192.168.10.0/24 --name admin_subnet
```

Listing 3.18: Set the router gateway

```
1 neutron router-gateway-set admin_router public_network
```

3.3.3 Troubleshooting

This section is devoted to resolving issues that I encountered while installing and configuring OpenStack.

3.3.3.1 Issue deleting floating IP port

Listing 3.19: Stop Neutron Services

```
NEUTRON_SERVICES=`systemctl --all --full --no-page -t service | grep
    neutron | awk '{print $1}'`
for SERVICE in $NEUTRON_SERVICES; do systemctl stop $SERVICE; done
```

Listing 3.20: Find the MariaDB root password

```
grep -i mysql packstack-2013_12_05_10_01_35
...
# Username for the MySQL admin user
CONFIG_MYSQL_USER=root
# Password for the MySQL admin user
CONFIG_MYSQL_PW=5729f733fd9f4731
...
```

Listing 3.21: Login to MariaDB

```
mysql -u root -p
```

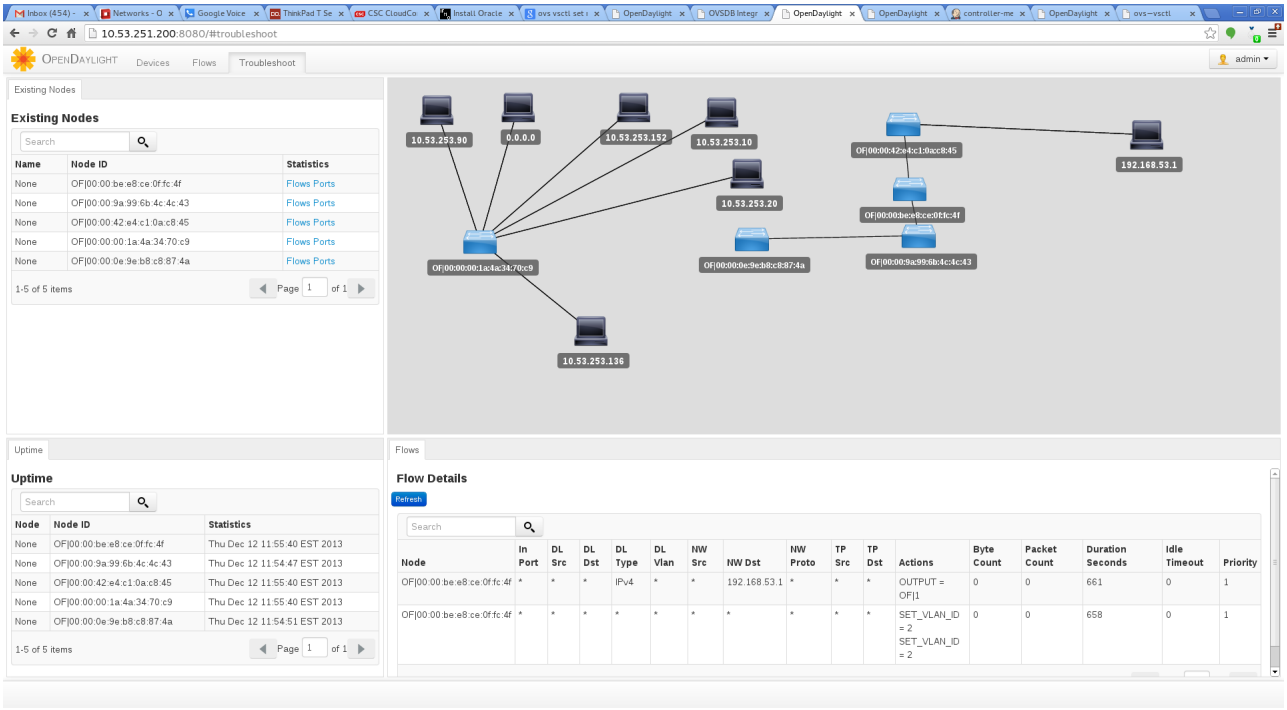
Listing 3.22: Get the ID for the floating IP and delete

```
use ovs_neutron;
select * from floatingips;
+--------------------------------------+
| id                                   |
+--------------------------------------+
| f5af35d0-7a4c-459c-85dc-9f6af9fb1b6c |
+--------------------------------------+
delete from floatingips where id = 'f5af35d0-7a4c-459c-85dc-9
    f6af9fb1b6c'
```

3.4 OpenDaylight

I just started to test with OpenDaylight controller. I am fairly certain that I set this up incorrectly. The only positive is a web-based UI for viewing Open vSwitch / OpenStack networking.

Figure 3.2: OpenDaylight Controller UI



3.4.1 Install Controller

I built a new Fedora 19 x64 virtual machine for the OpenDaylight controller.

Listing 3.23: Install Oracle Java JDK

```
1 sudo yum install jdk-7u45-linux-x64.rpm
```

Listing 3.24: Disable firewalld

```
1 systemctl stop firewalld
2 systemctl disable firewalld
```

Listing 3.25: Get the latest controller

```
1 wget https://jenkins.opendaylight.org/controller/job/controller-merge/
    lastSuccessfulBuild/artifact/opendaylight/distribution/opendaylight/
    target/distribution.opendaylight-osgipackage.zip
2 unzip distribution.opendaylight-osgipackage.zip
```

Listing 3.26: Start the controller

```
cd opendaylight
./run.sh
```

3.4.2 Configure Open vSwitch

Listing 3.27: Neutron node set controller and manager

```
ovs-vsctl set-controller br-tun tcp:10.53.251.200
ovs-vsctl set-controller br-ex tcp:10.53.251.200
ovs-vsctl set-controller br-int tcp:10.53.251.200
ovs-vsctl set-manager tcp:10.53.251.200
```

Listing 3.28: Nova node set controller

```
ovs-vsctl set-controller br-tun tcp:10.53.251.200
ovs-vsctl set-controller br-int tcp:10.53.251.200
```

Chapter 4

Instances

4.1 Security Group Configuration

RDO/Packstack creates two "default" security groups. This seems to cause issues with iptables as the instances created are not network available. First we will delete the rules.

Listing 4.1: Delete existing security group rules

```
SECURITY_GROUP_RULE_LIST=`neutron security-group-rule-list \
--column id --format csv \
| awk 'NR > 1 split($0, a, "\"") {print a[2]}'`
for ID in ${SECURITY_GROUP_RULE_LIST};
do
neutron security-group-rule-delete $ID;
done
```

Now we will delete the groups themselves. If there is a error stating that the group cannot be deleted because a rule exists that is OK. Just as long as one of the "default" groups has been deleted.

Listing 4.2: Delete existing security groups

```
SECURITY_GROUP_LIST=`neutron security-group-list --column id --format
    csv | awk 'NR > 1 split($0, a, "\"") {print a[2]}'`
for ID in ${SECURITY_GROUP_LIST};
do
neutron security-group-delete $ID;
done
```

The rules have been deleted lets create our own. Of course these rules open all available ports, probably not what you want to do in production.

```
neutron security-group-rule-create --protocol icmp --direction ingress
    default
```

```
2 neutron security-group-rule-create --protocol tcp --port-range-min 1 --
    port-range-max 65535 --direction ingress default
3 neutron security-group-rule-create --protocol tcp --port-range-min 1 --
    port-range-max 65535 --direction egress default
```

4.2 Create Images

OpenStack documentation has details for obtaining images for QEMU/KVM Windows and Linux flavors with cloud-init already installed. See link in sources.

4.2.1 Ubuntu LTS 12.04

4.2.2 Windows 2012 R2

The problem that I ran into was that my / was too small. When using LVM for KVM machines qcow2 images must be converted to raw and copied (dd) into the volume that is created. Glance and/or Cinder use the / filesystem to perform this action (I need to research more information about this). So in order to add Windows I had to manually convert qcow2 image to RAW within a temporary logical volume and then copy into a new logical volume.

Listing 4.3: Windows 2012 R2 Image Create

```
1 gunzip -cd windows_server_2012_r2_standard_eval_kvm_20131031.qcow2.gz |
     glance image-create --name "Windows Server 2012 R2 Std Eval" --disk
    -format=qcow2 --container-format=bare
2 +------------------+--------------------------------------+
3 | Property         | Value                                |
4 +------------------+--------------------------------------+
5 | checksum         | 5fb0a80e56479d31f8b5c1a8e6339206     |
6 | container_format | bare                                 |
7 | created_at       | 2013-11-06T17:35:34                  |
8 | deleted          | False                                |
9 | deleted_at       | None                                 |
10 | disk_format      | qcow2                                |
11 | id               | 32c94fcb-ea3a-4c69-8a49-a6095b8b25ba |
12 | is_public        | False                                |
13 | min_disk         | 0                                    |
14 | min_ram          | 0                                    |
15 | name             | Windows Server 2012 R2 Std Eval      |
16 | owner            | c70e8967165a4b57b8cd0d4265104f01     |
17 | protected        | False                                |
18 | size             | 17182752768                          |
19 | status           | active                               |
20 | updated_at       | 2013-11-06T17:39:48                  |
21 +------------------+--------------------------------------+
22
```

```
23 lvcreate -n temp -L 100G cinder-volumes
24 mkfs.ext3 /dev/cinder-volumes/temp
25 mount /dev/cinder-volumes/temp /mnt/tmp/
26 cd /mnt/tmp/
27 qemu-img convert /var/lib/glance/images/32c94fcb-ea3a-4c69-8a49-
       a6095b8b25ba -O raw ./windows.raw
```

4.3 Create Instance

Before we create an instance lets import the public key.

Listing 4.4: Add public key

```
1 nova keypair-add userkey --pub-key /root/.ssh/id_rsa.pub
```

When working with a Windows OS the Administrator password will be set by nova. To retrieve the password use the following command.

Listing 4.5: Windows administrator password

```
1 nova get-password windows /root/.ssh/id_rsa
2 38MQ21a1LxYMpH
```

Appendix A

Notes

A.1 Neutron

With external egress and ingress virtual machine traffic traversing the Neutron node it should be highly-available. How is this accomplished? What is the length of downtime or in a GRE/VXLAN scenario have multiple "uplinks" to another Neutron host?

```
Port "gre-1"
            Interface "gre-1"
                type: gre
                options: {in_key=flow, local_ip="10.53.253.71", out_key
                    =flow, remote_ip="10.53.253.101"}
```

A.2 Horizon

How to limit users on Horizon to create instances with new volume.

A.3 Cinder

When deleting a volume cinder by default will dd /dev/zero this takes forever.

Listing A.1: Change Cinder volume clear

```
sed -i 's/#volume_clear=zero/volume_clear=none/' /etc/cinder/cinder.
    conf
# or
vi /etc/cinder/cinder.conf
...
# Method used to wipe old voumes (valid options are: none,
# zero, shred) (string value)
volume_clear=none
```

```
...
SERVICES=`systemctl --no-page --full -t service | grep openstack-cinder
    | awk '{print $1}'`
for CINDER in $SERVICES; do systemctl restart $CINDER; done
```

A.4 Nova

How to properly configure Windows images with the correct QEMU/KVM settings?

Appendix B

Links

Under the hood Open vSwitch
Neutron with OVS and VLANs
Install RDO Havana on Fedora 19
Obtaining Images for OpenStack
Awk extracting substring using regular expressions with capture groups