Nested Virtualization with Contrail

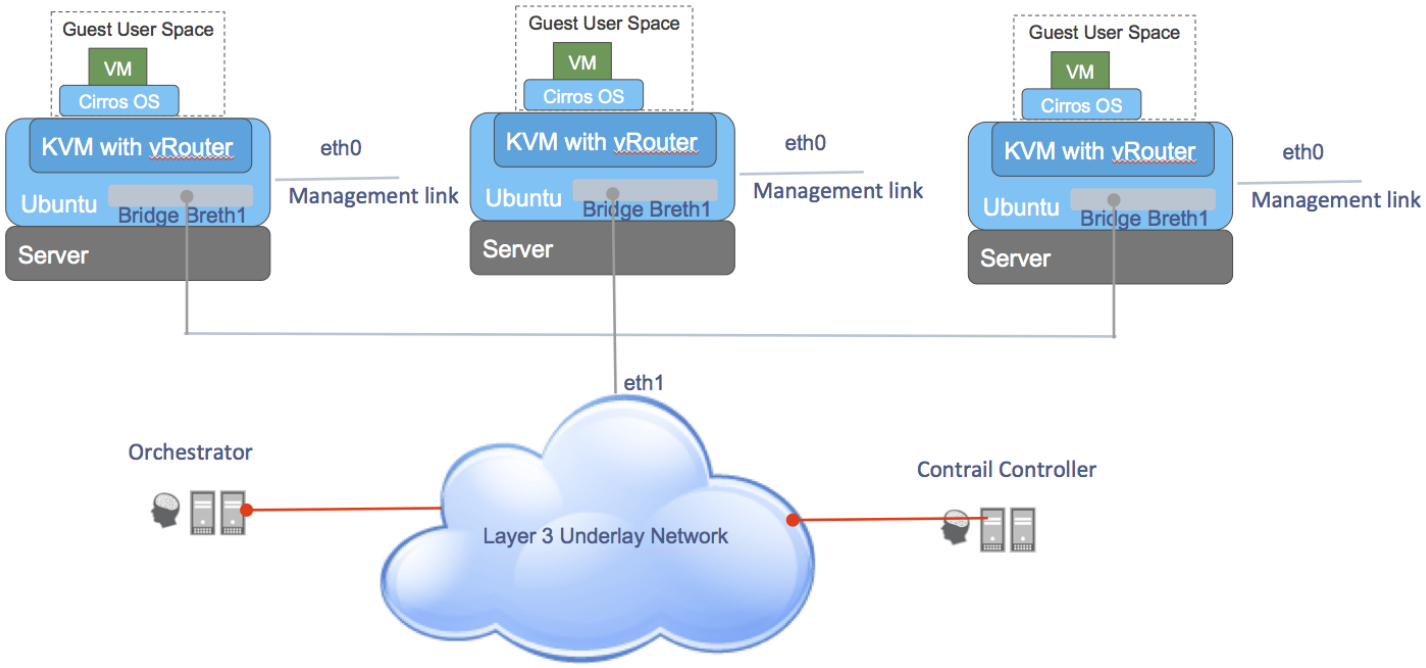
March 2016

Sethuraman Ramanathan - System test
Thiyagarajan Devendiran - System test

Executive Summary:

Generally type-1 hypervisors are designed to run on-top of a bare-metal x86 machine, with a processor that supports virtualization, i.e. Intel VMX and AMD SVM. Using this, we can make a physical machine run multiple virtual machines on-top (using associated components like qemu and libvirt).

In our case, we want to test the scale of MPLSoUDP tunnels from compute nodes with the Juniper Contrail SDN solution. Each tunnel terminates on one hypervisor. Normally each x86 machine has one hypervisor. As shown in the below picture, to test the scale of 1000 tunnels, we will need to have 1000 physical x86 machines (with hypervisors installed).

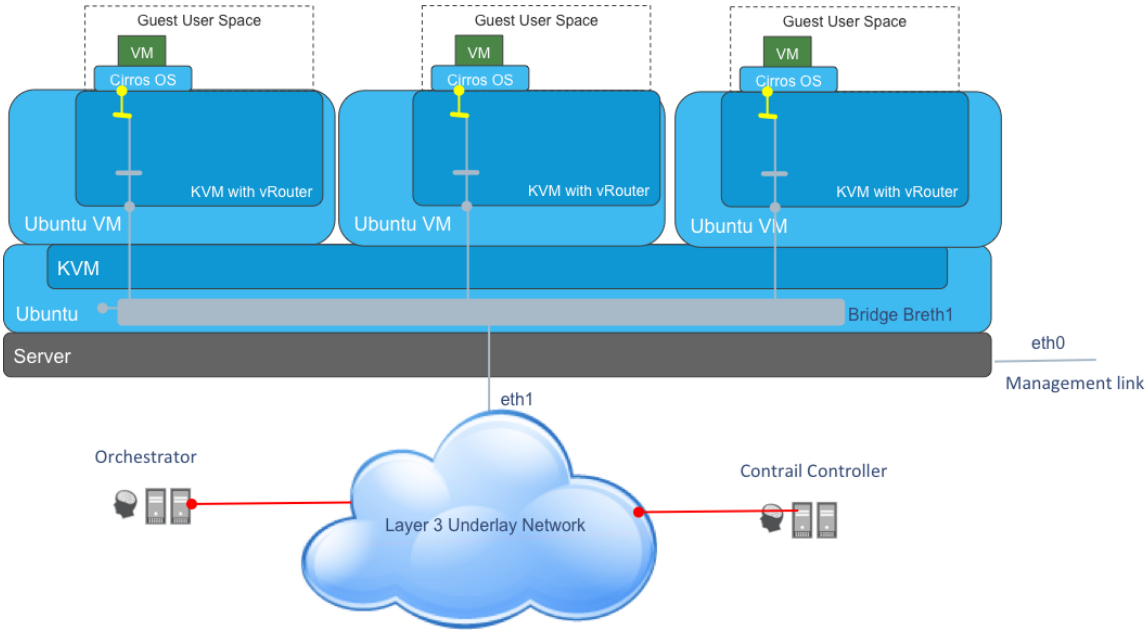


To over come this huge hardware constraint, we use ‘nested virtualization’. Nested virtualization is the act of running a hypervisor inside a virtual machine, effectively

nesting a hypervisor within a hypervisor. Using this method, we can run multiple hypervisors on the same physical host. The hypervisor running on the physical hardware is referred to as the host hypervisor. The hypervisors running within the VM are referred as the guest hypervisor or nested hypervisor.

There could be some performance problems with doing this, but for a test environment, it's quite a nice solution.

Overview:



The above picture shows one physical x86 machine and 3 nested hypervisors created in this machine. First, on a baremetal server we need to install an Ubuntu Linux OS. After that, install the KVM hypervisor (host hypervisor) on top of Ubuntu.

Create three guest VMs on the KVM host hypervisor. After this install KVM (guest hypervisor) on each of the guest VMs. This step is done by adding each VM as a compute node in the OpenStack controller/Contrail controller.

Once this is done the OpenStack controller/Contrail controller treats each VM as a separate compute node. Now from the OpenStack controller we can spawn VMs on these compute nodes.

After this the traffic sent from each created VMs would be forwarded through separate overlay tunnels.

Description:

Steps to create nested virtualization in KVM with the Ubuntu 14.04 version are given below. We have 2 physical NICs, eth0 and eth1, on an x86 physical machine. Eth0 is used for management. Eth1 is used as the uplink for all the nested VMs(guest hypervisors). This is done by bridging the physical NIC Eth1 with all nested VMs vNIC.

First, we need to install Ubuntu 14.04 and KVM on the physical server. Then VMs are created on top of KVM (host hypervisor). These VMs are added to the Contrail controller as compute hosts. Contrail installs vRouter and KVM on these VMs. After this we can create VMs inside these compute node, on top of the vRouter.

For our MPLSoUDP testing, we have used 'nested virtualization' on compute node, due to huge scaling requirement. For controller nodes, we have used physical servers.

1. Install Ubuntu 14.04 version on physical x86 machine. We have used "ubuntu-14.04-server-amd64.iso".

root@test:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 14.04 LTS
Release: 14.04
Codename: trusty

root@test:~# uname –a
Linux vxlan-lnx01 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

root@test:~# uname -r
3.13.0-24-generic

```
root@test:~# cat /sys/module/kvm_intel/parameters/nested
Y
```

2. Configure basic networking on physical NIC eth0, for management purposes. Edit the /etc/network/interfaces file, as per your network details.

```
# The primary network interface
auto eth0
iface eth0 inet static
  address xx.xx.xx.xx
  netmask xx.xx.xx.xx
  network xx.xx.xx.xx
  broadcast xx.xx.xx.xx
  gateway xx.xx.xx.xx
  # dns-* options are implemented by the resolvconf package, if installed
  dns-nameservers xx.xx.xx.xx
```

3. Install KVM, libvirt and bridge utilities on the physical server.

```
$ sudo apt-get install qemu-kvm libvirt-bin bridge-utils
```

4. Create a bridge and add physical NIC (eth1) to the bridge.

```
brctl addbr breth1
brctl addif breth1 eth1
```

5. Configure IP address to the bridge and clear the IP address from eth1.

```
ifconfig eth1 0.0.0.0
ifconfig breth1 30.30.0.2 netmask 255.255.0.0
ifconfig breth1 down
ifconfig breth1 up
```

6. Verify the bridge to eth1 mapping.

```
root@test:~# brctl show
bridge name     bridge id               STP enabled     interfaces
<…>
breth1          8000.0cc47a0d1e85       no              eth1
```

7. Add appropriate routes to the controller nodes via eth1. Edit the /etc/network/interfaces file, as per your network details.

```
auto breth1
```

```
iface breth1 inet static
 address 30.30.0.2
 netmask 255.255.0.0
dns-nameservers xx.xx.xx.xx
 dns-search <.>
 bridge_ports eth1
 bridge_stp off
bridge_ageing 0
 post-up route add -net 2.2.2.0 netmask 255.255.255.0 gw 30.30.0.1
   post-up route add -net 4.4.4.0 netmask 255.255.255.0 gw 30.30.0.1
   post-up route add -net 3.3.3.0 netmask 255.255.255.0 gw 30.30.0.1
   post-up route add 10.10.0.0 netmask 255.255.0.0 via 30.30.0.1
```

Note: Also appropriate routes need to be added on the Contrail controller to compute node.

8. Ping the gateway to verify the connectivity.

```
root@test:~# ping 30.30.0.1
PING 30.30.0.1 (30.30.0.1) 56(84) bytes of data.
64 bytes from 30.30.0.1: icmp_seq=1 ttl=64 time=0.497 ms
64 bytes from 30.30.0.1: icmp_seq=2 ttl=64 time=0.501 ms
64 bytes from 30.30.0.1: icmp_seq=3 ttl=64 time=0.457 ms
^C
--- 30.30.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1999ms
rtt min/avg/max/mdev = 0.457/0.485/0.501/0.019 ms
root@test:~#
```

9. Install virt-manager and ubuntu-desktop and then reboot the server.

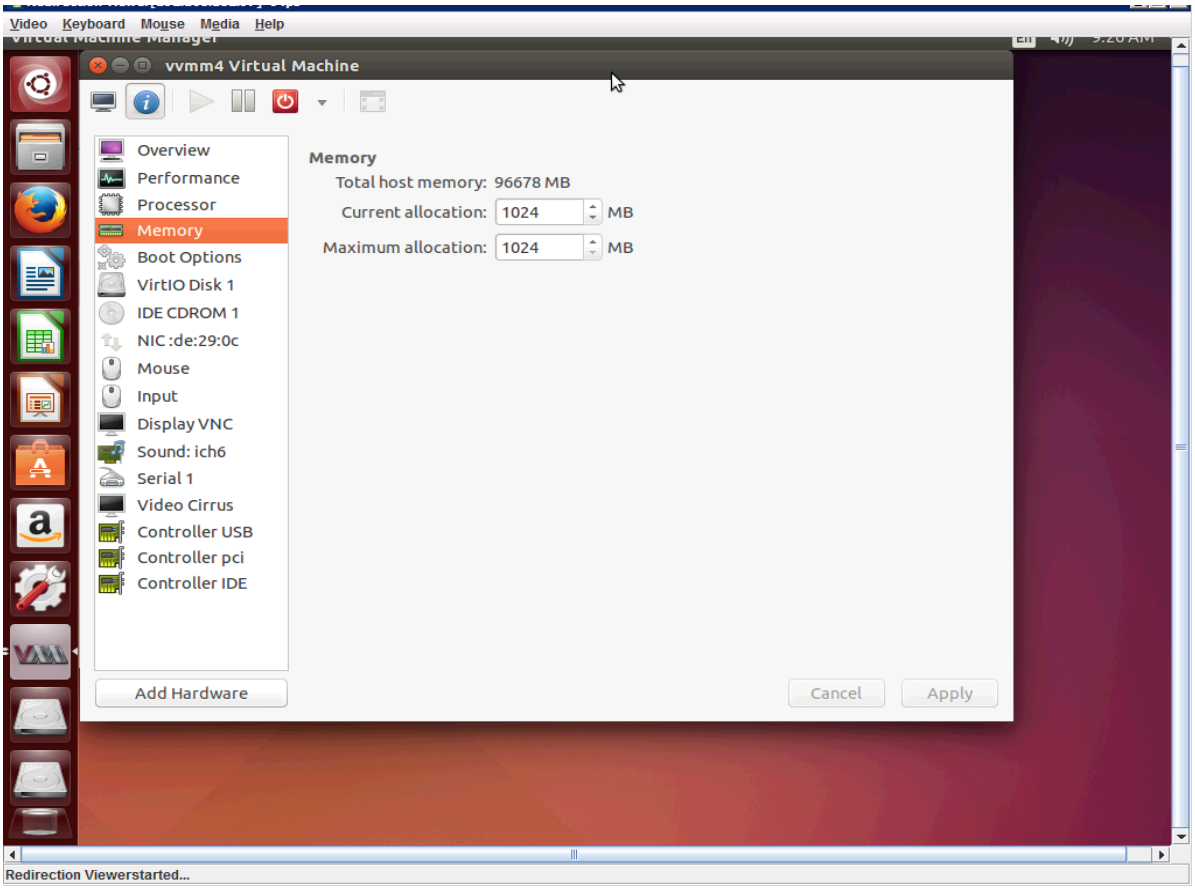
```
$apt-get install virt-manager ubuntu-desktop
```

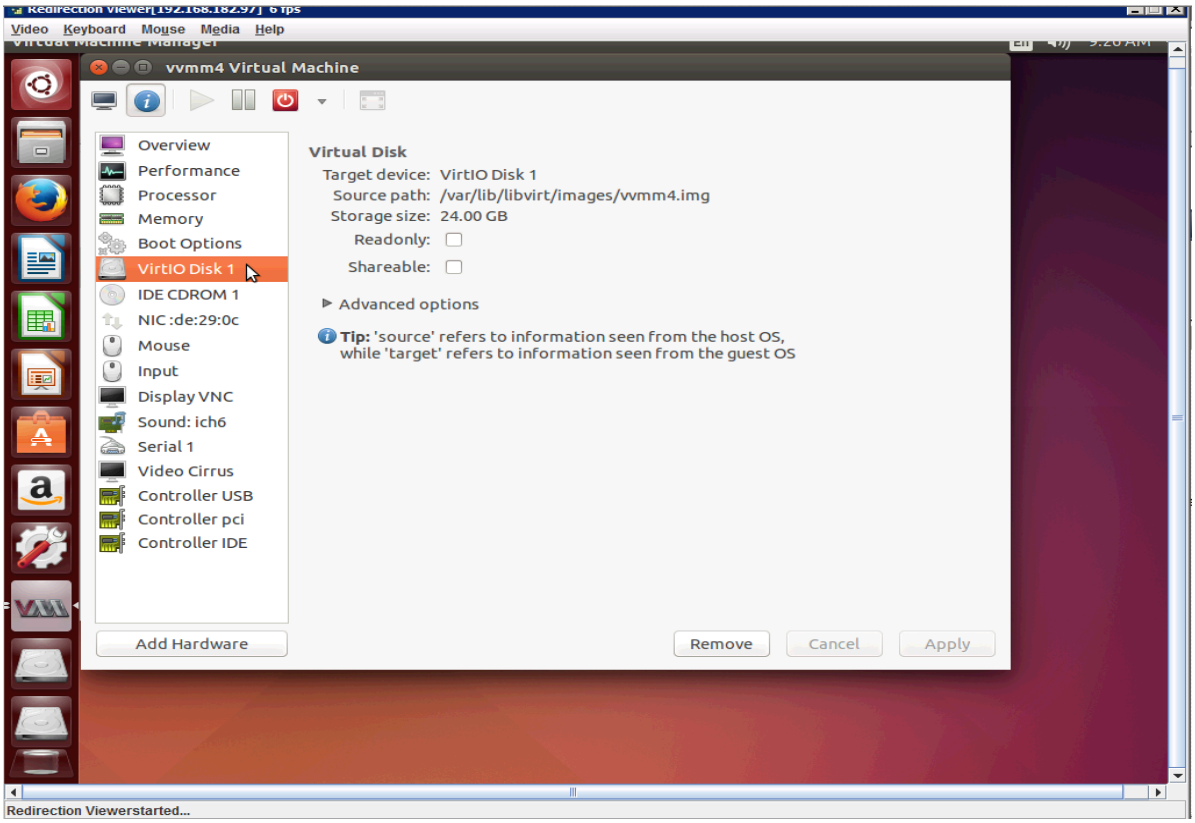
10. Download Ubuntu ISO image to /var/tmp folder. We have used “ubuntu-14.04-server-amd64.iso” for guest VMs.

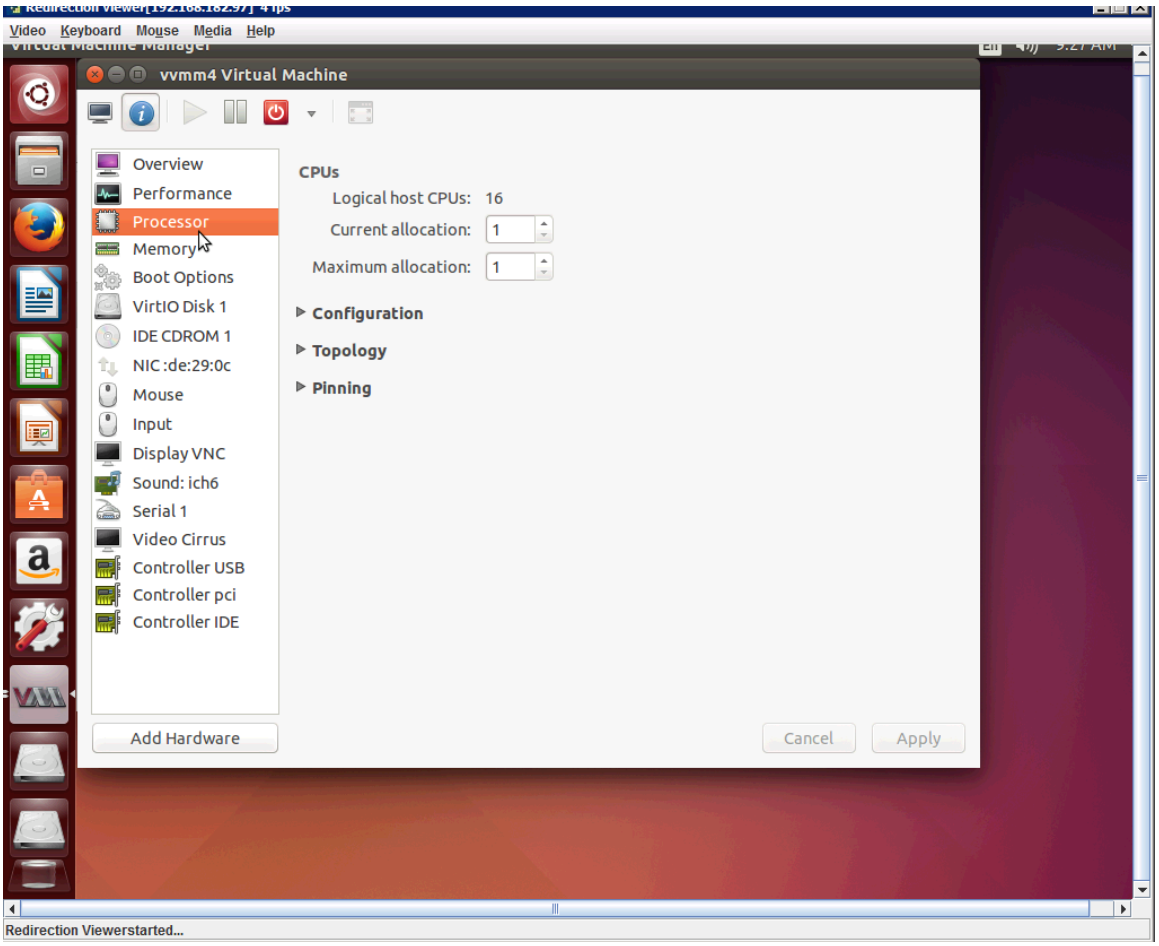
11. Login to physical server through IPMI WEB GUI. Create VM using ‘Virtual Machine Manager’ using the ISO image mentioned in step 10. Configure guest VM vNIC as part

this network 30.30.0.0/24. This network should have Internet access. This is required, as we need to install Ubuntu packages in these VMs.

We have used 1GB of RAM, 24GB harddisk and 1 cpu for the VM specifications.




Video Keyboard Mouse Media Help
vvmm4 Virtual Machine
Overview
Performance
Processor
Memory
Boot Options
VirtIO Disk 1
IDE CDROM 1
NIC :de:29:0c
Mouse
Input
Display VNC
Sound: ich6
Serial 1
Video Cirrus
Controller USB
Controller pci
Controller IDE
Virtual Disk
Target device: VirtIO Disk 1
Source path: /var/lib/libvirt/images/vvmm4.img
Storage size: 24.00 GB
Readonly:
Shareable:
Advanced options
Tip: 'source' refers to information seen from the host OS, while 'target' refers to information seen from the guest OS
Add Hardware
Remove
Cancel
Apply
Redirection Viewerstarted...



12. After creating guest VM, SSH to the physical machine and verify the VM status.

```
root@test:~# virsh list
 Id    Name                           State
----------------------------------------------------
 10    virvm3                         running
 11    virvm2                         running
 12    virvm4                         running
```

13. Then SSH to the guest VM from the physical machine shell.

```
root@test:~# ssh root@30.30.0.19
root@30.30.0.19's password:
root@virvm3:~#
```

14. On Guest VM, install NTP. This package is downloaded from Ubuntu website. So the VM needs to have Internet access to download this.

```
$apt-get install ntp
```

Note: If it gives an error like unable to find package then do "apt-get update" and then try again.

15. On Guest VM, enable root access for SSH.

user1@virvm3:~$ sudo -i
[sudo] password for user1:
root@virvm3:~#
root@virvm3:~# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

16. On Guest VM, add NTP server in the file /etc/ntp.conf

server <server name>

17. On Guest VM, restart ntp service

$sudo service ntp restart

18. On Guest VM, check NTP status

root@virvm3:~# ntpq -p
remote refid st t when poll reach delay offset jitter
==
****<ntp server> .GPS. 1 u 808 1024 377 10.780 -1.235 4.926***

root@virvm3:~#

19. On Guest VM, set the timezone data properly

$sudo dpkg-reconfigure tzdata

20. On Guest VM, edit /etc/network/interfaces and remove the DNS entries. You can add DNS entries after installing vROUTER. During vROUTER installation sometimes we see an error when we have a DNS entry.

 # The primary network interface
auto eth0
iface eth0 inet static
address 30.30.0.19
netmask 255.255.0.0
network 30.30.0.0
broadcast 30.30.255.255
gateway 30.30.0.1

21. On Guest VM, add the below entries in /etc/rc.local. This file is executed during start up. This will make sure these interfaces are brought up after reboot of VM.

```
ip link set pkto up
ip link set pkt1 up
ip link set pkt2 up
ip link set pkt3 up
ifup etho
ifup eth1
ifup vhosto
ifconfig etho up
ifconfig vhosto up
ifconfig pkto up
ifconfig pkt1 up
ifconfig pkt2 up
ifconfig pkt2 up
```

22. Now we will add vRouter to the VM virvm3. For this follow the steps 22,23,26 and 27 from the Contrail controller. Edit /opt/contrail/utils/fabfile/testbeds/testbed.py in the Contrail controller. Add the VM ip(30.30.0.19) and host name(virvm3) to the testbed.py file as shown below. In this case host5 is mapped to the VM virvm3.

```
host4 = 'root@30.30.0.18'
host5 = 'root@30.30.0.19'

env.roledefs = {
   'all': [host1, host2, host3, host4,host5,],
   'cfgm': [host1, host2, host3],
   'openstack': [host1],
   'control': [host1, host2, host3],
   'compute': [host4,host5,],
   'collector': [host1, host2, host3],
   'webui': [host1],
   'database': [host1, host2, host3],
   'build': [host_build],
}
env.passwords = {
   host1: 'password123',
   host2: ' password123',
   host3: ' password123',
   host4: ' password123',
   host5: ' password123',
```

```
host_build: ' password123',
}
env.ostypes = {
   host1: 'ubuntu',
   host2: 'ubuntu',
   host3: 'ubuntu',
   host4: 'ubuntu',
   host5: 'ubuntu',
   host6: 'ubuntu',
   host7: 'ubuntu',
}
env.hostnames = {

'all': ['<name1>',' <name2>',' <name3>','<name4>','virvm3']

}
```

23. Change the guest VM kernel version to 3.13.0-40 by running this fab command from the Contrail controller. IP address given here is the IP address of the VM we just created.

#fab upgrade_kernel_node:root@30.30.0.19

Note: After the command is done, reboot the VM and verify the kernel version.

```
root@virvm3:~# uname -r
3.13.0-40-generic
root@virvm3:~#
```

24. In this section we will give steps to clone VM.

So that we can skip steps from to 1 to 19 and 23 for the next VM creation. Here virvm8 is the new VM and virvm3 is old VM. After creating the new VM, change the hostname and IP address of the new VM, so that new VM will have a unique hostname and IP address.

For this login to the physical host then issue these commands.

Shutdown the old VM:

 # virsh shutdown virvm3

Clone the VM:

#virt-clone --connect=qemu:///system -o virvm3 -n virvm8 -f /var/lib/libvirt/images/virvm8

Start the new new VM:

 # virsh start virvm8

25. The following steps are required only in new cloned VM. Initial SSH login to new cloned VM should be with same old IP address.

a. Change the hostname under /etc/hostname

b. Change the hostname under /etc/hosts

c. Change IP address under /etc/network/interfaces

d. Reboot the new VM

After performing these steps, then SSH login to new VM would be with new IP address.

26. Install Contrail package to the VM. This fab command is issued from the Contrail controller.

#fab install_pkg_node:/tmp/ contrail-install-packages_2.21-102~ubuntu-14-04juno_all.deb,root@30.30.0.19

27. Then issue this fab command from the Contrail controller. This will install vRouter in this VM.

#fab add_vrouter_node:root@30.30.0.19

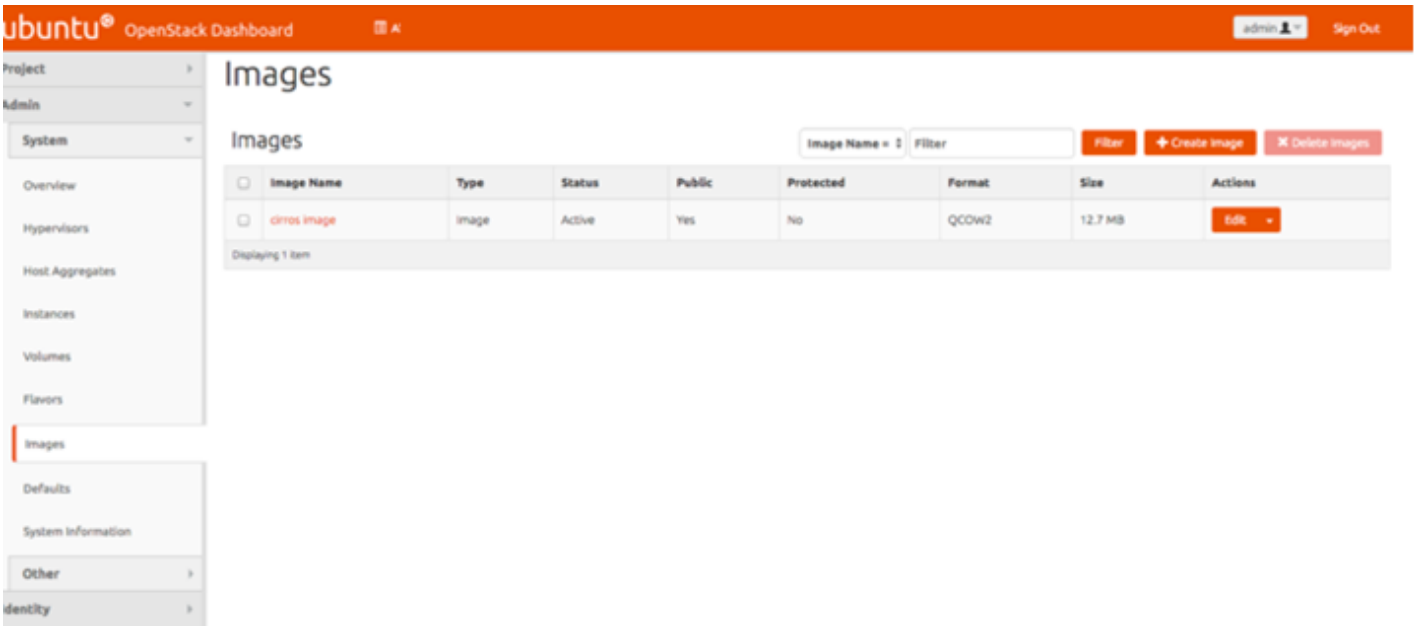
28. SSH to the VM and verify vRouter status.

root@virvm3:~# contrail-status
== Contrail vRouter ==
supervisor-vrouter: active
contrail-vrouter-agent active
contrail-vrouter-nodemgr active
root@virvm3:~#

29. Now repeat the steps 24 and 25, from compute node physical machine, to clone next VM. Then repeat the steps 22,23,26 and 27 from Contrail controller.

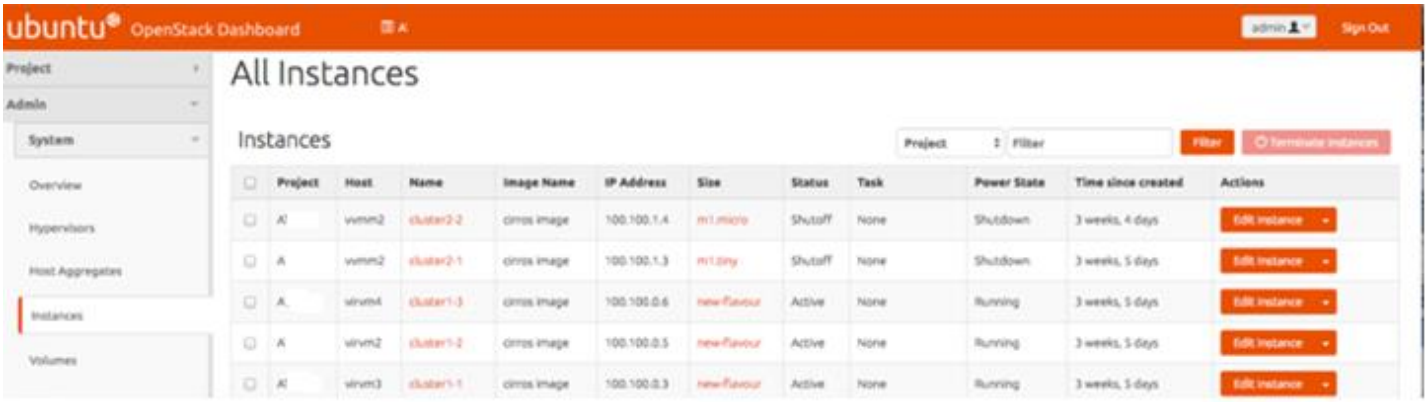
30. The number of nested VMs created on a physical machine depends the memory and CPU capacity of the machine. We have used 1GB of RAM, 24GB harddisk and 1 CPU for each VM.

31. Now we have the compute nodes ready. We can start using this to spawn VMs from the OpenStack controller, on the compute nodes. We can use the cirros image for spawning VMs on these compute nodes. This image will take up a small amount of memory.

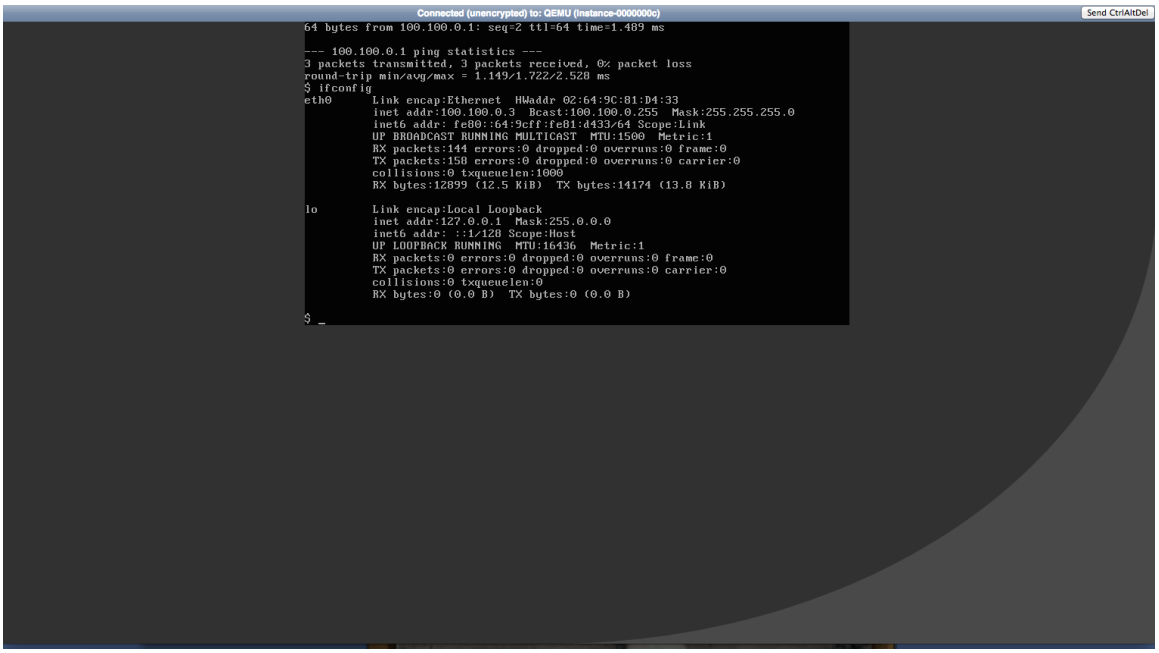


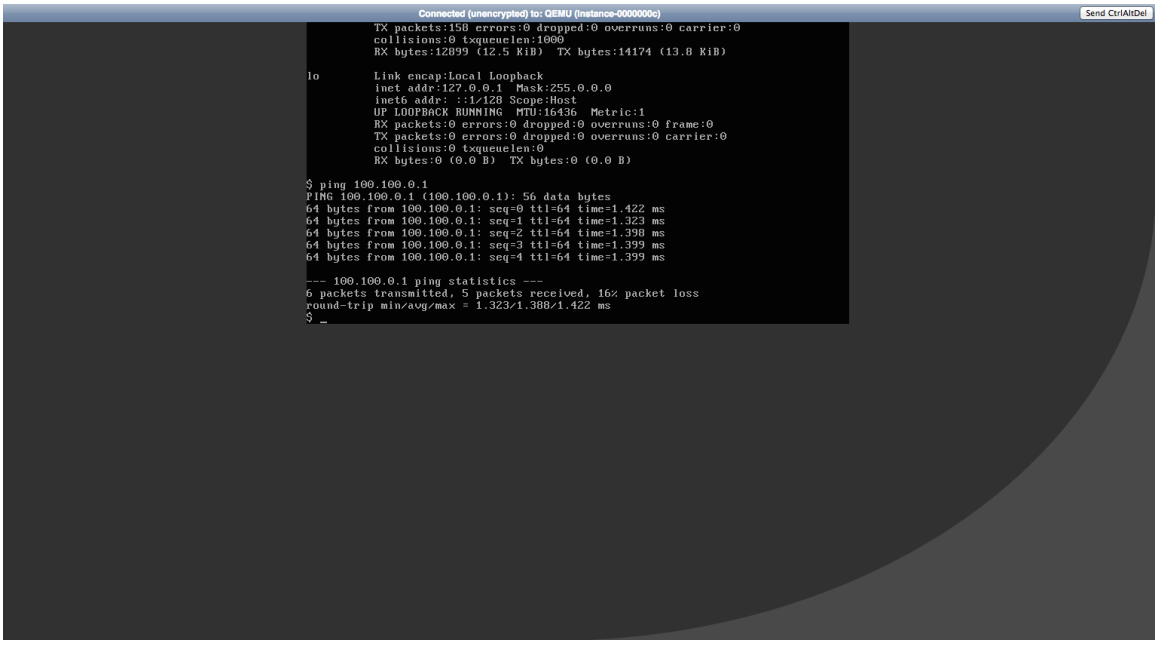
32.The following pictures from OpenStack controller show the virvm3 VM is acting as compute host.

33. This picture shows VM instance cluster1-1 with the ip address 100.100.0.3 is spawned on the compute host virvm3.



34. From the VM instance cluster1-1 test the connectivity to the gateway. Here the VM network segment is 100.100.0.0/24 and gateway is 100.100.0.1.





Conclusion

In this document we have shown the use case of nested virtualization. This document also gives details on how to implement nested hypervisors on Ubuntu version 14.04.