Department of Electrical and Computer Engineering

MINT 709

Capstone Project

Transforming IT service delivery by leveraging Private Clouds

Abstract

The transformation of IT began several years ago when VMware introduced server virtualization to increase the efficiency, utilization of servers and help customers consolidate data centers and gain agility and availability in the data centers. Since then, there has been a consistent progress towards virtualizing other services inside data centers such as storage, networking and security services. The end goal of this is to build truly service oriented shared infrastructure data centers which can be deployed in minutes. Traditionally, networks have not kept up with server virtualization, so even though servers can be brought up very quickly, networks have to be still provisioned manually which causes impedance and slows down the application delivery. In order to truly harness the private cloud movement, network virtualization is a key.

There are several options emerging in the market place for network virtualization with VMware NSX, Cisco ACI, Juniper Contrail, Alcatel NUage, PLUMgrid, Midokura and others. All of these products decouple the tenant network from the physical network hence closing the gap between server provisioning and network provisioning. Tenant networks are essentially overlayed tunnels that are provisioned dynamically on top of an underlay network and services are applied to those overlay tunnels.

In this project, I will be exploring the Juniper Contrail option and learn about building private cloud data center. Cloud computing infrastructures will likely be a key component of future data centers especially considering the emerging Network Virtualization and Software Defined Networking technologies. The cloud infrastructure will then determine the performance of the networking environment. This project will be implementing the Juniper OpenContrail, a widely adopted cloud infrastructure management platform. In particular, the project provides an insight on how OpenContrail deals with multi-tenant network virtualization and also how Juniper deals with the major issues of network virtualization, such as security, multi-tenant / virtual networks scalability, communication between VMs and physical devices etc. in an experimental test-bed.

Submitted to:
Dr. Mike Macgregor
Director MINT Program &
Professor Department of Computing Science
University of Alberta

Submitted by
Furqan Ali Khan

Acknowledgments

I cannot express enough thanks to my Head of Department of Computer Science for his continued support and encouragement: Dr. Mike Macgregor. I offer my sincere appreciation for the learning opportunities provided by my department. I also would like to thank Mr. Salman Zahid, my supervisor for his all guidance without which this project cannot be possible.

My completion of this project could not have been accomplished without the support of Mr. Shahnawaz Mir; thank you sir. Also, my children, to Umaynah, and Hana – thank you for allowing me time away from you to research, study, and write. You folks deserve a trip to Disney! Thanks to my parents as well who have been our constant source of inspiration.

Finally, to my caring, adoring, and supportive wife, Nighat: Your encouragement when the times got rough are much appreciated and duly noted. It was a great comfort and relief that you provided all management of our household activities while I completed my work and degree. My honest and deepest thanks.

…

“I would like to dedicate my work to… my loving wife, Nighat”

…

Table of Contents

Table of Figures

Introduction to Clouds

A cloud can be defined as a common environs built over virtualized framework. Virtualization has turns out to be broadly utilized technology in clouds because that has eliminated the traditional networking hurdles. A cloud models are categorized as per below:

- **Private cloud**: This framework is for the enterprises and businesses for their sole purposes as it provides good security as compare to public clouds.
- **Public cloud**: The cloud is open to over-all public.
- **Community cloud**: This is typically administrated by more than one enterprise that has a common industry.
- **Hybrid cloud**: This is a combination of public and private clouds.

A cloud whether it be private, public, or hybrid consists of the below core features:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity or expansion
- Measured service
- Shared by multiple tenants (1)

Cloud has three possible service models, which means there are three key sorts of services provided by cloud (2).

- **Infrastructure-as-a-service (IaaS):** In this form of service, service provider actually controls the infrastructure and client has the choice to select own platform.
- **Platform-as-a-service (PaaS):** In this form of service, service provider is responsible for infrastructure and platform in order to facilitate client. Client is only responsible for their applications.
- **Software as a service (SaaS):** In this form of service, client is neither handles infrastructure nor platform, instead only using software via web browser to access software or application which is actually hosted on provider's space. You can think of it as a hotmail, gmail, or yahoo mail.

OpenContrail is a system that can be used for Private clouds for Enterprises or Service Providers, Infrastructure as a Service (IaaS) and Virtual Private Clouds (VPCs) for Cloud Service Providers (3).

The Infrastructure as a Service (IaaS) involves in a multi-tenant virtualized data centers. Multiple-tenants in a data center share the similar infrastructure. For each tenant logical resources (virtual machines, virtual storage, and virtual networks) are assigned individually. These logical resources are separated from one another, unless particularly permitted (3).

Infrastructure-as-a-Service and private clouds offer a perfect method to solve some of your enterprise's major business and technology difficulties. A private cloud service delivery model

benefits in reducing budgets, reach new levels of effectiveness, and acquaint with state-of-the-art new industry models. Therefore, your organization may turn into more agile and competent, while streamlining its operations and infrastructure.

Private Cloud is one of the quickest expanding solutions these days and the importance of the safe multi-tenant Data Centre on business goals is growing. Juniper is deploying cutting-edge technology Data Centre solutions to automate and virtualize storage, networks, and servers. Juniper's enterprise Data Centre architecture and Infrastructure-as-a-Service are built on Contrail technology, and I am using this technology in this project to build a virtual network.

Network Virtualization

What is Network Virtualization?

Network Virtualization is well-organized use of network resources by logical separation of a solo network. For instance, several logical networks on a common infrastructure can be departments on an individual enterprise network. The primary thought behind NV is that it permits numerous VNs to overlay a cloud supplier's physical network and uproots the limits of VLAN and IP address task from virtual machine provisioning. This makes it simple for organizations to move to IaaS and capable for datacenter admins to deal with their infrastructure.

Need of Network Virtualization

The need for Network Virtualization emerges to cut down total cost of ownership through imparting resources of network whereas still keeping up secure separation between businesses, communities, or individuals.

Synopsis

For each virtual machine network, which can be made out of one or more virtual subnets, is not dependent of another virtual machine networks and also of the provider’s own core physical network. In other words, the exact physical location of an IP subnet on the provider’s physical network is separated from the virtual network topology of each client’s network (4).

The advantage of this separation is that customers can easily move physical server workloads to a provider’s cloud while preserving the IP addresses and network topology of their workloads. For instance, let’s say that an enterprise has few physical servers present in their on-site and having private IP addresses of 10.3.31.40, 10.3.31.41, and 10.3.31.42. If you virtualize the servers and then move them to the provider’s cloud. Your physical servers are using your address space 10.3.31.0/24, and provider uses 172.16.150.0/29 address space. As in this project, I have used 10.3.31.40 for my physical dell server running Ubuntu server edition and the virtual machines on different virtual networks, running on top of physical servers, are using 172.16.150.0/29 subnet (4).

The way this works is that network virtualization enables you to assign two different IP addresses to each virtual machine running any host. These two addresses are:

- **Customer Address (CA)** is the IP address that the server has when it exist on the customer's location before it was migrated into the cloud. In the above example, this might be the 10.3.31.40 address for a particular server that the customer wants to move to the cloud.
- **Provider Address (PA)** is the IP address assigned by the cloud provider to the server once the server has been migrated to the provider's data center. In the above example, this might be 172.16.150.11, or some other address in the 172.16.150.0/29 address space (4).

The CA for each virtual machine is mapped to the PA for the underlying physical host on which the virtual machine is running. Virtual machines link over the network by sending and receiving packets in the CA space. The virtual machine's packets are then encapsulated into new packets that have a PA as source and destination address so they can be routed over the provider's physical network (4). Juniper uses MPLSoUDP, MPLSoGRE, and VxLAN for this purpose.

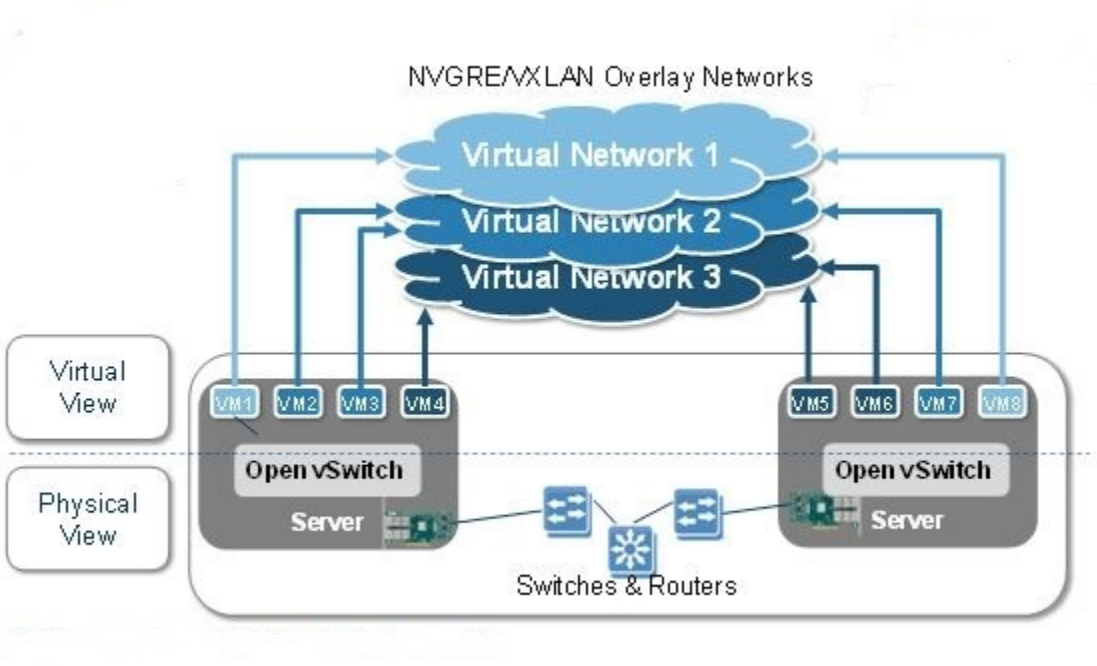


Figure 1: Physical and Virtual view of Virtual Networking (5).

The goal of network virtualization is to allow the provider to run multiple customer virtual networks on top of an underlying network the exact similar means as server virtualization runs multiple virtual servers on a single physical server. Network virtualization isolates each virtual network from every other virtual network so that each virtual network has the impression that it is an entirely separated network. Several clients can use the exactly same addressing scheme for their virtual networks (think of it as VRF in routing); client networks will remain fully separated from one another and impersonate that each network is the only one present with that particular addressing scheme.

OpenContrail (3)

Introduction

OpenContrail is intended to work in an open-source cloud environment. In order to provide a fully integrated end-to-end solution:

- The OpenContrail System is incorporated with Kernel-based Virtual Machines (KVM) and Xen.
- The OpenContrail System is integrated with open source virtualization orchestration systems such as OpenStack and CloudStack.
- The OpenContrail System is joined with open-source physical server administration systems such as chef, puppet, cobbler, and ganglia (3).

Juniper Networks also offers a commercial form of the OpenContrail System.

Controller and the vRouter (2)

The OpenContrail System contains two main components:

- The OpenContrail Controller.
    - Logically centralized, but physically distributed SDN controller.
    - Accountable of providing the management, control, and analytics functions of the virtualized network (3).
- The OpenContrail vRouter.
    - It is a forwarding plane (of a dispersed router) that runs in the virtualized abstract layer of a virtualized server (3).
    - Extends the physical network in a data center into a virtual overlay network accommodated in the virtualized servers.
    - Conceptually alike to current open source vSwitches, for instance the VMWare vSwitch (VSS) but it also provides routing and higher layer services (hence vRouter instead of vSwitch).

The OpenContrail Controller responsible for the logically central control plane and management plane of the system and orchestrates the vRouters.

The vRouters running in the hypervisors of the virtualized servers make a virtual overlay network over physical network using a mesh of dynamic “tunnels” amongst themselves. In OpenContrail these overlay tunnels can be MPLS over GRE/UDP tunnels, or VXLAN tunnels.

The underlay physical routers and switches do not contain any per-tenant state: they do not contain any Media Access Control (MAC) addresses, IP address, or policies for virtual machines. The forwarding tables of the underlay physical routers and switches only contain the IP prefixes or MAC addresses of the physical servers.

The vRouters, on the other hand, do contain per tenant state. They contain a separate forwarding table per virtual network (2).

Overlays based on MPLS L3VPNs and EVPNs (3)

The OpenContrail System is inspired by and theoretically very akin to standard MPLS L3VPNs (for L3 overlays) and MPLS EVPNs (for L2 overlays).

Data Plane:

- OpenContrail supports MPLS over GRE, a data plane encapsulation.
- MPLS over UDP (better multi-pathing and CPU utilization) and VXLAN.
- NVGRE will be added in the future releases.

Control Plane:

- BGP is a protocol amongst the control plane nodes and physical gateway router.
- Netconf uses for administration intentions.
- The protocol between controller and vRouter is XMPP.

Control and Management Plane Protocols being used in OpenContrail

IF-MAP (6)

- It is an open standard client/server protocol developed by the Trusted Computer Group (TCG).
- Control and Configuration nodes exchanges data by means of IF-MAP.
- IF-MAP offers an extensible mechanism for characterizing high and low level configuration data models.

XMPP (7)

- The Extensible Messaging and Presence Protocol (XMPP) is an application profile of the XML that allows the near-real-time exchange of organized, however, extensible data between any two or more network objects.
- Developed originally within the Jabber open-source community.
- It is a south-bound protocol serving in the OpenContrail system for the purpose of exchanging various information between compute and control nodes, for instance, routes, configuration, state, forwarding policy, proxy requests, statistics, logs, and events.

BGP

- It is a south-bound protocol; OpenContrail uses BGP to interchange routing info between control nodes.
- iBGP sessions are made between Control nodes for synchronization to check the network state.
- Control nodes and gateway nodes exchange routes with each other.

SANDESH (8)

- Sandesh is the XML over TCP protocol utilize by the software units in the Contrail Controller and Contrail vRouter to transport the data to the Analytics node.
- There are two types of Sandesh data:
    - Asynchronous messages to the Analytics Node to report system logs, traces, events, flow statistics appears in the Contrail Controller and the Contrail vRouter (3).
    - The analytics node request and receive response messages with components in control and configuration nodes to collect exact working state.
- A NoSQL database is being used for storing the information collected from above mentioned point.

NETCONF (9)

- This south-bound protocol defines a simple way through which a physical network device can be taken care of, and configuration data information can be recover, and new configuration data can be uploaded and manipulated.
- A main facet of NETCONF is that it permits the functionality of the management protocol to carefully reproduce the built-in functionality of the device (3).

OpenContrail Architecture (3)

For each node in OpenContrail system can be employed as an isolated physical server or it can be implemented as a Virtual Machine. None of the single node becomes victim of bottleneck because all the nodes of any type run in active-active configuration state, thus providing redundancy and horizontal scalability (*Horizontal scalability is the ability to increase capacity by connecting multiple hardware or software entities so that they work as a single logical unit*).

Compute Node

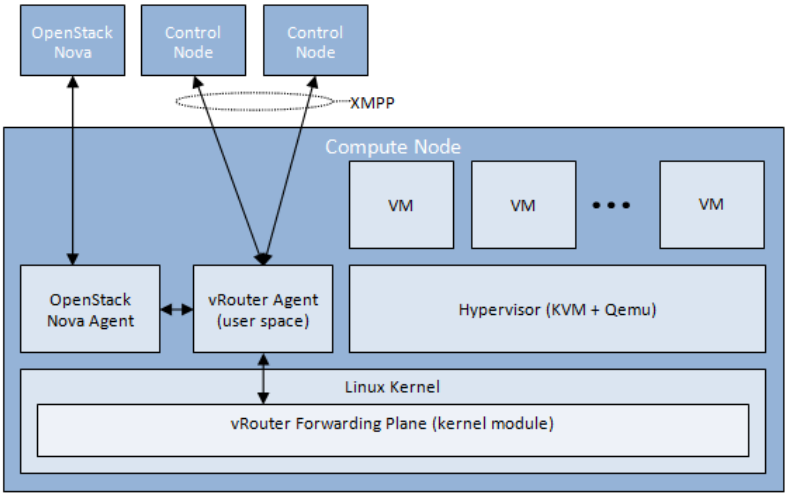


Figure 2: Organization of Compute Node

- Compute node hosts VMs. And these can be tenant VMs running any application:
  - i.e. web/database servers, any enterprise wide application or,
  - Service Chaining, aka Network Function Virtualization (NFV) involves in orchestration and management of networking functions such as a Firewalls, Intrusion Detection or Preventions Systems (IDS / IPS), Deep Packet Inspection (DPI), caching, WAN optimization, etc. in virtual machines instead of on physical hardware appliances (3).
- Configuration requires Linux as a host OS and KVM/Xen serving the purpose of virtualized layer.
  - VMware ESXi or Windows Hyper-V will also be supported in the future (3).
- Every occurrence of a compute node runs the following processes:

- nova-compute
- contrail-vRouter

Compute node holds vRouter that implements the forwarding plane and the distributed part of the control plane. The OpenContrail vRouter is theoretically akin to current commercial and open source vSwitches, for example, the Open vSwitch, but it also provides routing and higher layer services (therefore, vRouter instead of vSwitch) (3).

- The vRouter forwarding plane resides in the Linux Kernel.
- The vRouter Agent serve as a local control plane.

The vRouter Agent (3)

- It is a process running in user space inside Linux shell.
- It acts as a local, lightweight control plane.
  - Feign *vRouter agent* exact same as a Control Plane in any type of router.
    - The vRouter agent is primarily about the learning of routes.
- It report system logs, and events to the analytics node.
- It exchanges routes with the Control node using XMPP.
- vRouter agent configures the virtual network for the newly create virtual machine informed by Nova agent.
- It also handle the requests for DHCP, DNS, and ARP (ARP is only for L3 device, there is no ARP for L2 devices).

The vRouter Forwarding Plane (3)

- It runs in a Linux Kernel.
- Encapsulate and de-capsulate packets sent back and forth from the overlay network (don't mix it with underlay (physical) network).
- Allocating packets to the routing occurrence, based on MPLS labels or VNIs.
- The routes could be L3 prefixes or L2 MACs.
- It keeps the routing table for L3 packets and CAM table for L2 frames.

- Furthermore, it does supports MPLSoUDP, MPLSoGRE, and VxLAN encapsulation in the overlay network.

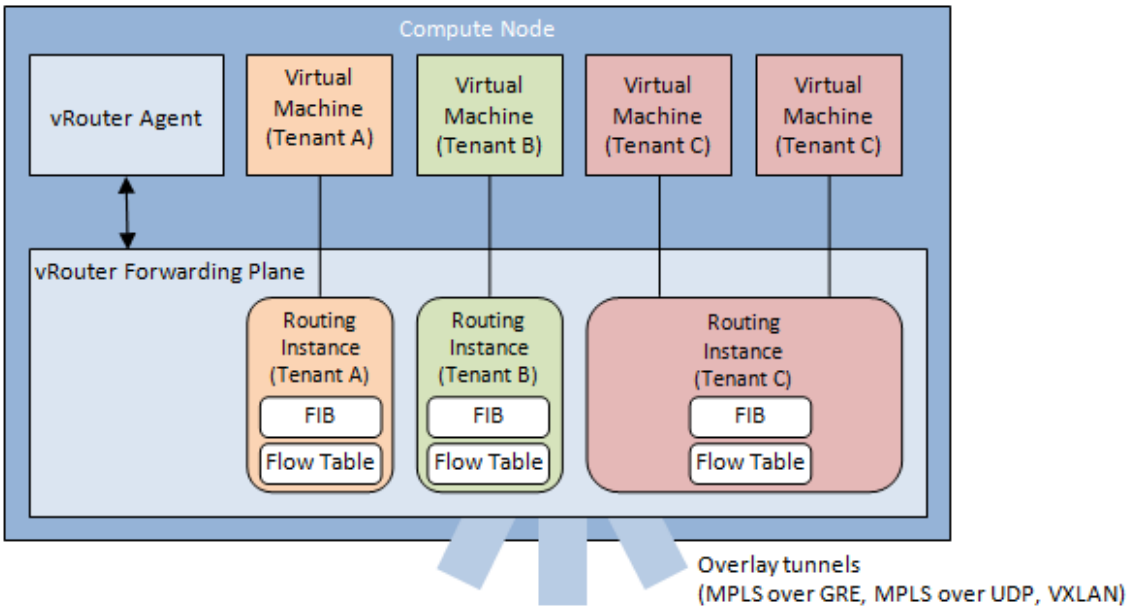


Figure 3: Internal structure of the vRouter forwarding plane (3).

Control Node

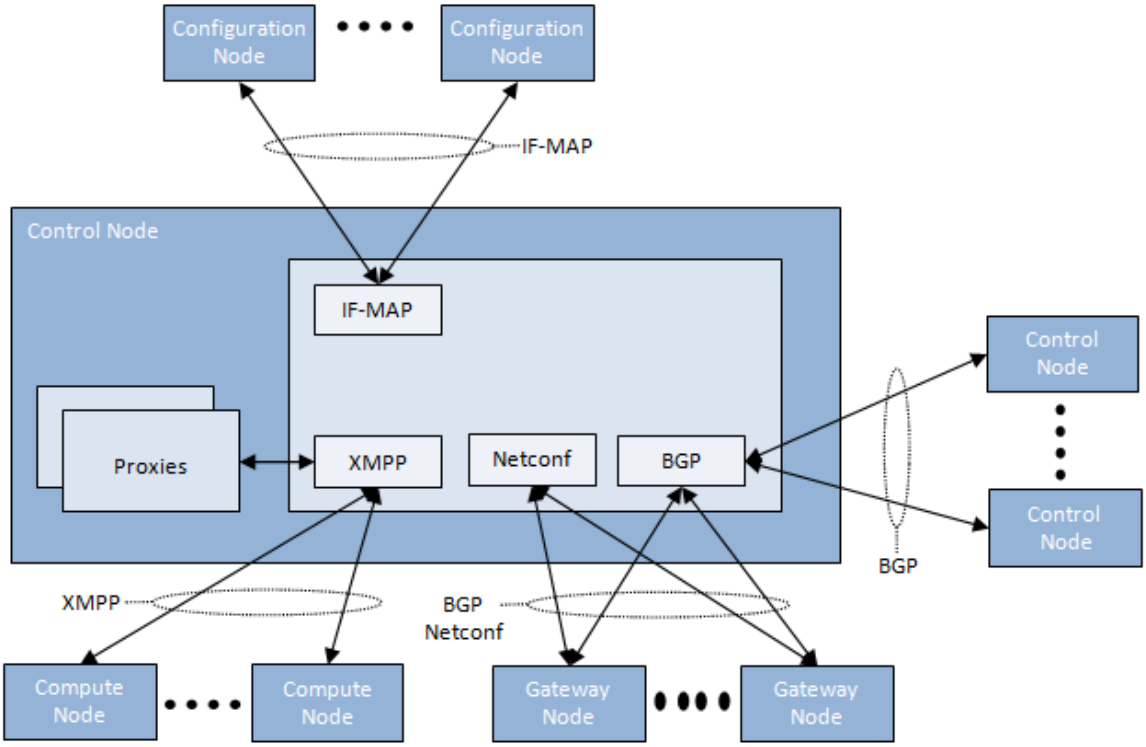


Figure 4: Internal structure of Control Node. (3)

The purpose of this node is to communicate with other several forms of nodes to gather system status information.

- It receives the subset of configuration state in which they have interest via IF-MAP protocol.
- It exchanges routes with other control node via iBGP in order to make sure that entire control nodes have identical network status.
- It uses XMPP to exchange routes from compute node (actually using vRouter agent).
    - It also uses XMPP to send routing instances and forwarding rules (3).
- It, also, exchange routes with gateway nodes (underlay network) with BGP using Netconf protocol.
- Every instance of a control node runs the following processes:
    - control-node
    - contrail-dns
    - contrail-named

Configuration Node

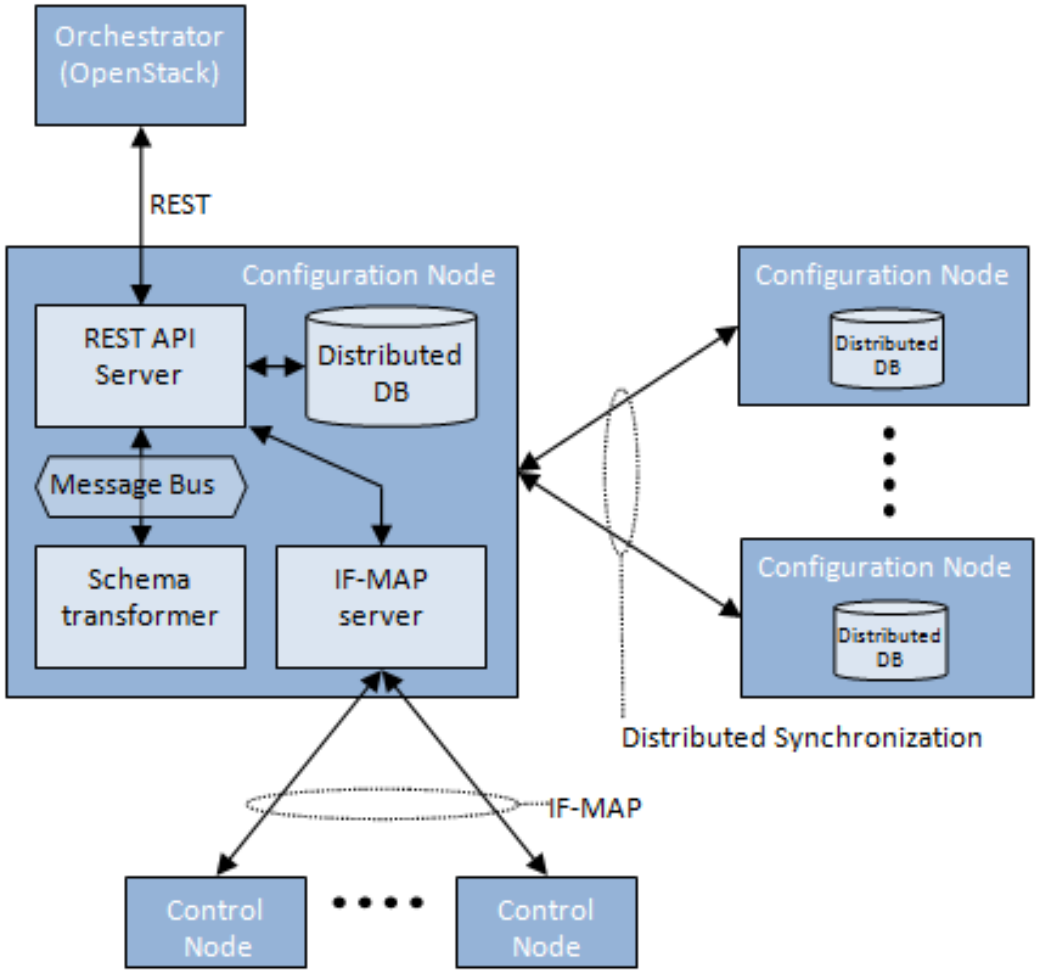


Figure 5: Internal structure of Configuration Node (3).

A REST API living on the configuration node will permit coordination of virtual networks, interfaces, and network policies to control the flow of traffic between virtual networks (3). It has the obligation of recalling and storing the persistent network state. This permits the framework to characterize the desired network state (3).

- Configuration node runs Neutron server, configuration API server, IF-MAP server, discovery server and configuration related services (3).
- The API server, which helps in hand out the emerging condition of schema objects and the data to the IF-MAP server (6) (3).
- The Schema transformer that changes basic input into layered, complex output (3).
- The Service monitor that makes and screens VM instances and implements technologies such as Network address translator, firewalls, WAF, or load balancers (3).

- The Discovery service, which tries to distribute IP address and port data, and the DNS Server, which is a multi-tenant aware DNS server.
- Every instance of a configuration node runs the following processes (10):
    - contrail-discovery.
    - neutron-server.
    - contrail-api.
    - if-map.
    - contrail-schema.
    - contrail-svc-monitor.
    - rabbitmq-server (this can optionally be located on an external server) (10).

Analytics Node

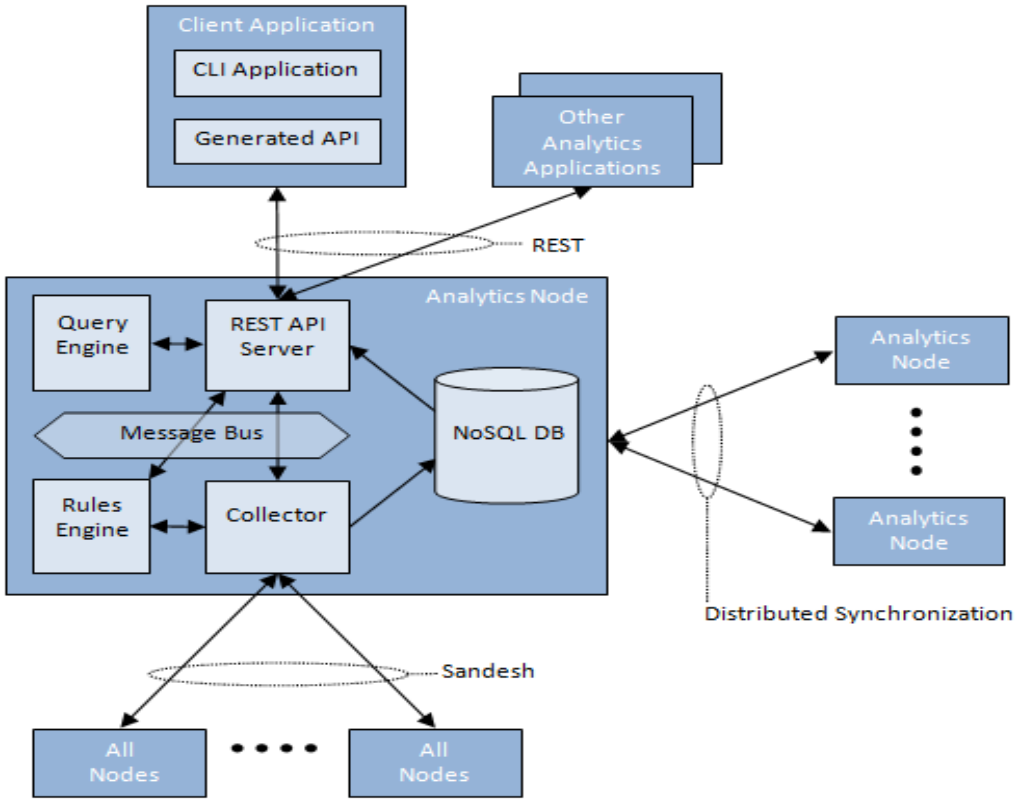


Figure 6: Internal structure of Analytics Node (3).

This node provides REST interface to a series of databases containing the state information of virtual-networks, virtual-machines, configuration, and control nodes using Sandesh protocol (3). This too comprises traffic stream records kept in a distributed NoSQL database.

- A collector exchanges Sandesh messages with modules in control and configuration nodes in order to gather analytics data (3).

- If any specific event triggers, a rules engine automatically gather operational state.
- The purpose of query engine to execute received queries over REST APIs. It can as well handle large amount of analytics data.
    - The majority of OpenContrail queries are time series.
- Every instance of analytics node runs the following processes:
    - contrail-collector
    - contrail-analytics-api
    - contrail-query-engine (query engine)

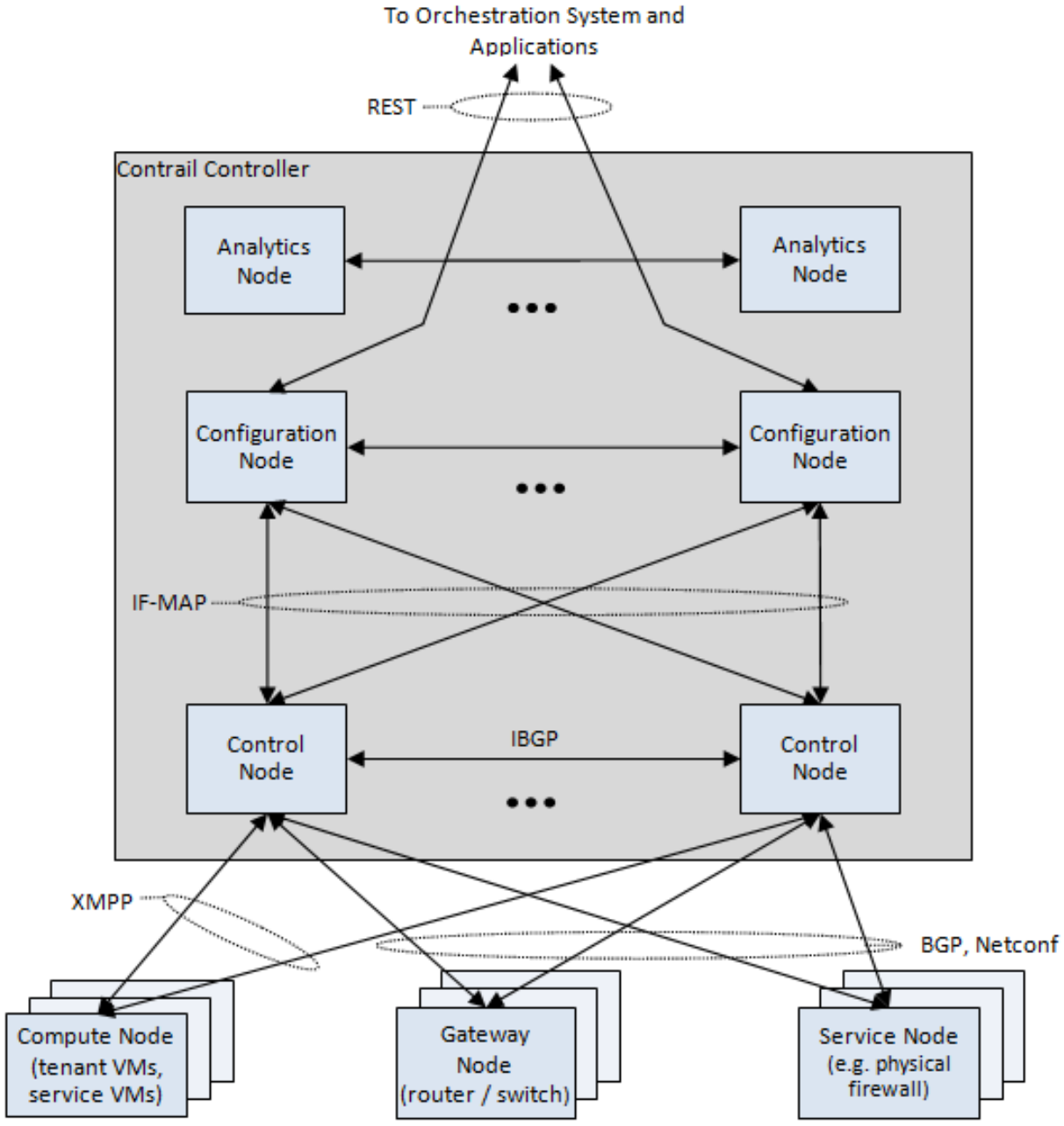


Figure 7: OpenContrail overall implementation (3).

System is implemented as a collaborating set of nodes running on general-purpose x64-86 servers. Each node may be implemented as a separate physical server or it may be implemented as a Virtual Machine (VM) (3). No single node creates a bottleneck in the network because each and every node is running in active-active mode. This type of design provides scalability for both failover and horizontal scalability (3).

The Analytics Node, Configuration Node, and Control Node together make the Contrail Controller. All the administration, data analysis, and control-plane processes happens here.

The data-plane component (otherwise known as vRouter) is available in a Compute Node. The physical network can be connected through Gateway Node in order to connect the tenant to the physical network for accessing internet, VPN, etc. WAN optimizers and load balancers can be connected by Service Node to provide network services.

OpenStack Integration

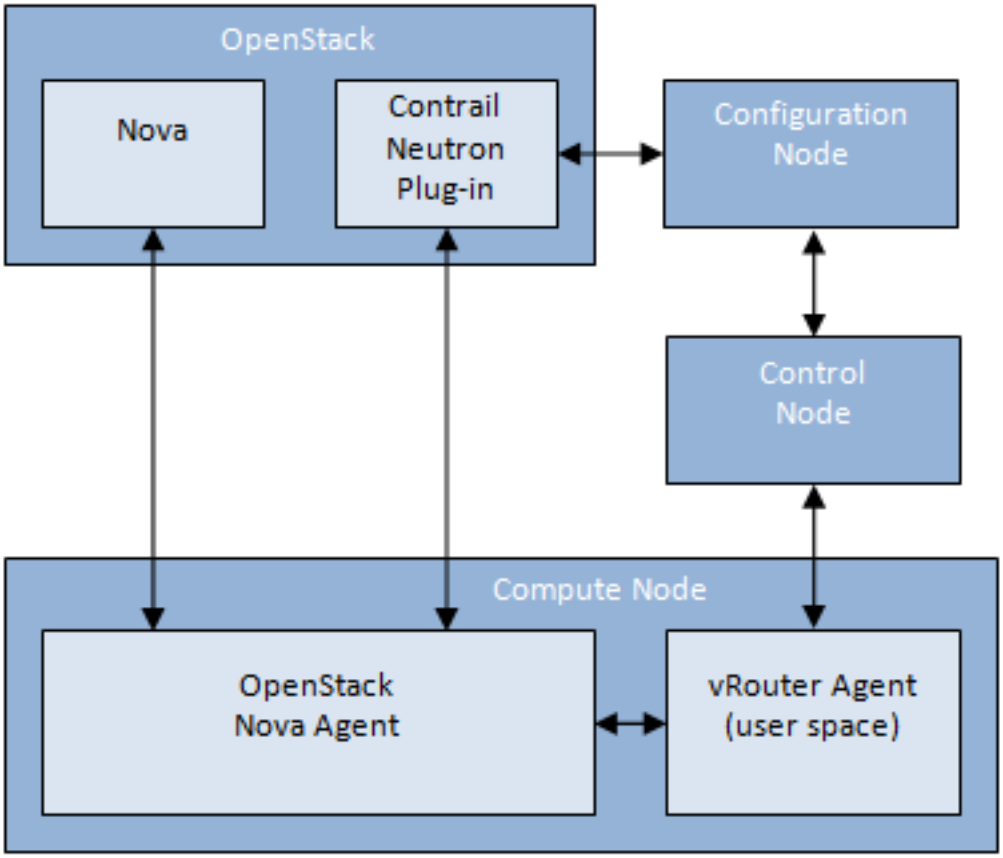


Figure 8: OpenStack integration with OpenContrail (3).

The Nova module in OpenStack collaborates with Nova Agent in the compute node to generate the VM (3). The Nova Agent communicates with the OpenContrail Neutron plug-in in OpenStack to retrieve the network elements of the newly created VM (e.g. the IP address). Once the VM is formed, the Nova Agent in Compute Node informs the vRouter agent who establishes the VN for the freshly created virtual machine (e.g. new routes in the routing-instance) (3).

The OpenStack hooks used by Contrail are

1. core_plugin - This is used in the neutron config to point to Contrail Plugin (10).
2. libvirt_vif_driver - This is used in the nova compute config to point to Contrail VRouter VIFDriver (10).
3. MQ broker IP and Port - corresponding IP and port needs to be configured in the neutron and nova config (10).

Juniper OpenStack is a distribution by Juniper that is used by Juniper Private Cloud, Juniper's NFV Solution for Service Provider Market, OpenContrail build system, and OpenContrail test system (12). In addition to Contrail networking this includes following components:

- Block and Object storage.
- Installation, provisioning and monitoring of cluster (12).
- Contrail extension to OpenStack components such as Horizon, Neutron Client etc.
- High availability for OpenStack (12).
- Other features to ease the deployment and operation of cloud.

Juniper OpenStack is based on Ubuntu cloud archive OpenStack packages with fixes/extension as needed. It is bundled with all the dependent packages and installation scripts. Note that Ubuntu Cloud archive has additional OpenStack packages which may not be qualified for Juniper OpenStack and so will not be part of the release. At this time, CentOS and its variant are not distributed as part of Juniper OpenStack.

Security

Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) are being used to provide authentication, and reliability for communication (3).

- Initially, for authentication service discovery certificates being used.
- Later transmission uses token-based authentication.
    - The service discovery server put forth the tokens to both the servers and the clients over certificate authenticated TLS connections (3).
- All REST APIs in the system use role-based authorization (3).
    - The roles determine which objects in the data model the client is allowed to access (3).

Implementation of Virtual Network

Installation

I have used Ubuntu Server 12.04 on Dell Servers to install OpenContrail Controller. The Ubuntu is installed as a bare metal and on top of that Contrail packages installed then provisioning scripts are run that launch role-based components of the software.

The roles used for the installed system include:

- **cfgm**—Runs Contrail configuration manager (config-node)
- **collector**—Runs monitoring and analytics services
- **compute**—Runs vRouter service and launches tenant virtual machines (VMs)
- **control**—Runs the control plane service
- **database**—Runs analytics and configuration database services
- **openstack**—Runs OpenStack services such as Nova, Quantum, and the like
- **webui**—Runs the administrator web-based user interface service

The above mentioned roles can run on a single server for testing purposes.

Hardware Specifications

The below are minimum specifications for the server in the Contrail system.

- 16 GB memory
- 500 GB hard drive
- 4 CPU cores
- 1 Ethernet port

Downloading Installation Package

I have downloaded the contrail package from
http://www.juniper.net/support/downloads/?p=contrail#sw

Configuring Server Settings

After installing the base image to server, and before running role provisioning scripts, do the following steps to configure items specific to your environment.

1. On Ubuntu and Debian Linux, you can edit the head file, which is prepended to resolv.conf on boot: vi /etc/resolvconf/resolv.conf.d/head

```
root@contrail:~# cat /etc/resolv.conf
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 10.3.31.10
nameserver 129.128.5.233
search dc1
root@contrail:~#
```

2. Update /etc/network/interfaces with the hostname and domain information specific to your system.
    a. You can also do it at the time of installation of Ubuntu Server 12.04.
    b. Make sure you are using static IP Address binding otherwise the installation will be failed.

Installing the Contrail Packages

Using Ubuntu

1. Copy the downloaded Contrail install package file to /tmp directory on the config node.
2. I used WinSCP to copy the Contrail packages to /tmp.
3. Install the package by running the command: dpkg -i /tmp/contrail-install-packages-1.xx-xxx~openstack_version_all.deb (xxx means the version number of Contrail package).
4. Run the **setup.sh** script. cd /opt/contrail/contrail_packages; ./setup.sh
5. Once setup is done than edit the testbed.py file for role provisioning. Location is /opt/contrail/utils/fabfile/testbeds/testbed_singlebox_example.py for a single server system.

```
from fabric.api import env

#Management ip addresses of hosts in the cluster
host1 = 'root@10.3.31.40'

#External routers if any
#for eg.
#ext_routers = [('mx1', '10.204.216.253')]
ext_routers = [('mx1', '10.3.31.31')]

#Autonomous system number
router_asn = 64512

#Host from which the fab commands are triggered to install and provision
host_build = 'root@10.3.31.40'

#Role definition of the hosts.
env.roledefs = {
    'all': [host1],
    'cfgm': [host1],
    'openstack': [host1],
    'control': [host1],
    'compute': [host1],
    'collector': [host1],
    'webui': [host1],
    'database': [host1],
    'build': [host_build],
    'storage-master': [host1],
    'storage-compute': [host1],
}

#Openstack admin password
env.openstack_admin_password = 'secret123'

#Hostnames
env.hostnames = {
    'all': ['contrail.dc1']
}

env.password = 'secret'
#Passwords of each host
env.passwords = {
    host1: 'secret',

    host_build: 'secret',
}

#For reimage purpose
env.ostypes = {
    host1:'ubuntu',
}
```

Figure 9: Testbed.py file that I have used.

6. Contrail requires the Kernel version for Ubuntu systems should be 3.13.0-34, and you have to upgrade it if you are using Ubuntu Server edition 12.04.3 LTS.

a. You can do so by running cd /opt/contrail/utils; fab upgrade_kernel_all, and this will reboot the server as well.

7. Now, install the Openstack Icehouse by running this command, apt-get install nova-compute-libvirt=1:2014.1-0ubuntu1~cloud0
8. The last step is to install the packages and provision the server.
    a. cd /opt/contrail/utils; fab install_contrail
    b. cd /opt/contrail/utils; fab setup_all - This will reboot the machine

At this time you have fully functional OpenContrail system and Openstack.

You can access horizon web UI http://server-ip/horizon
You can access OpenContrail web UI http://server-ip:8080

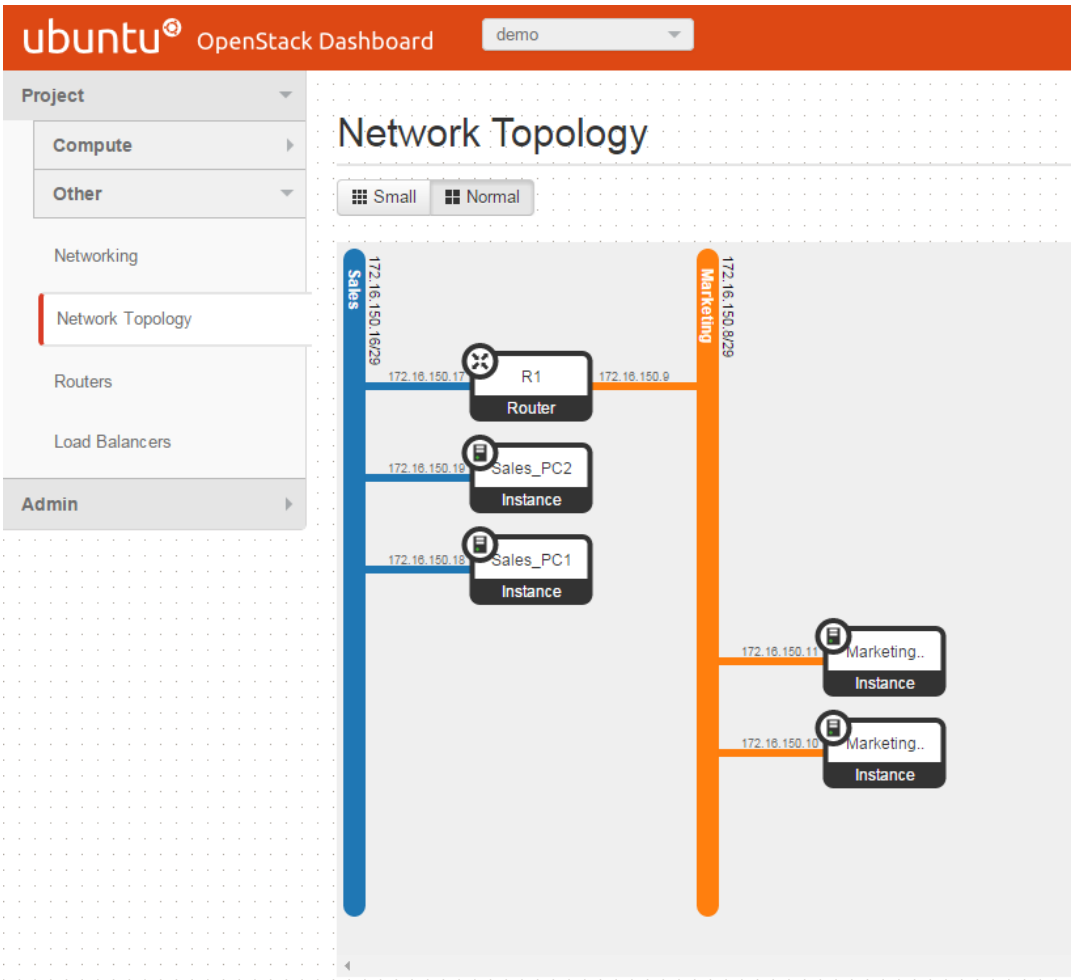
Once you are up and running, you can verify the health of OpenContrail system by running the command, contrail-status on your server.

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

== Contrail Control ==
supervisor-control:             active
contrail-control                active
contrail-control-nodemgr        active
contrail-dns                    active
contrail-named                  active

== Contrail Analytics ==
supervisor-analytics:           active
contrail-analytics-api          active
contrail-analytics-nodemgr      active
contrail-collector              active
contrail-query-engine           active

== Contrail Config ==
supervisor-config:              active
contrail-api:0                  active
contrail-config-nodemgr         active
contrail-discovery:0            active
contrail-schema                 active
contrail-svc-monitor            active
ifmap                           active

== Contrail Web UI ==
supervisor-webui:               active
contrail-webui                  active
contrail-webui-middleware       active
redis-webui                     active

== Contrail Database ==
supervisord-contrail-database:active
contrail-database               active
contrail-database-nodemgr       active

== Contrail Support Services ==
supervisor-support-service:     active
rabbitmq-server                 active
```

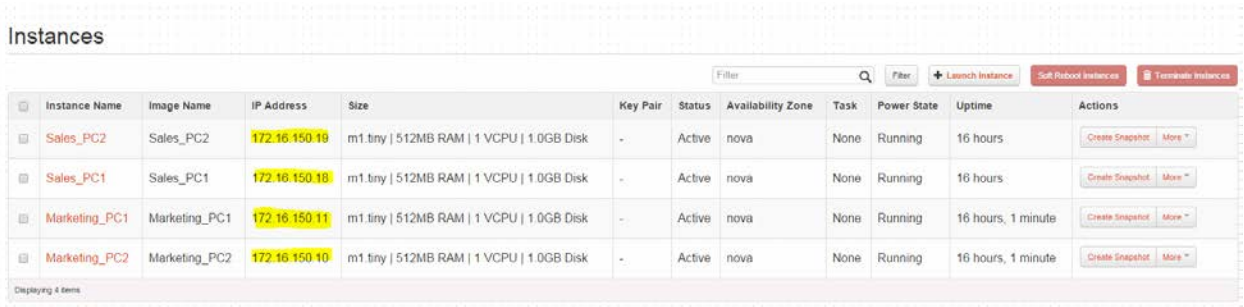
Executing a Scenario

As an example, I have created three different virtual networks for different departments working for same company – Marketing-VN, Accounting-VN, and Sales-VN. They are having 172.16.150.8/29, 172.16.150.16/29, and 172.16.150.24/29 networks respectively. Each VN contains few virtual machines spawned on Marketing, Sales, and Accounting VNs. Once these machines were up and get an IP address via DHCP I will do a ping in the same VN and also between VMs in different VNs

Below are the steps showing how to deploy the above scenario using OpenStack and OpenContrail.


ubuntu OpenStack Dashboard
demo
Project
Compute
Other
Networking
Network Topology
Routers
Load Balancers
Admin
Network Topology
Small
Normal
Sales
172.16.150.16/29
Marketing
172.16.150.8/29
172.16.150.17
R1
Router
172.16.150.9
172.16.150.19
Sales_PC2
Instance
172.16.150.18
Sales_PC1
Instance
172.16.150.11
Marketing..
Instance
172.16.150.10
Marketing..
Instance

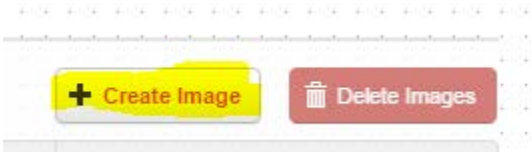

Images
Image Name
Type
Status
Public
Protected
Format
Actions
Sales_PC2
Image
Active
No
No
QCOW2
Sales_PC1
Image
Active
No
No
QCOW2
Marketing_PC2
Image
Active
No
No
QCOW2
Marketing_PC1
Image
Active
No
No
QCOW2


Instances
Instance Name
Image Name
IP Address
Size
Key Pair
Status
Availability Zone
Task
Power State
Uptime
Actions
Sales_PC2
Sales_PC2
172.16.150.19
m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk
Active
nova
None
Running
16 hours
Sales_PC1
Sales_PC1
172.16.150.18
m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk
Active
nova
None
Running
16 hours
Marketing_PC1
Marketing_PC1
172.16.150.11
m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk
Active
nova
None
Running
16 hours, 1 minute
Marketing_PC2
Marketing_PC2
172.16.150.10
m1.tiny | 512MB RAM | 1 VCPU | 1.0GB Disk
Active
nova
None
Running
16 hours, 1 minute

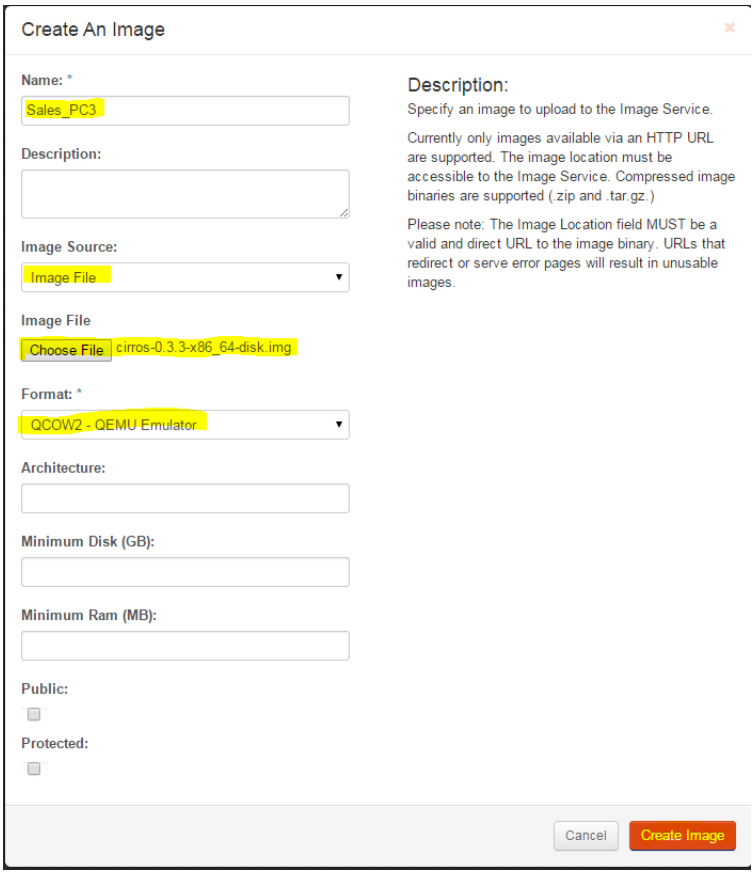
Above screen shots are taken from OpenStack which is being used for creating VMs and storing images. Instances of these machines and have been launched and they did get their IP addresses successfully from their respective networks.

Process of attaching VMs to the network:

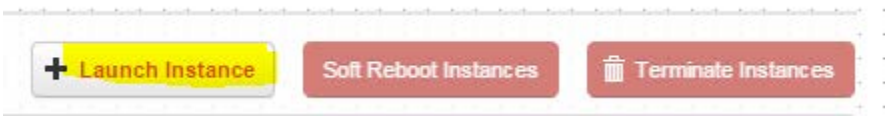
Under Compute →go to Images→on the right side→Click on Create Image

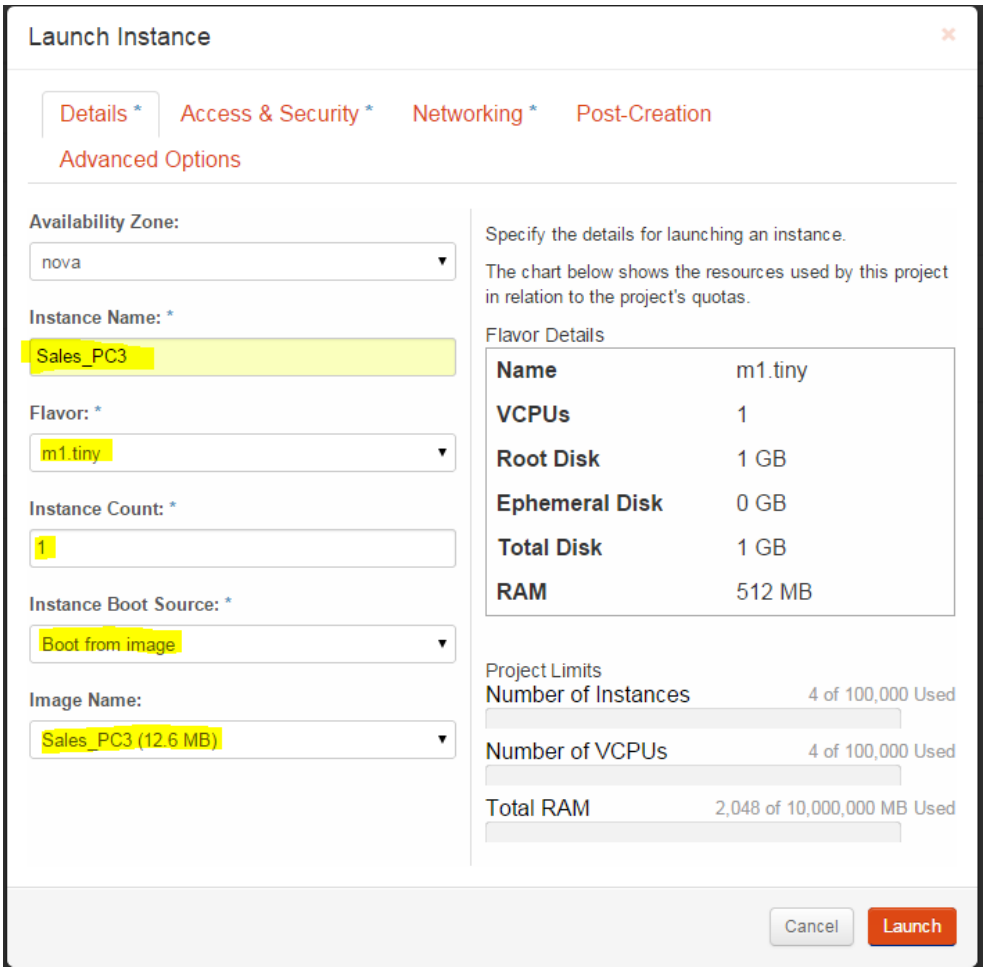


I am using CirrOS (test) images – and the most recent 64-bit qcow2 image as of this writing is cirros-0.3.3-x86_64-disk.img. It only takes 12.6MB of the space and gives you good base functionality to test the network.

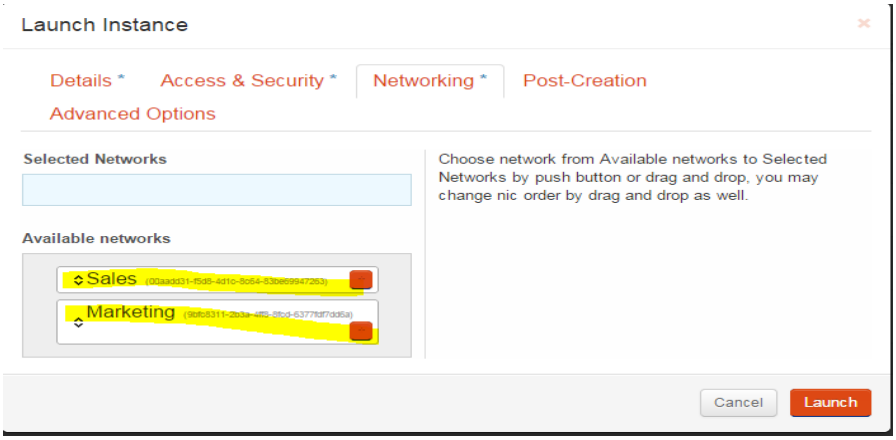


After clicking on Create Image, OpenStack will give you success notice. Then go to Instances→and click on Launch Instance

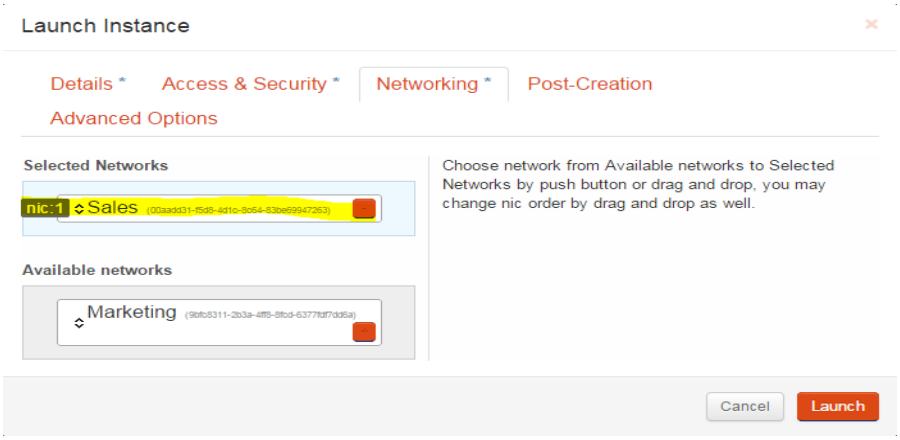




After filling out the information go to Networking tab.

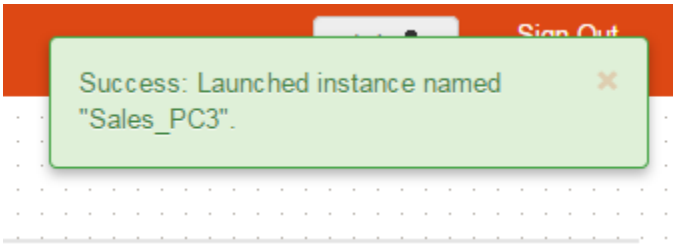


You will see Available networks, and you can drag and drop the one you required to connect to Selected network.

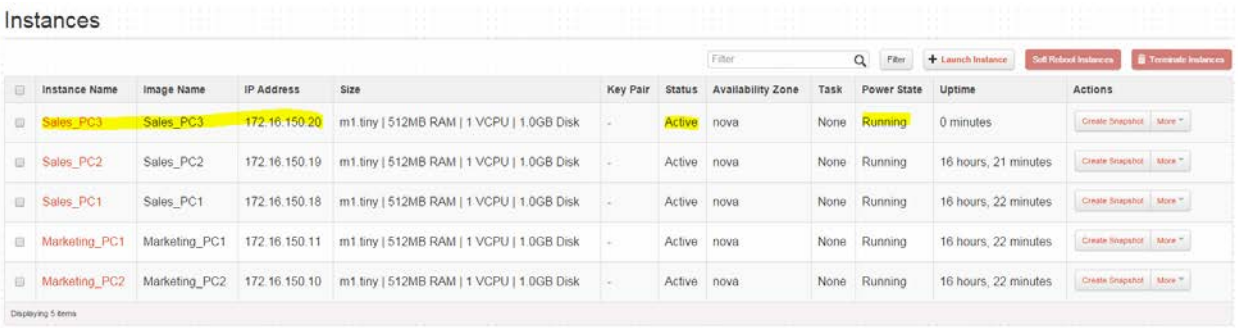


In Selected Networks – your Sales network will be connected to nic1. Click on launch.

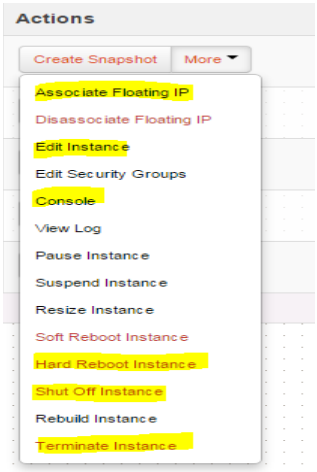
After successful launching the instance you will see a notification like this:



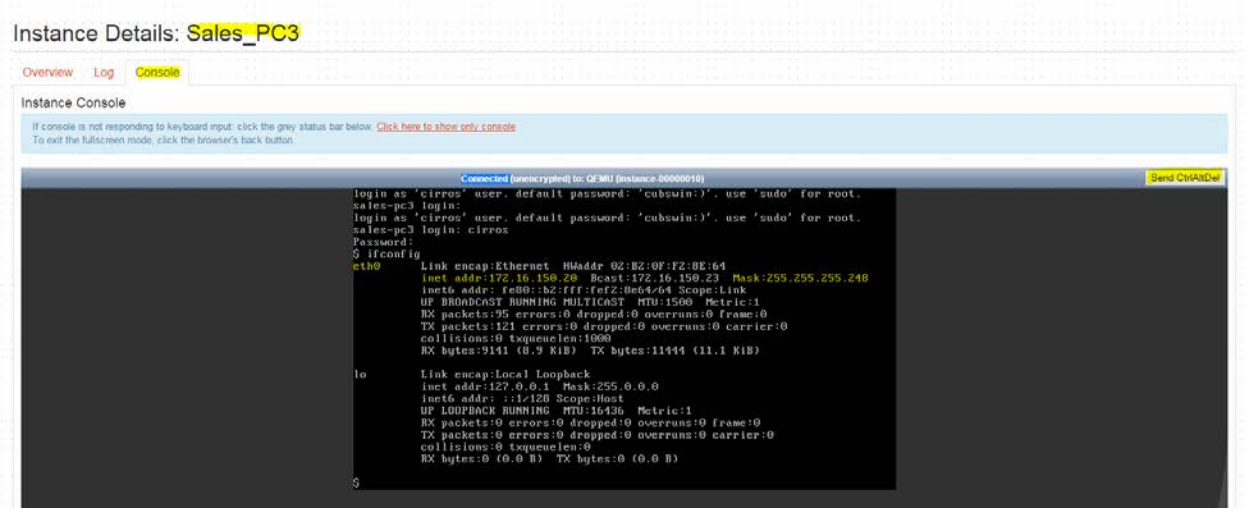
In the below figure, you will see that Sales_PC3 get the IP Address from the Sales subnet, status is Active, Power Status is running.



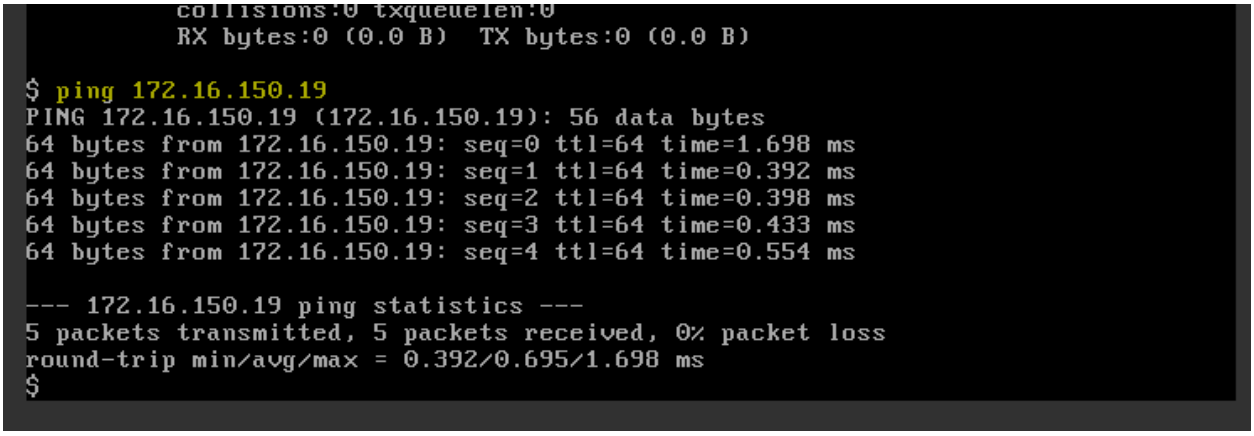
If you click on any VM More option – you will see different options to perform, such as, accessing Console, Reboot VM, etc.



See the console IP configuration by ifconfig command.



Successfully Ping to VM in same subnet



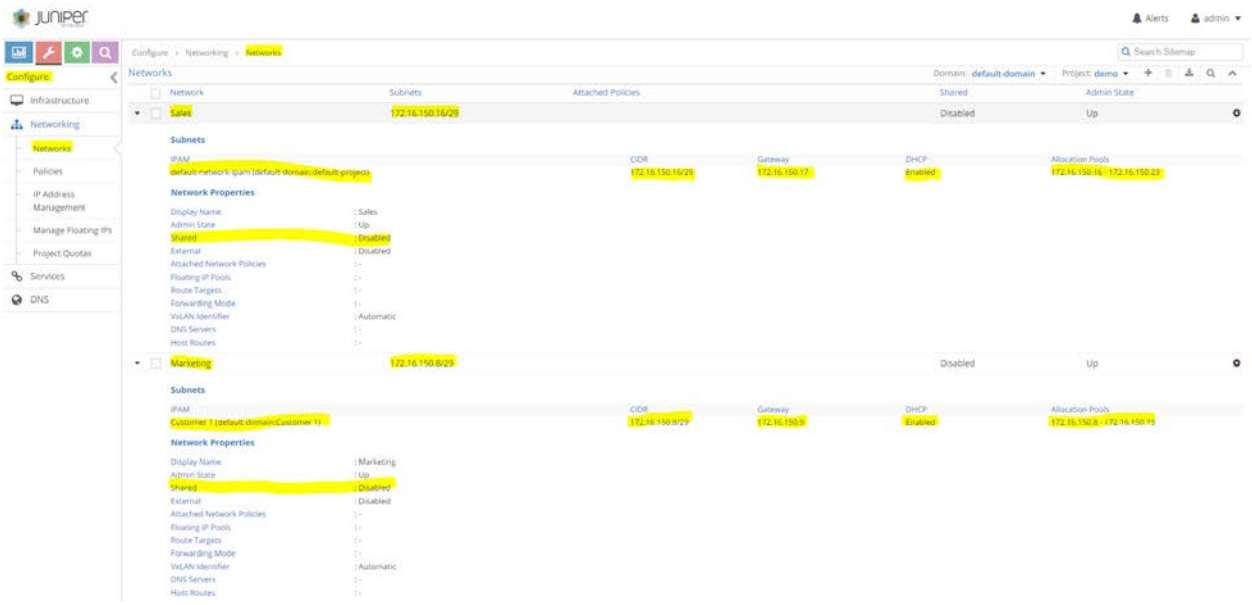
Successfully Ping to VM in different subnet

```
$ ping 172.16.150.11
PING 172.16.150.11 (172.16.150.11): 56 data bytes
64 bytes from 172.16.150.11: seq=0 ttl=64 time=2.063 ms
64 bytes from 172.16.150.11: seq=1 ttl=64 time=0.449 ms
64 bytes from 172.16.150.11: seq=2 ttl=64 time=0.386 ms
64 bytes from 172.16.150.11: seq=3 ttl=64 time=0.433 ms

--- 172.16.150.11 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.386/0.832/2.063 ms
$ _
```

Now I will show you OpenContrail where all this networking occurs.

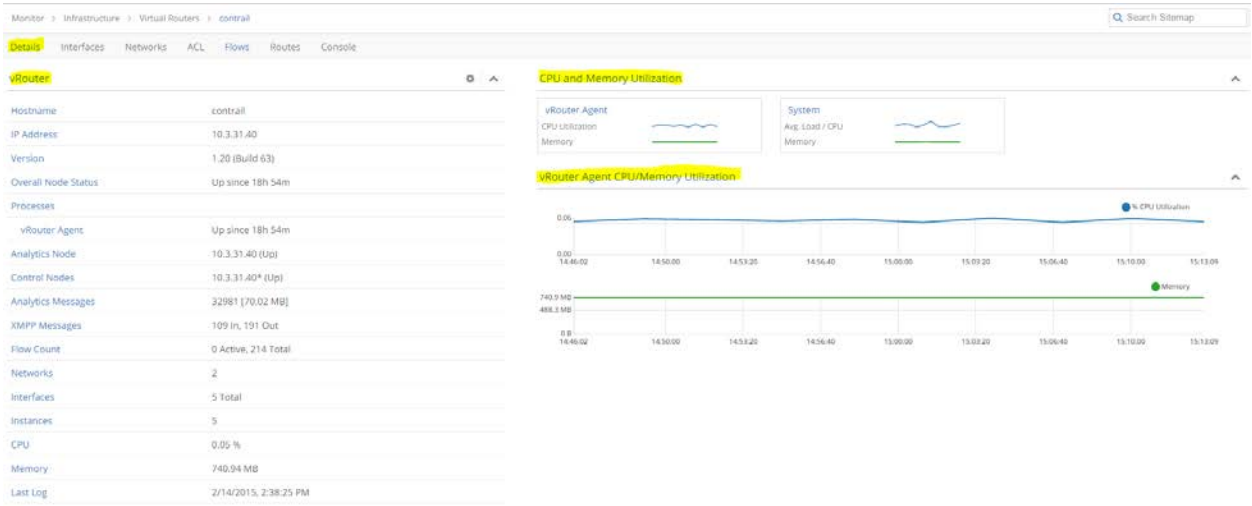
This is the main of the Juniper Contrail after login – dashboard shows you the quick look of the networks, Instances, and Virtual Networks running. It also shows you the memory and CPU consumption.

You see two virtual networks below, where CIDR, IPAM, Gateway, DHCP, Allocation Pools, and other different settings.

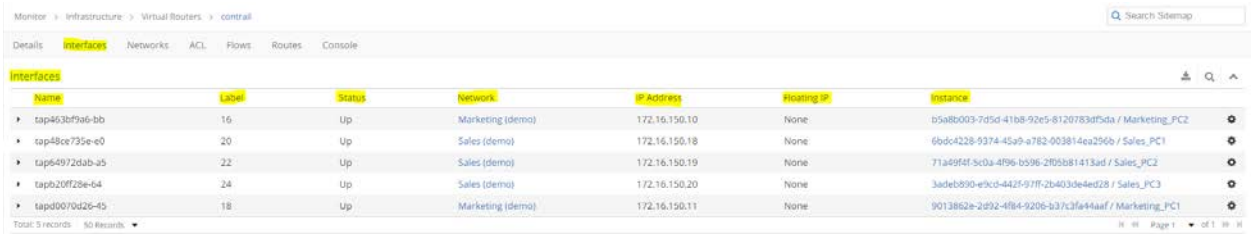


Now, go to →Monitor →Infrastructure →Virtual Routers →contrail, you will see different options to monitor.

This is the quick look of vRouter detail pane.



Now, click on Interface tab, here you see interfaces details to the attached network.

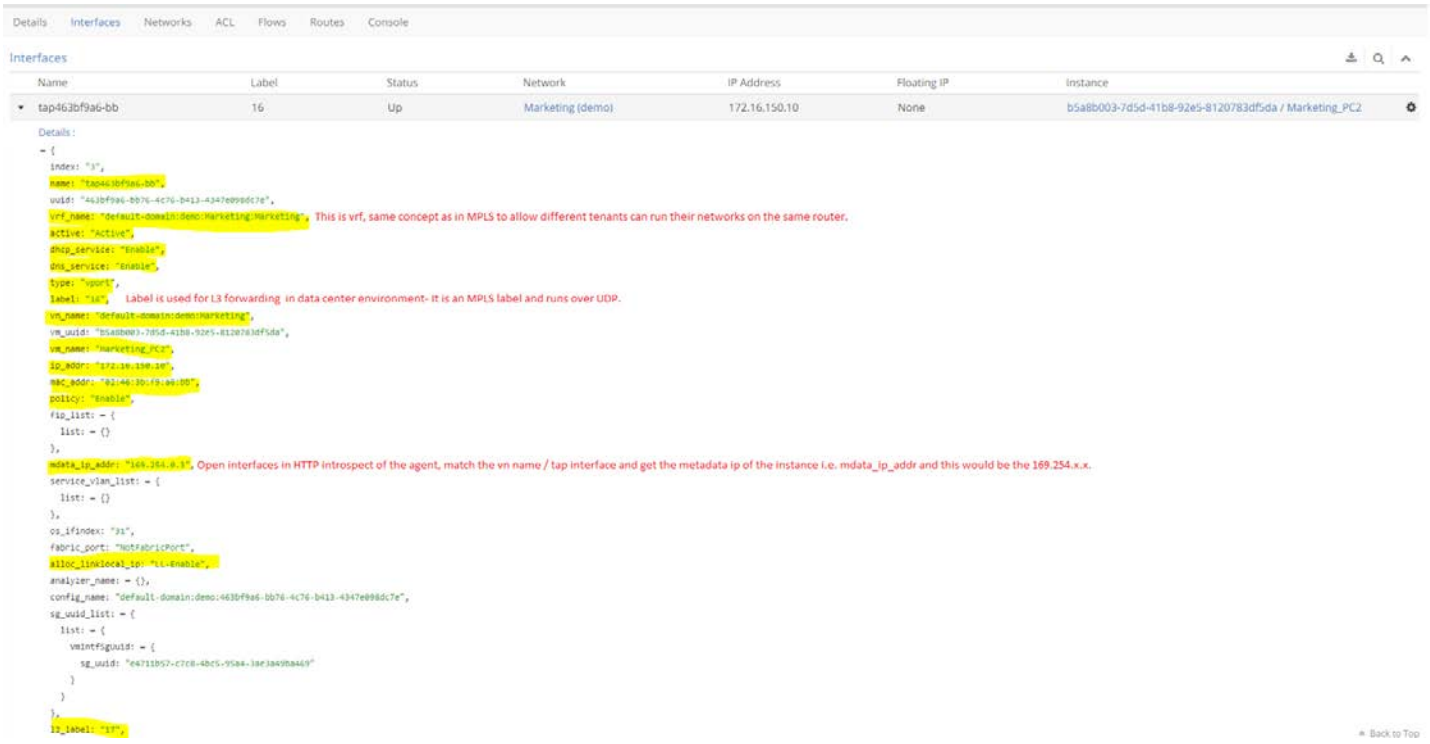


And if you click on the arrow of any interface Name, it will show you the code snippets.

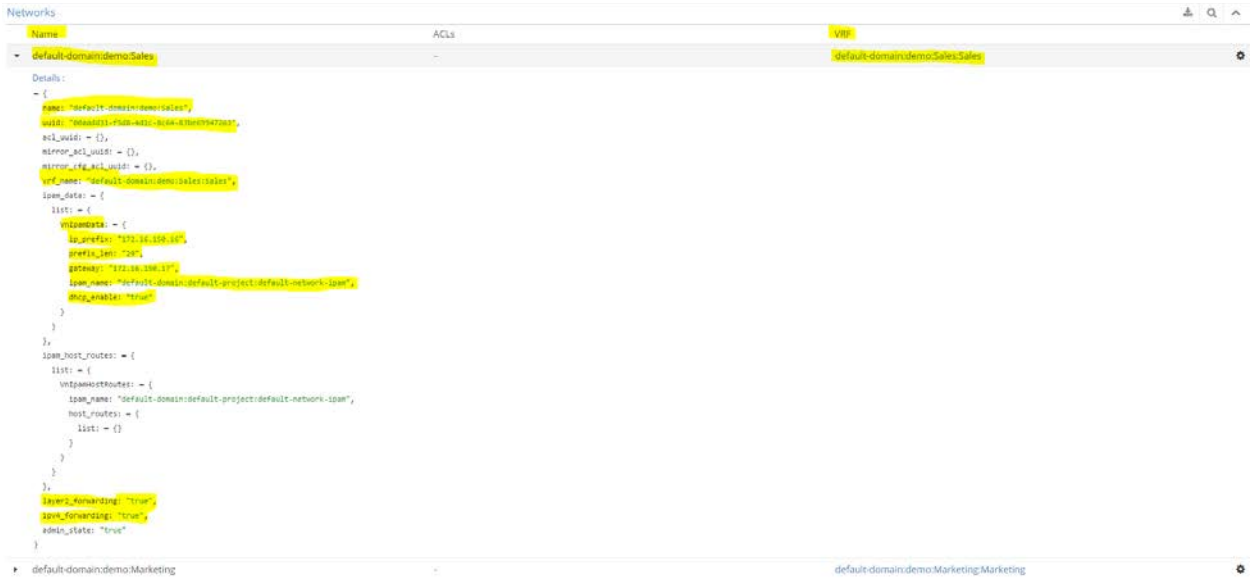
Before going further, I would like to introduce the concept of Metadata Service as this concept is heavily used in Contrail and OpenStack.

OpenStack allows VMs to access metadata by distributing a HTTP request to the link-local address 169.254.x.x. The metadata request from the VM is proxies to Nova, with additional HTTP header fields added. Nova uses these to identify the source instance and responds with appropriate metadata. Metadata IPs is also assigned to the VMs on the compute node.

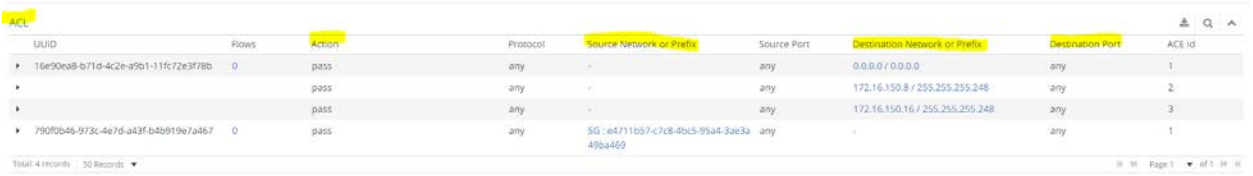
Contrail vRouter acts as the proxy, catching the metadata requests, adding the essential header fields and transfer the requests to the Nova API server.



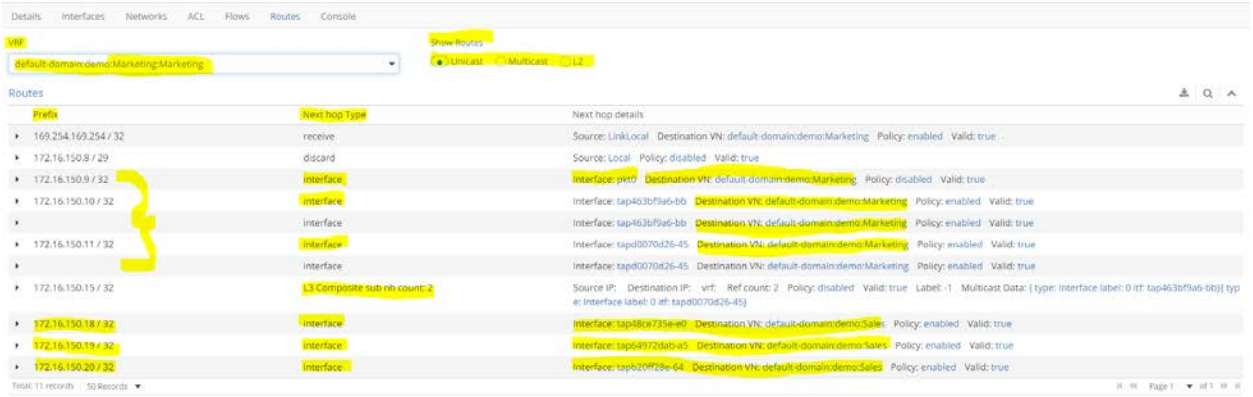
Now go to Networks tab.



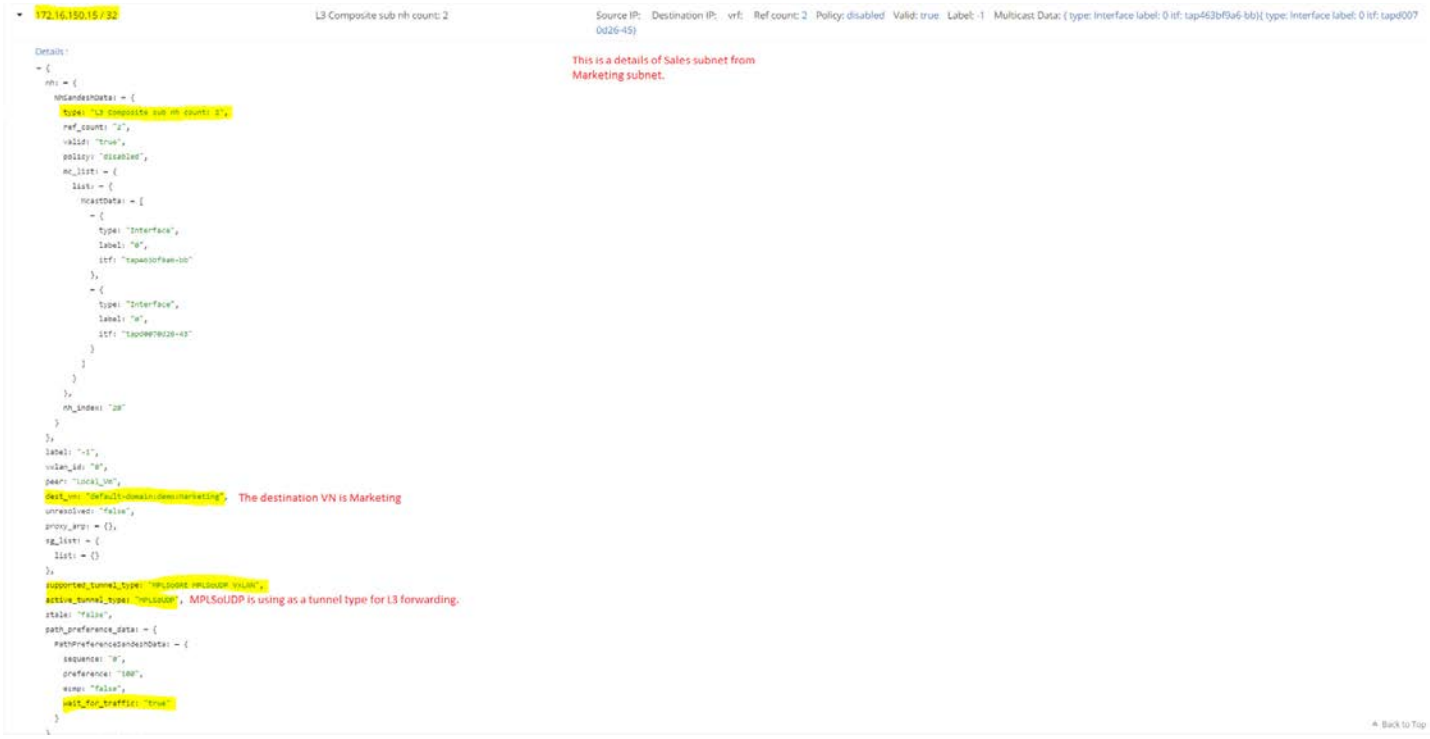
Now go to ACL, where I am using the default behavior.



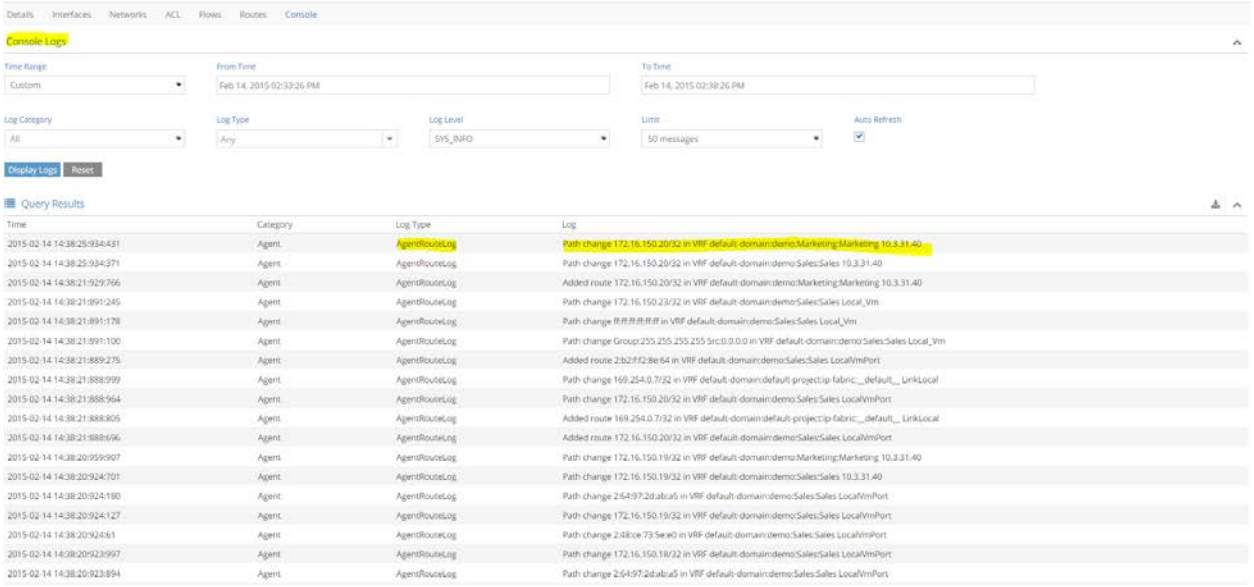
Now go to Routes, and see the details of all routes in specific VRF.



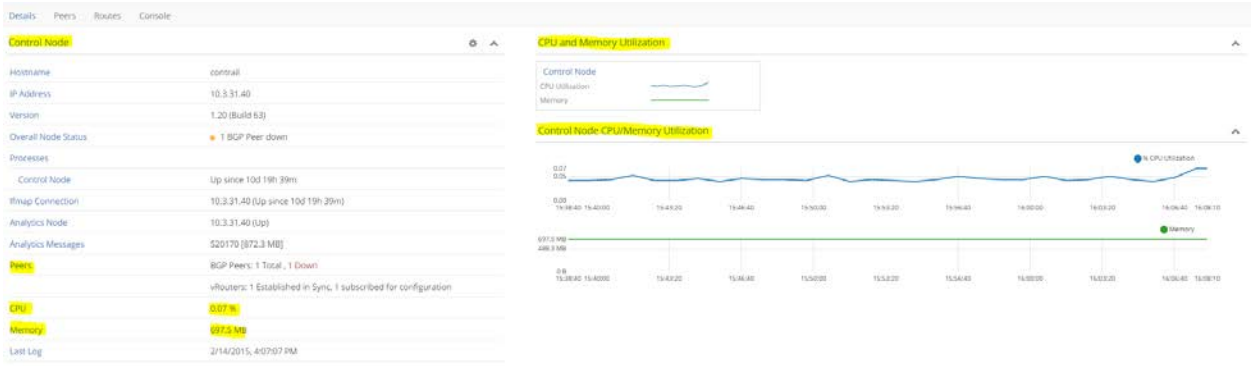
You can click on arrow to see the details of the route.



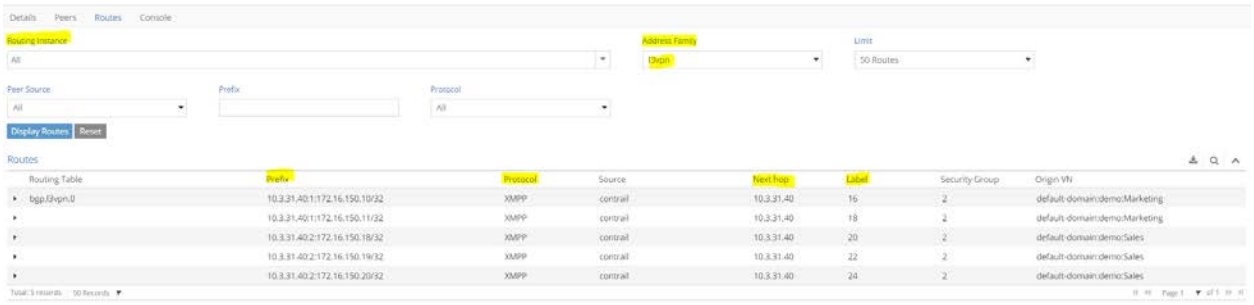
You can also see the console logs.



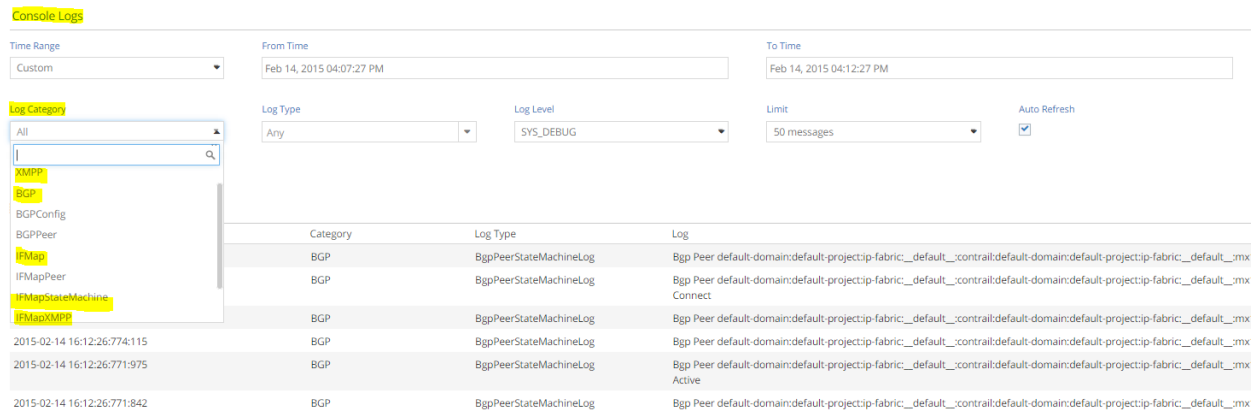
Control Node



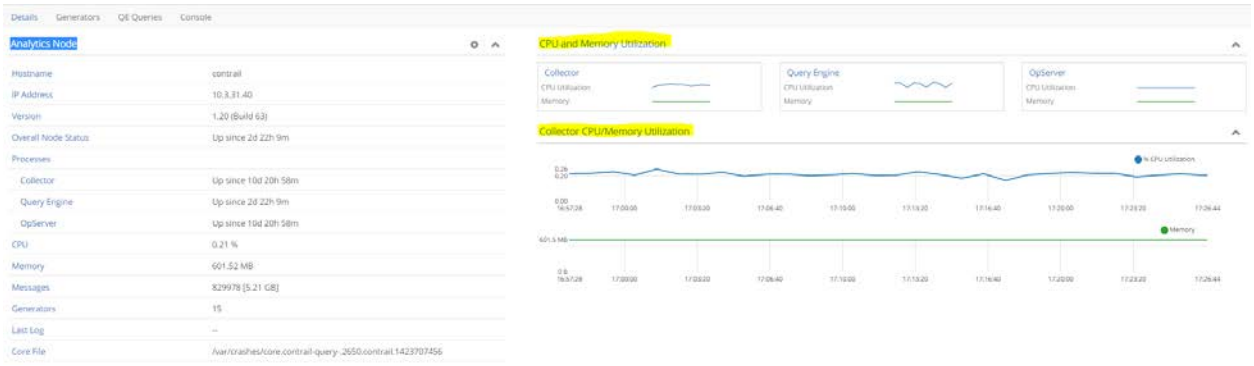
Control Nodes routes



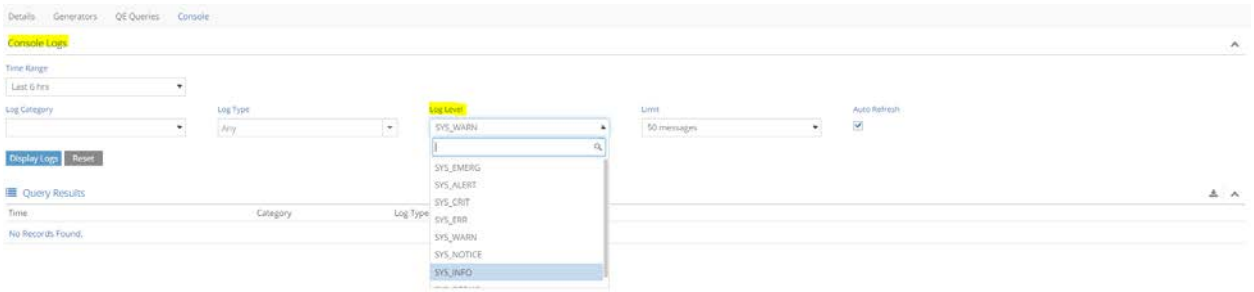
You can also see and select different console logs for Control Nodes.



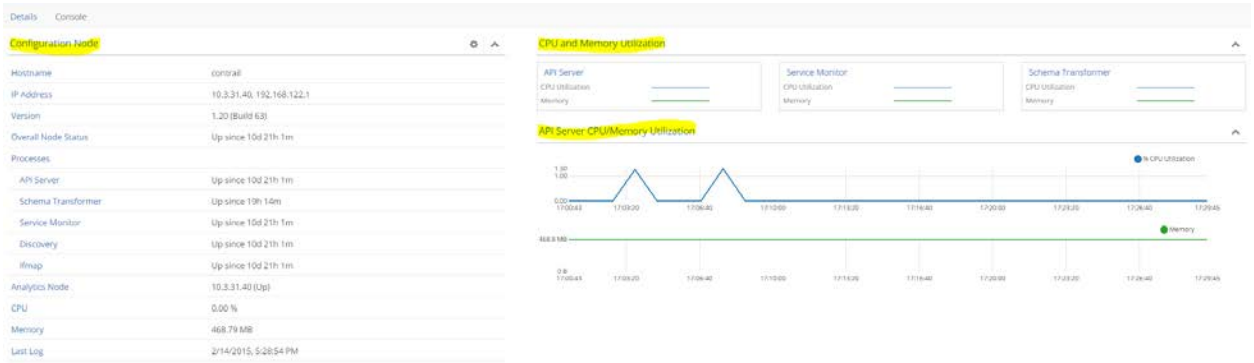
Analytics Node



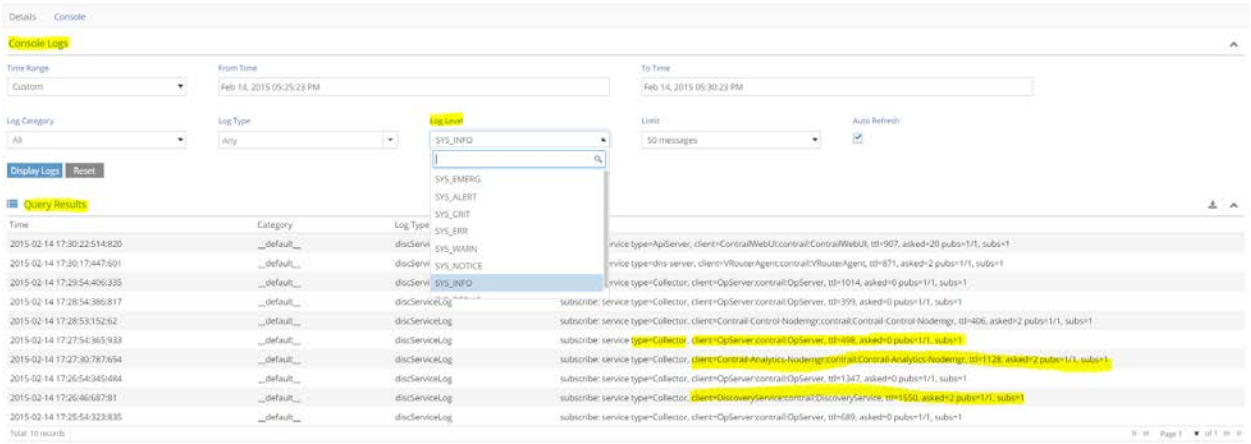
You can check the detail logs of Analytics node by changing the log level.



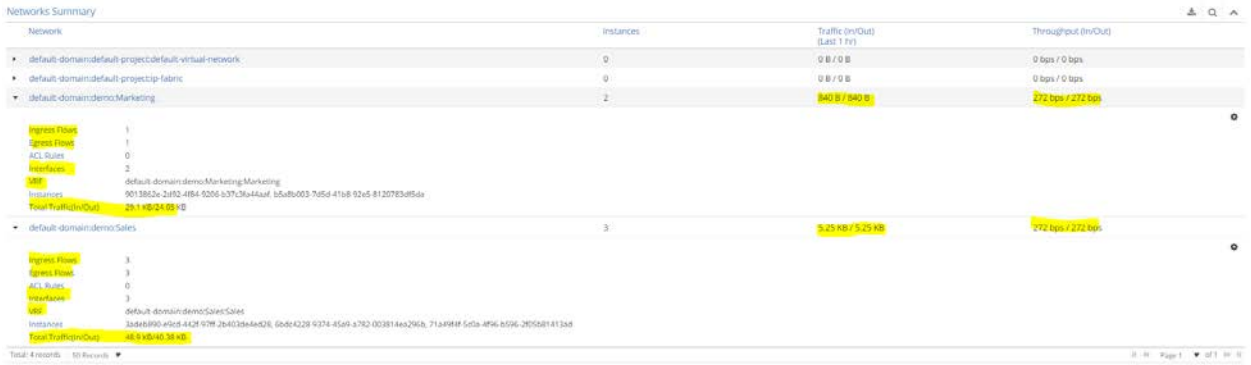
Configuration Node



You can also check the console logs of Configuration Node by changing the log levels.

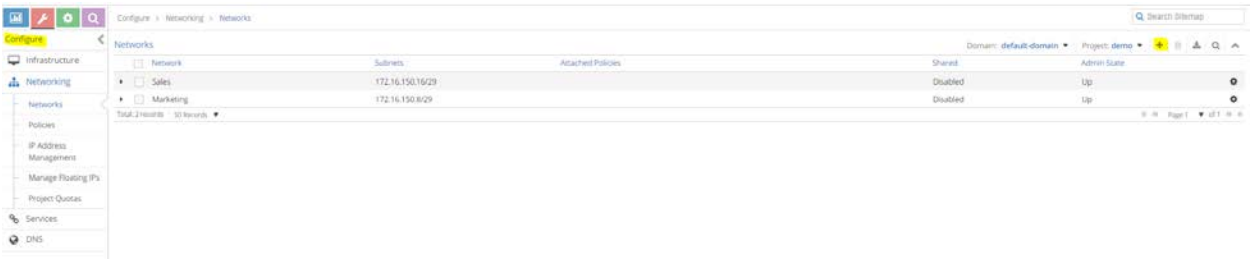


Below is the Virtual Network Traffic summary.

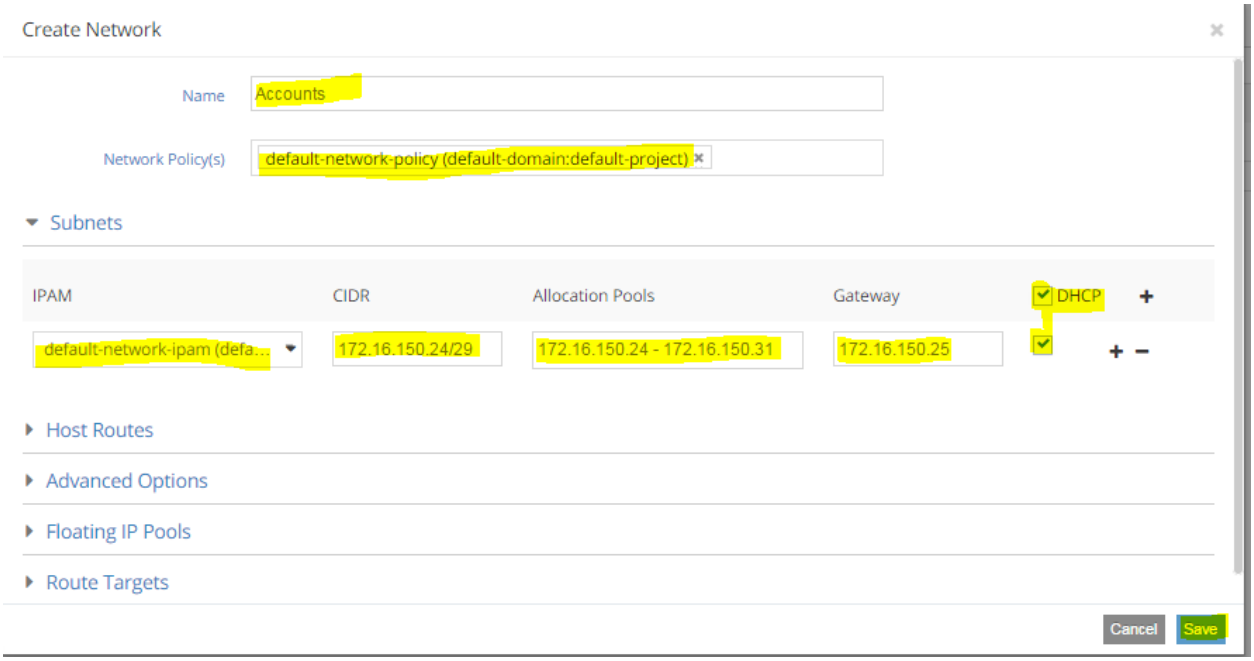


Below is the Instances summary.

To create a virtual network in OpenContrail, go to Configure and click on + sign to create VN.

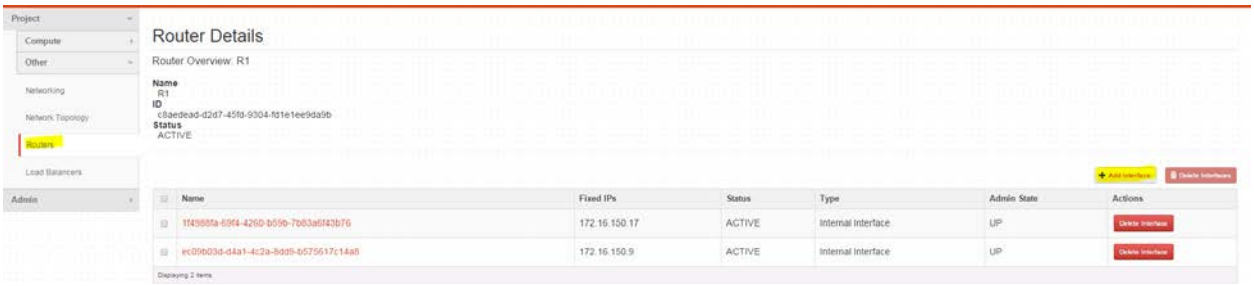


Fill the appropriate fields and click on save.



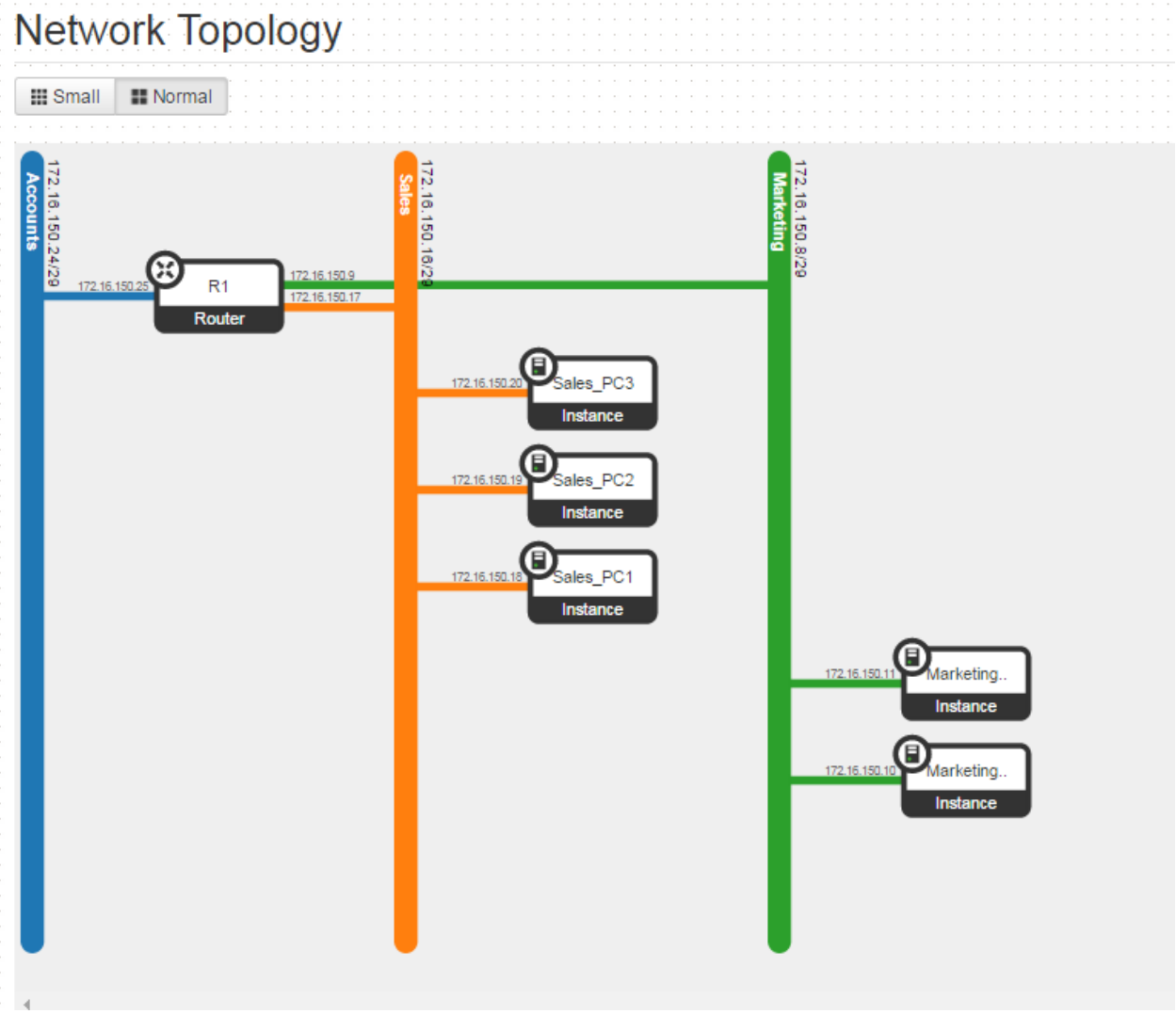
Creating a new Virtual Network

Now go back to OpenStack and attach the new VN to the router. Click on Add interface.

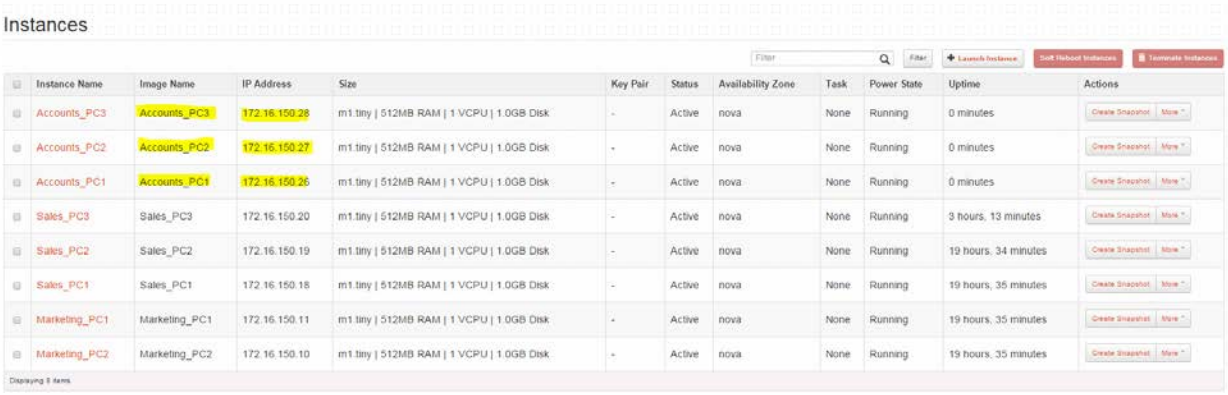


Select the subnet which you want to attach to the network and click on Add Interface,

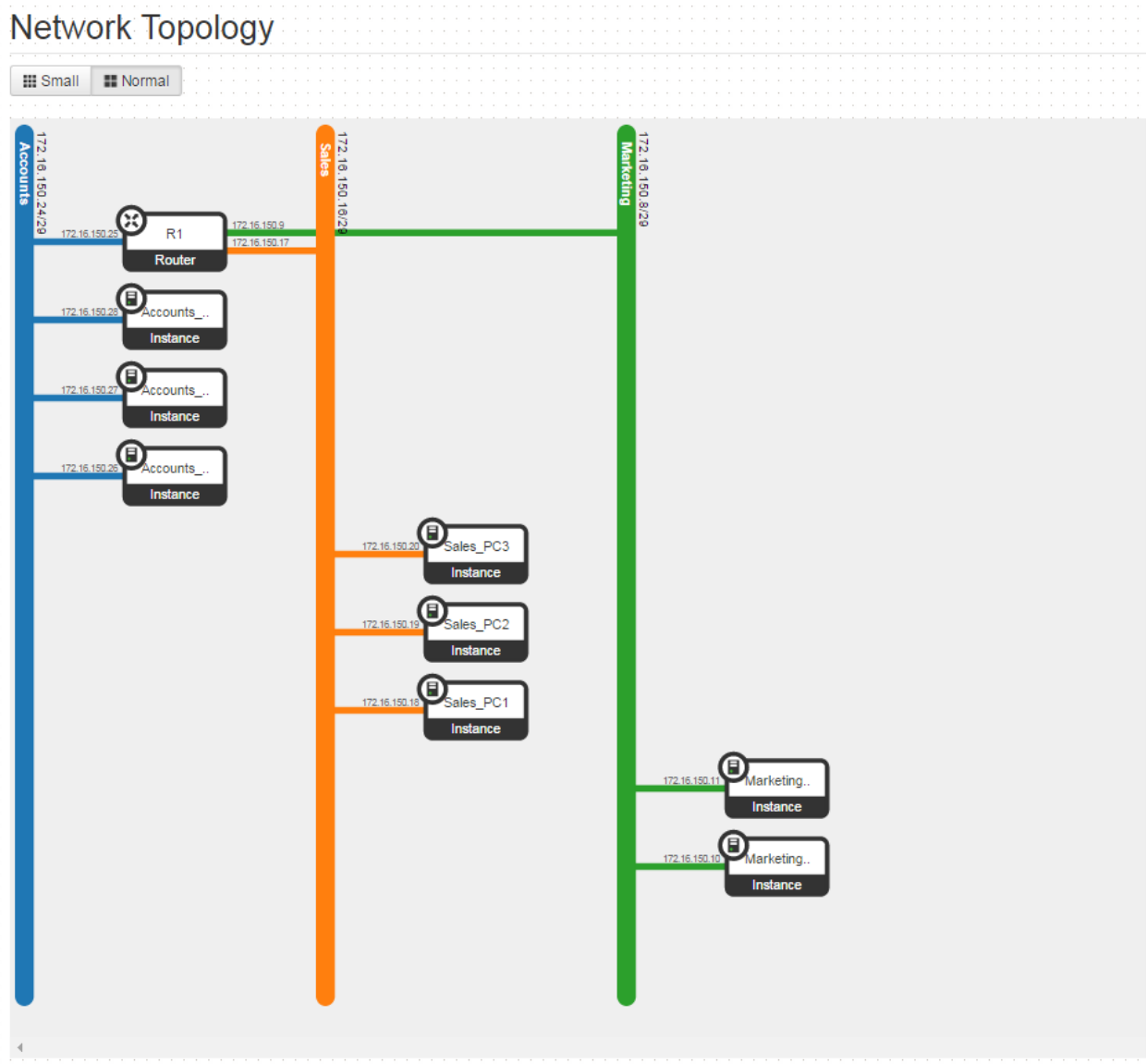
Go to Network Topology to verify if the network is connected to router.



Now create the Images and launch the instances in the Account network. See above for details on how to configure this.

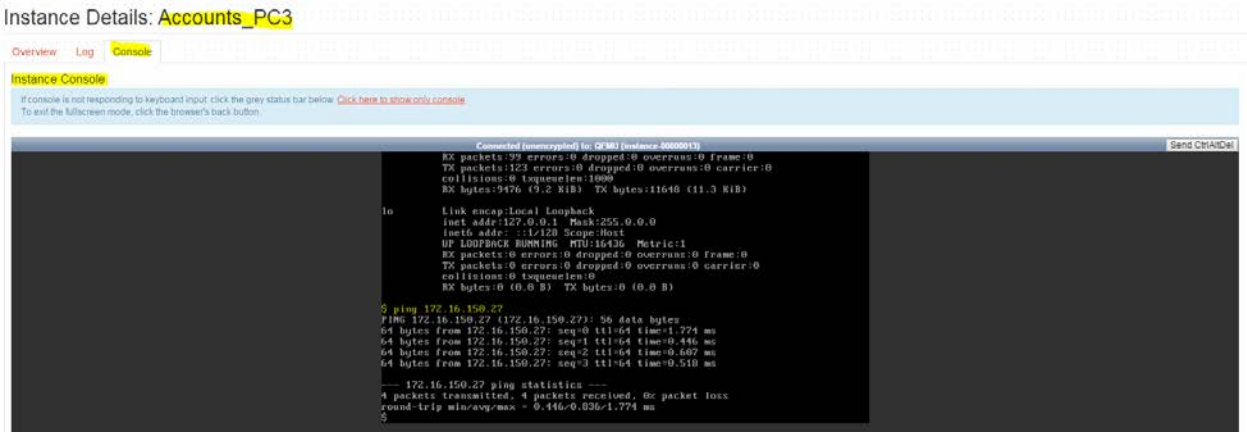


Network Topology should look like this:

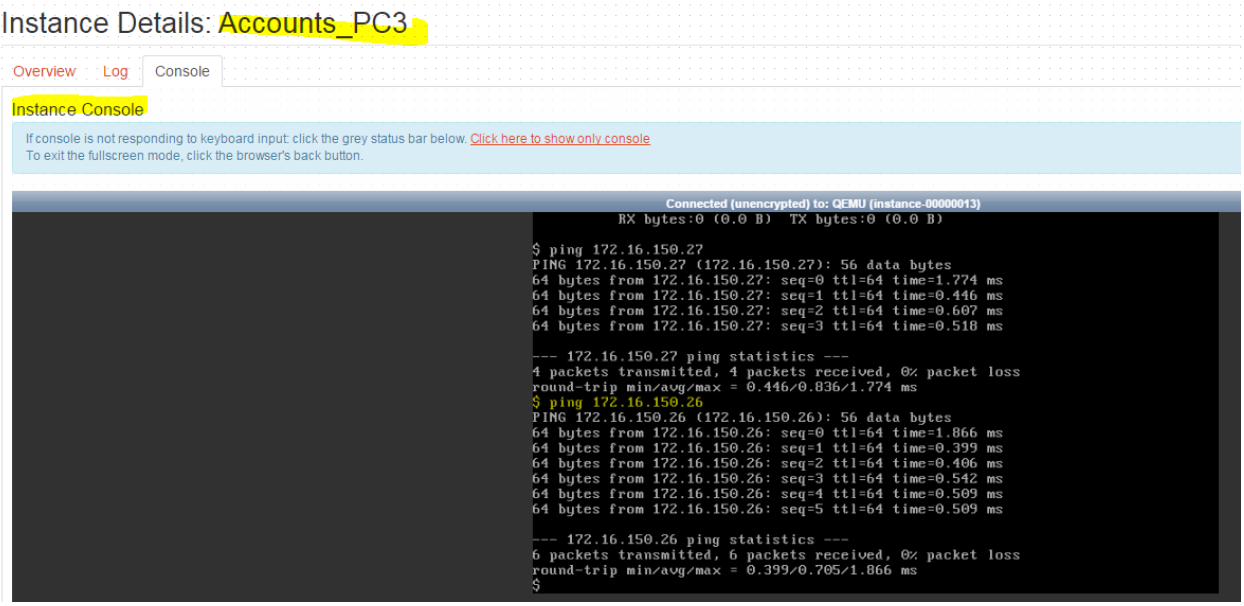


Now we will ping Accounts VMs to each other, and the VMs in Marketing and Sales Virtual Networks.

Ping from Accounts_PC3 to Accounts_PC2

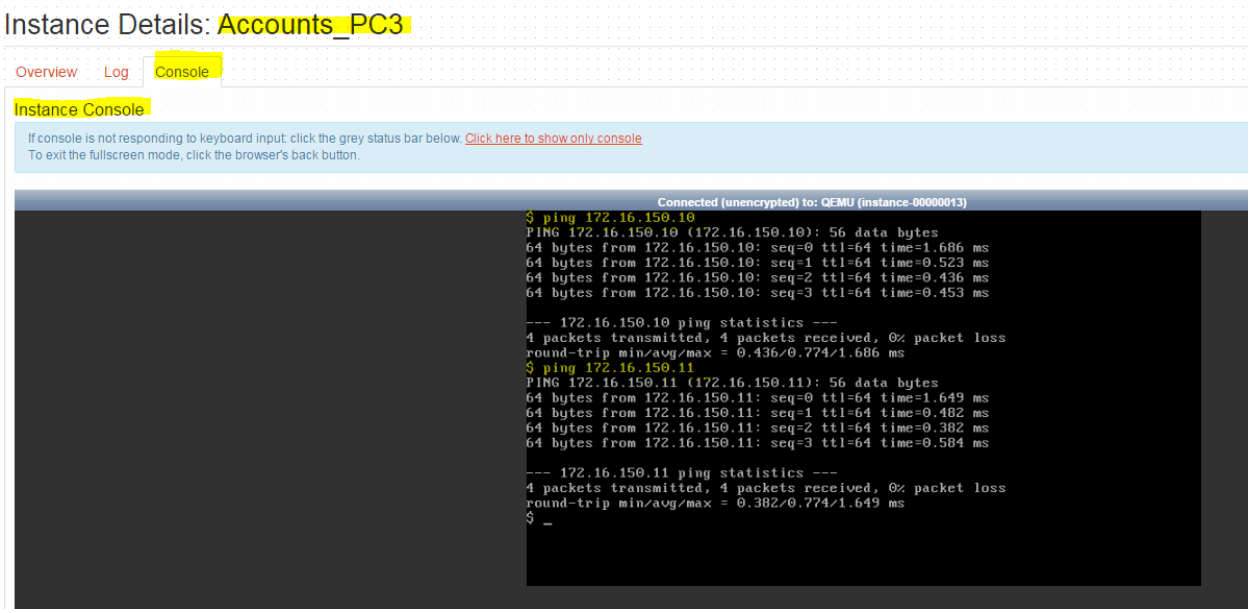


Ping from Accounts_PC3 to Accounts_PC1

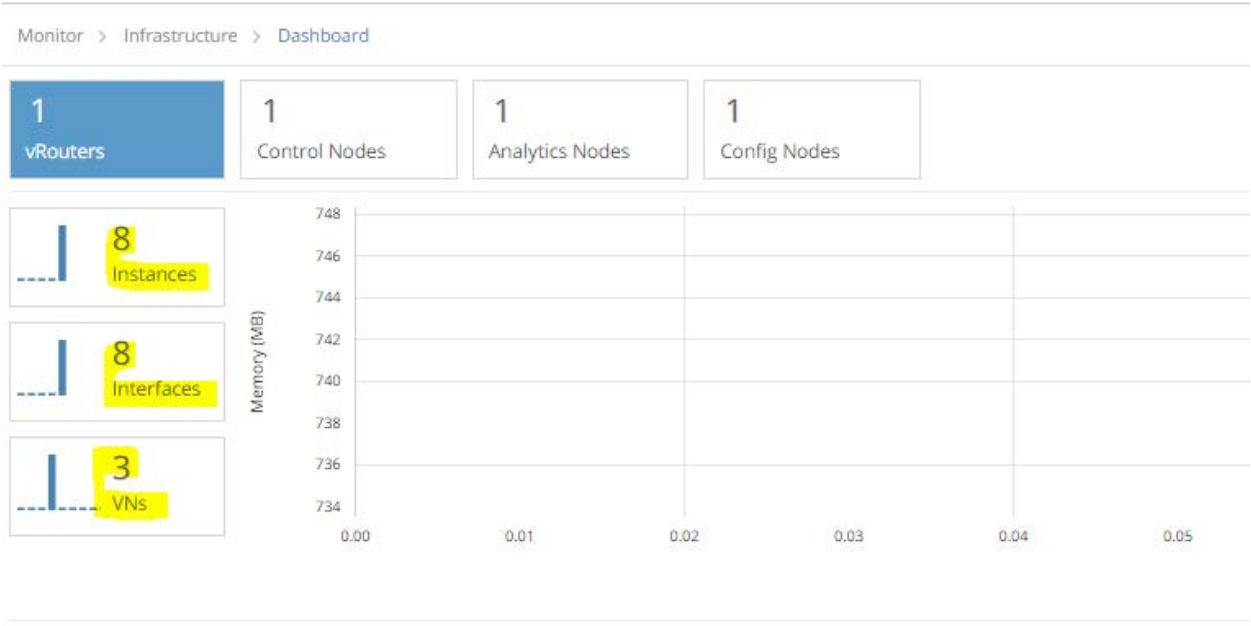


Ping from Accounts_PC3 to Sales_PC1, Sales_PC2, and Sales_PC3

Ping from Accounts_PC3 to Marketing_PC1, and Marketing_PC2.



Now we will take a look at Juniper Contrail Network Summary.



I have done this with a very limited amount of hardware, one can create a with OpenContrail setup. There is so much more that OpenContrail has to offer than what is covered here but this should lays the foundation for building simple virtual networks and to interconnect each of them.

Conclusion

As you see in the implementation section that ***Network Virtualization*** creates an artificial view of the network that hides the physical network (also known as Underlay Network) from clients and servers. It provides network routing between different virtual networks, different network services (i.e., NAT, IDS/IPS, Load-balancing, Firewall, etc.), and network isolation (where clients and servers only allowed to communicate with specific systems).

A question arises now that why people care about all of this and why this challenges the status quo on how networking is done today?

When business requirements mandate that network traffic is separated on two or more different networks, the traditional solution is to build two or more separate physical networks. However, this solution often leads to not optimal use of resources. And having separate physical networks can also lead to the difficulty of managing separate networks. Network virtualization allows you to combine the different networks onto a shared physical infrastructure while still keeping the logical separation to satisfy the original business requirements. This kind of technology allows for higher resource efficiency and cost while also reducing the time to perform network changes.

In traditional networking, we use STP to avoid loops in Ethernet networks, therefore, resulting in network resources that cannot be used and a fairly challenging implementation one has to plan. Whereas, network virtualization transforms the physical connection into simpler logical entities improving resource utilization and reducing design complexities.

Network virtualization technology truly enhances the:

- Reliability
    - It makes it possible for network communications to fail over from one network to the other in the case of a failure.
- Security
    - Only specific network traffic may pass through from one zone to another or from the internal network to the external network.
        - This reduces the possibility of clients or servers can be affected by viruses.
    - It is also possible to allow clients to only see servers they are allowed to access, even though other network resources are available.
- Diversity
    - It is very unlikely that single network architecture fits all uses. The diversity of network use is so large that there are too many conflicting demands that cannot be brought into agreement in a single network.
        - For example, on-demand video conferencing or voice over IP has a requirement of explicit QoS functionality – another example would be

secure online banking, where application may require heavy security protocols to provide authentication of end-system identities, Privacy of communication, and defenses against DoS or man-in-the-middle attacks.

- In network virtualization technology, we can slices two different virtual networks in order to achieve different protocol stacks on a single network to accommodate above mentioned example.

Juniper OpenContrail is grounded on the Border Gateway Protocol (BGP). The controller also employs XMPP (Extensible Messaging and Presence Protocol), a protocol for transmitting message-oriented middleware messages, to control the virtual switches inside hypervisors. Juniper uses Multiprotocol Label Switching (MPLS), which encapsulates packets on a network and controls their forwarding through those labels; it truly makes easier for enterprises to be work ion existing technologies instead investing on the learning curve creates by inventing new technologies for forwarding L3 packets.

One of the biggest differences between an OpenFlow-based controller and Juniper's controller is that the Contrail retains the original copy of the forwarding tables on the controller and duplicates them to the switches rather than holding the original copies on the switches and combining them on the controller after they have been altered.

Installation of Juniper Contrail's packages are bit difficult (not straight forward as compare to VMware NSX) but once the servers are up and running with all the roles have provisioned, it is very user friendly (as shown in Implementation section), manage, and monitor the whole cluster.

The advantage of using Juniper Contrail over Cisco ACI is that Juniper's solution is implemented with an open technologies like OpenStack for orchestration and OpenContrail for network virtualization to implement cloud networking and uses physical network (underlay network) so very cost effective to both enterprise and SPs. While Cisco ACI solution is solely deployed physically through Cisco Nexus 9000 switches and no cost benefits would avail by SPs or enterprises. OpenFlow is not supported by Juniper whereas Cisco ACI supports it.

Bibliography

1. Apache Cassandra website. [Online] http://cassandra.apache.org/.

2. **Sabharwal, Navin and Shankar, Ravi.** *Apache CloudStack Cloud Computing.*

3. **Juniper** OpenContrail Architecture. [Online] http://www.opencontrail.org/opencontrail-architecture-documentation/.

4. **Tulloch, Mitch.** *Introducing Windows Server 2012 R2.* s.l. : Microsoft Press, 2013.

5. *Figure 1.* [Online] http://www.enterprisenetworkingplanet.com/datacenter/network-virtualization-shooting-star-or-brave-new-world.html.

6. *if-map.org website.* [Online] http://www.if-map.org/.

7. *IETF XMPP working group.* [Online] http://datatracker.ietf.org/wg/xmpp/.

8. *Sandesh.* [Online] http://juniper.github.io/contrail-vnc/sandesh.html.

9. *Netconf RFC-6241.* [Online] https://tools.ietf.org/html/rfc6241.

10. *OpenContrail bring up and provisioning.* [Online] https://github.com/Juniper/contrail-controller/wiki/OpenContrail-bring-up-and-provisioning.

11. *Configuring Contrail for OpenStack.* [Online] https://github.com/Juniper/contrail-controller/wiki/Configuring-Contrail-for-OpenStack.

12. *Contrail Project Overview.* [Online] https://github.com/Juniper/contrail-controller/wiki/Contrail:-Project-Overview.

13. **Juniper.** OpenContrail – Quick Start Guide. [Online] http://www.opencontrail.org/opencontrail-quick-start-guide/.

- Figure 1: http://www.enterprisenetworkingplanet.com/datacenter/network-virtualization-shooting-star-or-brave-new-world.html
- http://www.opencontrail.org/architecture-document/
- http://xml.coverpages.org/news2003Q4.html
- https://github.com/Juniper/contrail-controller/wiki/Metadata-service
- http://techwiki.juniper.net/Documentation/Contrail/Contrail_Common_Support_Answers/000_In_Context%3A_Common_Support_Answers
- http://www.tuicool.com/articles/F3Evqu
- http://www.opencontrail.org/opencontrail-architecture-documentation/

- http://techwiki.juniper.net/Documentation/Contrail/Contrail_Common_Support_Answers/000_In_Context%3A_Common_Support_Answers
- http://www.juniper.net/us/en/local/pdf/whitepapers/2000535-en.pdf