University of Alberta

MINT 709 Capstone Project Report March 2016

OPENSTACK ORCHESTRATE PUBLIC AND PRIVATE CLOUD USING OPENSTACK/VCENTER INTEGRATION AND PROVIDING TENANT SEPARATION

BIVEK RAJ SINGH

Masters of science in Internetworking

Supervisor: Muhammad Durrani Brocade Communications Systems

Abstract

Rapid development and advancement in virtualization technology has made it possible to effectively distribute physical resources of a machine across multiple operating systems running simultaneously. With the virtualization technology getting more and more popular the need was felt to centrally control and manage these virtual infrastructures which led to the evolution of cloud computing.

With the concept of cloud computing gaining momentum many solutions emerged which promise to provide complete virtualization and cloud computing package, but most of the solution being proprietary solution followed different standard and architecture making them incompatible with one another forcing a corporation to stick to a single solution. This led to a need of a solution which could support multiple proprietary solution. The solution came in the form of a community supported open source software package "OpenStack".

This project presents a proof of concept implementation of integrating OpenStack with vCenter. It demonstrates the step by step deployment of Mirantis OpenStack using Fuel (Intuitive GUI based tool) and integrate it with the vCenter server to utilize the vSphere ESXi infrastructure as the compute hypervisor providing central control and management using OpenStack dashboard. This project also puts light into OpenStack Heat orchestration to create instances and finally demonstrates the use of Vyatta vRouter as the network management tool for OpenStack cloud.

With OpenStack having the potential to be the future of cloud operation and management system, the knowledge of OpenStack architecture and the ability to implement and administer cloud infrastructure using OpenStack as shown in the project would be a great asset to have for an internetworking graduate who wish to pursuit their career in the field of datacenter or cloud design and administration.

Acknowledgement

I would like to thank Dr. Mike MacGregor (*MINT Director*) and Mr. Shanawaz Mir (*MINT Coordinator*) for providing me the opportunity to work on this project. I would also like to express my gratitude to Mr. Muhammad Durrani for supervising and providing valuable suggestions and guidance during the implementation of this project. Finally, I would like to thank my family and all my friends who have directly or indirectly helped me for the successful completion of this project.

TABLE OF CONTENTS

1	Intro	oduction	1
2	Terr	minology and Concept	2
	2.1	Virtualization	2
	2.2	Hypervisor	2
	2.3	CLOUD COMPUTING	3
	2.4	Restful API	3
3	Con	nponents Of Project	4
	3.1	OpenStack	4
	3.2	VMWare vSphere	6
	3.3	OpenStack vCenter Integration	7
	3.4	Brocade Vyatta vRouter	8
4	Net	work Topology Design Considerations	9
	4.1	Block Diagrammatic View of the Project Infrastructure	9
	4.2	Physical and Logical Network Connection of Mirantis OpenStack with vCenter 1	0
	4.3	Physical and Logical Network connection view of Project Infrastructure 1	1
5	Imp	blementing Virtual Infrastructure 1	12
	5.1	VMware Workstation Infrastructure Setup 1	12
	5.1.	1 Vyatta 1	13
	5.1.	2 ESXi 1 1	15
	5.1.	3 ESXi 2 1	17
	5.1.	4 vCenter Server 1	8
	5.2	vCenter Infrastructure Setup	20
	5.3	Deploying Mirantis OpenStack 5.0.1 VApp	21
	5.3.	1 Setting up network	21
	5.3.	2 Importing Mirantis OpenStack 2	22
	5.3.	3 Installing Fuel	24
	5.3.	.4 Installing OpenStack	26
	5.4	OpenStack Dashboard	35
6	Lab	Experiment Demo With Result	36
	6.1	Creating Instances using Heat Orchestration	36
	6.1.	1 Creating Image	36
	6.1.	2 Flavors	37
	6.1.	3 Orchestration	37
	6.2	Tenant Separation Using Vyatta Firewall 4	12
	6.2.	1 Vyatta vRouter Network Configuring 4	12
	6.2.	2 Vyatta vRouter Firewall configuration using Rest API 4	16
7	Sun	nmary and Conclusion4	19

TABLE OF FIGURES

Figure 2.1 type 1 hypervisor	2
Figure 2.2 type 1 hypervisor	······ 2
Figure 2.1 OpenStack components overview	2
Figure 3.1 OpenStack components overview	
Figure 5.2 Viviware vspilere overview	0
Figure 5.5 v M ware driver architecture	/
Figure 4.1 Block diagram representation of OpenStack/vCenter integration	9
Figure 4.2 Mirantis OpenStack physical and logical connection with vCenter	10
Figure 4.3 Physical and logical network connection diagram	11
Figure 5.1 View of VMWare Workstation listing all the Virtual Machines	12
Figure 5.2 View of VMWare Workstation Virtual Networks	12
Figure 5.3 Vyatta VM settings on VMware Workstation	13
Figure 5.4 View of Vyatta VM deployed on VMware Workstation	14
Figure 5.5 View of Vyatta vRouter VM after installing image on local drive	14
Figure 5.6 ESXI 1 VM setting on VMWare Workstation	15
Figure 5.7 Set static IP to ESXi 1	16
Figure 5.8 View of ESXI 1 on VMWare workstation	16
Figure 5.9 ESXI 2 VM setting on VMWare Workstation	17
Figure 5.10 View of ESXI 2 on VMWare workstation	17
Figure 5.11 VCenter Server VM setting on VMWare Workstation	18
Figure 5.12View of VCenter Server on VMWare workstation	18
Figure 5.13 vCenter Server Setup login screen	19
Figure 5.14 step by step vCenter Server Appliance Setup	19
Figure 5.15 VMware vSphere client login	20
Figure 5.16 Creating clusters OpenStack and NovaCompute	20
Figure 5.17 view of vCenter inventory after adding hosts to clusters	21
Figure 5.18 Virtual Machine Port Group created to deploy Mirantis OpenStack	21
Figure 5.19 VM switch ports security setting	22
Figure 5.20 Deploy Mirantis OpenStack VApp	22
Figure 5.21 vCenter host inventory view after Deployment of Mirantis OpenStack VApp	23
Figure 5.22 ESXi 1 network view after deploying Mirantis OpenStack VApp	
Figure 5.23 View of Fuel Master VM console	24
Figure 5.24 view of Fuel Master VM properties	24
Figure 5.25 view of Fuel Master VM console immediately after booting up with M	irantis
OpenStack iso	25
Figure 5.26 view of Fuel Master configuration setting	25
Figure 5.27 view of mos-child-1VM properties and console	25 76
Figure 5.28 view Fuel Master console	20 76
Figure 5.20 view of Evel Deshboard	20
Figure 3.29 view of Fuel Dashooard	21

Figure 5.30 Adding and assigning role to nodes	31
Figure 5.31 Configure interfaces on 3 nodes	32
Figure 5.32 OpenStack Network Setting	33
Figure 5.33 OpenStack Setting	34
Figure 5.34 view of OpenStack Horizon Dashboard login screen	35
Figure 5.35 OpenStack Horizon Dashboard Overview	35
Figure 6.1 Create An Image dialog box	36
Figure 6.2 View of images tab after creating multiple images	37
Figure 6.3 view of default OpenStack Flavors	37
Figure 6.4 Select Template dialogue box	38
Figure 6.5 Launch Stack dialogue box	39
Figure 6.6 view of Stacks	40
Figure 6.7 view of WindowsXP instance Overview and console	40
Figure 6.8 view of Ubuntu instance Overview and console	41
Figure 6.9 view of Fedora instance Overview and console	41
Figure 6.10 view of vCenter NovaCompute cluster after OpenStack instant deplyoment	41
Figure 6.11 view of vCenter Networking	42
Figure 6.12 view of ip configuration of Windows, Ubuntu and Fedora instances	43
Figure 6.13 Enabling HTTPS on Vyatta system	46
Figure 6.14 start configuration session and create a unique session ID using curl command	46
Figure 6.15 List active configuration mode sessions	46
Figure 6.16 view of Vyatta router firewall configuration	47
Figure 6.17 ping response before configuring firewall	48
Figure 6.18 ping response after configuring firewall	48

1 INTRODUCTION

The advancement in compute, storage and networking capacity and an increase in the adaption of virtualization technology has changed the architecture of modern datacenter into a service oriented architecture which utilizes distributed computing resources across the datacenter to provide utility computing. This shift from on demand computing to a service based distributed computing led to the evolution of cloud computing. Offering features such as high performance, scalability and economic service the demand of cloud computing is increasing day by day. This increase in demand created a need for software that provides a complete cloud computing solution package.

There are many solutions developed that provide complete cloud computing solution. But the problem with most of the solutions is they are proprietary solution and have their own hardware, software and virtualization standards. This makes one solution incompatible with other which makes it very difficult to implement these solutions in large environments with multiple underline physical hardware and virtualization technology used. This initiated an open source community effort to develop a ubiquitous cloud computing solution that supports multiple hardware and hypervisor platform called "OpenStack". Initially developed by NASA and Rackspace, OpenStack is now managed by OpenStack Foundation. Within 5 short years of its development OpenStack has gained a huge popularity in the world of cloud computing with contributions from major player like Cisco, HP, RedHat, IBM etc.

In this project we demonstrate management of cloud on VMware vSphere infrastructure using OpenStack vCenter integration. VMware vSphere is one of the most popular virtualization solution used across enterprise datacenters. This integration of OpenStack with vCenter brings all the benefits of vSphere to OpenStack environment which includes the advance feature like vMotion, High Availability(HA), Fault Tolerance (FT) and Dynamic Resource Scheduling (DRS) making it more scalable, flexible, robust and enterprise friendly. Also, it opens the possibility to integrate other hypervisors (KVM, XEN etc.) to the cloud under the management of OpenStack.

Scope

The scope of this project is to presents the proof of concept demonstration to orchestrate a private and/or public cloud using OpenStack on vSphere infrastructure (OpenStack vCenter integration) and manage the network resources of the cloud using Vyatta vRouter providing tenant separation.

2 TERMINOLOGY AND CONCEPT

2.1 VIRTUALIZATION

Virtualization means creating a virtual version of an actual implementation or process. It is an abstraction layer separating hardware from the software using a software that emulates the underline hardware. Today with the advancement in virtualization technology almost all the physical attribute of a computer can be virtualized which include memory virtualization, storage virtualization, network virtualization etc.

By using virtualization technology, it is possible to capture the state of all the attributes at a point of time (take snapshot) and save it so that user can roll back to that instance at any time. That state can also be copied from one physical machine to another eliminating complete dependency on one physical machine.

2.2 HYPERVISOR

Hypervisor is the software program that makes virtualization possible. It acts as a bridge between the physical hardware and the operating systems running on that hardware. It allows multiple operating systems to run simultaneously on top of a physical hardware by managing and distributing the available physical resources to each operation system. Hypervisors can be classified into two types:

• Type 1 hypervisors:

Type 1 hypervisors are also called bare metal or native hypervisors. They are installed directly on physical hardware and are more efficient. They are used in production grade datacenters and servers across enterprise environment e.g. Cirtix Xen server, VMWare ESX/ESXi etc.



Figure 2.1 type 1 hypervisor

• Type 2 hypervisors:

Type 2 hypervisors are the hypervisors that are installed on an operating system. They run guest operating system inside host operating system. e.g. VMWare Workstation, VMWare player, virtual box etc.



Figure 2.2 type 2 hypervisor

2.3 CLOUD COMPUTING

Cloud computing is the process of utilizing the distributed hardware resources across the network to perform computing task. The unused processing cycle of multiple processors connected across the network are utilized to perform a computing operating which makes cloud computing more efficient in terms of resource utilization. It allows users to share high power computing resources securely and effectively. [14] Cloud computing has hugely benefited small and medium enterprise as it has allowed them to use computing power as per requirement as a service eliminating the upfront cost of implementing whole infrastructure and maintain them.

Deployment model of cloud

• Private Cloud:

Private cloud is the cloud that is entirely owned and managed by a single organization and is tailored as per the organization's requirements. All the infrastructure and services of a private cloud are connected to organization's private network. Creating a private cloud requires the organization to own significant physical resources and is expensive to build and manage. Private cloud is popular among the companies with very high data security and secrecy requirements such as financial institutions and intelligence services.

• Public Cloud:

Public cloud is the cloud which are built and managed by cloud service providers and are connected to public network which can be accessed through internet. The user of the public cloud has no control over the infrastructure and architecture of the cloud and use the cloud as a service. Public cloud is much more efficient and economical from end user prospective. Google, Microsoft, Amazon are some of the large public cloud service providers.

• Hybrid Cloud:

Hybrid cloud is the combination of public and private cloud. In hybrid cloud multiple clouds are integrated as per the requirement of the organization to form a new cloud thus its implementation may be different for different organizations.

2.4 RESTFUL API

RESTful (Representational state transfer) API is the application programming interface following REST software architectural style which communicates over HTTP (Hypertext Transfer Protocol). It is the API supported by most of the cloud based web services including Google, Amazon, Facebook, Twitter etc. The communication is done using the following four methods as defined in HTTP RFC 2616:

• PUT

Put is a idempotent method it is used to change the state or update data

- **GET** Get is a nullipotent method used to retrieve data but does not change any data
- **POST** Post method is used to create a data
- **DELETE** Delete is also an idempotent method used to remove data

In our project we use Restful API to configure firewall on Vyatta vRouter using **cURL** command line.

3 COMPONENTS OF PROJECT

3.1 OPENSTACK

OpenStack is an open source software that provides cloud computing services. It consists of several interrelated components which are responsible for the control of hardware, storage and networking resources packaged together to form a complete cloud operating system. There is an intuitive GUI based web dashboard for management. Users can also use Restful API or command line tools for management. OpenStack supports all the popular open source and enterprise solutions making it a highly ubiquitous cloud computing platform ideal for infrastructure consisting of solutions from multiple vendors.

Components of OpenStack

OpenStack consist of several different projects combined together to form a superior cloud computing platform. Listed below are the core components of OpenStack.

Compute (Nova)

Nova is the major component of OpenStack which manages the compute instances. It is responsible for spawning, scheduling and decommissioning of machines on demand. It manages and automates compute resources pool and supports all widely available virtualization technologies and bare metal high performance configurations. [1] [11]

Object Storage(Swift)

Swift is the object storage system in OpenStack. It is a highly scalable and redundant system. It uses RESTful API based on HTTP protocol to store and retrieve arbitrary unstructured data objects. It has a scale out architecture and provides data replication making it highly fault tolerant. [1] [11]

Block Storage(Cinder)

Cinder provides persistent block storage to running instances. It has a pluggable driver architecture. Creation, attachment and detachment of the block devices to servers is managed by cinder. [1] [11]

> Networking(Neutron)

Neutron is a system for managing network and IP address. It enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. It ensures the prevention of network bottleneck during cloud deployments. Advance routing services from vendors is support by neutron because of its pluggable backend architecture. [1] [11]

Dashboard(Horizon)

OpenStack Dashboard (Horizon) is a web based user interface. Administrators and users use this interface to provision and automate cloud-based resources. The design allows for third party products and services, such as billing, monitoring and additional management tools. It can be customized by vendors and services providers with their own brand. [1] [11]

> Identity Service(Keystone)

Keystone provides authentication and authorization service for other OpenStack services. It can be integrated to backend directory services like LDAP and supports multiple forms of authentication. [1] [11]

Image Service(Glance)

Glance provides discovery, registration and delivery for disk and server images. It stores and retrieves virtual machine disk images. It uses REST API to query information about disk image and allows clients to stream the image to new servers. [1] [11]



-Source: OpenStack Training Guides (May 10, 2015)

Figure 3.1 OpenStack components overview

Source: http://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html (13/3/2016)

Beside the above listed core components some of the other components of OpenStack are:

> Orchestration (Heat)

Heat orchestration is used to launch multiple cloud application based on templates. It automatizes the scaling and addition of compute, storage and network resources across the cloud platform through an OpenStack-native REST API by executing HOT (Heat Orchestration Template) templates written in YAML. [9]

> Telemetry (Ceilometer)

Ceilometer is to metering tool. It collects and stores data used for automated actions or billing / chargeback purposes in form of samples. [9]

-Source: https://www.OpenStack.org/software/icehouse/

3.2 VMWARE vSPHERE

VMWare vSphere is a cloud computing and virtualizing software package developed by VMware. It is the most popular virtualization and cloud management tool and widely used across enterprise datacenters. It manages large volume of computing, storage and networking resources across the datacenter seamlessly. It offers distributive services such as vMotion, storage vMotion, Distributed Resource Scheduler (DRS), High Availability HA, and fault tolerance which enable very efficient and automated management of resources providing a highly available and reliable virtual infrastructure. [3]



Components of VMware vSphere

VMware vSphere includes the following components:

> VMware ESXi

VMware ESXi a hypervisor layer that is installed on a bare metal (physical) servers. It provides effective distribution of processor, memory and storage resources across multiple virtual machines. [3]

> VMware vCenter Server

VMware vCenter Server is the management tool to manage ESXi hypervisors across the datacenters. It provides central management of compute, storage and network resources of all the ESX/ESXi across the datacenter by aggregating them into a cluster to create a highly reliable, fault tolerant and efficient virtual infrastructure. [3]

> VMware vSphere Client

VMware vSphere Client is a software that can be installed on any Windows PC that provides an interface to remotely connect and manage ESXi and vCenter server. [3]

> VMware vSphere Web Access

VMware vSphere Web Access is web based GUI interface which provides an intuitive way to manage and administer vCenter server. [3]

-Source: Introduction to VMware vSphere (EN-000102-00)

3.3 OPENSTACK VCENTER INTEGRATION



Figure 3.3VMWare driver architecture -source: OpenStack Configuration Reference icehouse (June 1, 2015)

One of the most important feature of OpenStack is its ability to integrate with vCenter to control and manage the vSphere infrastructure. VMware VSphere is the most popular virtualization and cloud management software used across the enterprise datacenters. By integrating OpenStack with vCenter the advance features offered by vCenter such as vMotion, High Availability (HA), Fault tolerance (FT) and Dynamic Resource Scheduler (DRS) can be leveraged on OpenStack environment.

OpenStack nova-compute communicates with vCenter using VMware vCenter driver. The nova compute service acts as a proxy to translate Nova API calls to vCenter API calls using vCenter driver. Nova scheduler chooses a cluster on vCenter to launch an instance. Nova-compute then makes an API call and hands over the request to vCenter to launch a VM on the cluster chosen by nova scheduler. The actual ESXi host is then chosen by vCenter using Dynamic Resource Scheduler (DRS). [17]

OpenStack Networking on vSphere is done using nova-network service. It can be configured in following ways:

- Nova-network service with the FlatManager or FlatDHCPManager In this configuration all VM NICs are attached to the same port group. The name of the port group should be same as the flat_network_bridge value defined in nova.conf. It is br100 by default. [2]
- Nova-network service with the with VlanManager In this configuration VM NICs are attached to the port groups created automatically by OpenStack compute to handle VLAN-tagged VM traffic. [2]

3.4 BROCADE VYATTA VROUTER

The Brocade Vyatta vRouter is a Debian-Linux distribution based software that provides virtual routing, firewall and VPN services for cloud computing environments. Listed below are some of the features of Vyatta vRouter:

Network Connectivity

- > IPv4 and IPv6 dynamic routing protocols support
 - o BGP
 - o OSPF
 - o RIP
 - o Multicast
- Policy Based routing (PBR)
- ➢ 802.11 wireless
- Serial WAN interfaces
- ➢ Ethernet interfaces up to 10Gbps

Firewall Protection

- ➢ IPv4/IPv6 stateful packet inspection
- > zone- and time-based firewalling
- > P2P filtering.

Secure Connectivity

- secure site-to-site VPN tunnels with standards-based IPsec VPN
- secure network access to remote users via Brocade SSL-based OpenVPN functionality
- Dynamic Multipoint VPN (DMVPN)

Administration and Authentication

- network-centric CLI,
- ➢ Web-based GUI,
- > external management systems using the Brocade Remote Access API.
- ▶ securely managed network management sessions using SSHv2, RADIUS, or TACACS+.

4 NETWORK TOPOLOGY DESIGN CONSIDERATIONS

4.1 BLOCK DIAGRAMMATIC VIEW OF THE PROJECT INFRASTRUCTURE



Figure 4.1 Block diagram representation of OpenStack/vCenter integration

4.2 PHYSICAL AND LOGICAL NETWORK CONNECTION OF MIRANTIS OPENSTACK WITH VCENTER



Legend

Physical network connection
 Logical network connections
 (VLAN, tunnels, etc.):
 Fuel Admin (PXE) network
 OpenStack Fixed (Private) network
 OpenStack Storage network
 OpenStack Public network
 OpenStack Public network
 }
}

Fuel Master and OpenStack Controllers nodes' eth0 NICs contain Fuel Admin (PXE) traffic as untagged. OpenStack Controllers nodes' eth0 NICs also contain Storage traffic tagged as VLAN 102. OpenStack nodes' eth1 NICs contain Public traffic as untagged. OpenStack nodes' eth2 NICs contain Management traffic as untagged.

Figure 4.2Mirantis OpenStack physical and logical connection with vCenter

Source: Mirantis Reference Architecture (17 December 2014), page 9

(NSX portion is removed from the diagram as it is not implemented in this project)



4.3 PHYSICAL AND LOGICAL NETWORK CONNECTION VIEW OF PROJECT INFRASTRUCTURE

Figure 4.3 Physical and logical network connection diagram

5 IMPLEMENTING VIRTUAL INFRASTRUCTURE

5.1 VMware Workstation Infrastructure Setup



Figure 5.1 View of VMWare Workstation listing all the Virtual Machines

The infrastructure for this project is setup in a VMware Workstation 12 Pro installed on Windows Server 2012 system with 64 GB Ram, 1TB Hard disk drive and Intel Xeon 2.4GHz processor. Following four Virtual Machines are created for the project:

- Vyatta
- ESXi 1
- ESXi 2
- VMware-vCenter-Server

Also a new host only virtual network "vMnet3" is created on VMware Workstation.

Virtual Network Editor						
Name	Туре	External Connection	Host Connection	DHCP	Subnet Address	
VMnet0	Bridged	Broadcom NetXtreme Gigabi	-		-	
VMnet3	Host-only	-	Connected	-	192, 168, 3,0	
VMnet8	NAT	NAT	Connected	Enabled	192.168.169.0	
Add Network Remove Network Add Network						
Bridge	ed to: Broad	com NetXtreme Gigabit Ethernet	#2	~	Automatic Settings	
O NAT (s	shared host's	IP address with VMs)			NAT Settings	
• Host-o	only (connect	VMs internally in a private netwo	ork)			
Connect a host virtual adapter to this network Host virtual adapter name: VMware Network Adapter VMnet3 Use local DHCP service to distribute IP address to VMs DHCP Settings						
Subnet IP: 192 . 168 . 3 . 0 Subnet mask: 255 . 255 . 255 . 0						
Restore Defaults OK Cancel Apply Help						

Figure 5.2 View of VMWare Workstation Virtual Networks

5.1.1 Vyatta

Vyatta VM boots using Brocade Vyatta 5410 vRouter ISO. The Vyatta vRouter performs following functions in this project:

- DHCP server (with Static MAC-IP Mapping)
- Network Address Translation (NAT)
- Firewall

Creation and Deployment of Vyatta VRouter

Step 1: Create a new Virtual Machine in VMware Workstation to boot from **Vyatta vRouter 5400 6.7 64-bit** ISO image, the 60-day trial version of which can be downloaded from the following link: <u>http://www1.brocade.com/forms/jsp/Vyatta-download/index.jsp</u>

Step 2: Go to Virtual Machine setting and set the memory for this virtual machine to be 512MB with one processor and 20GB hard disk space.

Step 3: Set Network Adapter to Bridged (Automatic) also add a second Network Adapter "Network Adapter 2" and set the network connection to VMnet3

	Virtual Mac	chine Settings	x
Hardware Options	Summary 512 MB 1 20 GB Using file C:\software\vyatta\vyatta Bridged (Automatic) Custom (VMnet3) Auto detect	Memory Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB. Memory for this virtual machine: 512 MB 64 GB - 32 GB - 16 GB - 8 GB - 4 GB - 2 GB - 1 GB - 57304 MB 512 MB - 128 MB - 64 MB - 32 MB - 64 MB - 16 Guest OS recommended minimu 64 MB - 8 MB - 4 MB - 16 MB - 18 MB - 19 MB - 10 M	y
	🕞 Add Remove		
		OK Cancel He	lp

Figure 5.3 Vyatta VM settings on VMware Workstation

Step 4: Power up the virtual machine and after successful boot, install image on local hard drive using **install image** command.



Figure 5.4 View of Vyatta VM deployed on VMware Workstation

drive VM. Step 5: After successfully installing image local reboot the on Loading open-vm-tools modules: vmhgfs vmmemctl. Starting open-vm daemon: vmtoolsd. Starting ACPI services.... Starting deferred execution scheduler: atd. Starting periodic command scheduler: cron. Starting network plug daemon: netplugd. Loading cpufreq kernel modules...done (none). Starting routing daemons: imi nsm mribd pimd ripd ripngd ospfd ospf6d bgpd. Starting Vyatta router: migrate rl-system firewall configure. Welcome to Vyatta – vyatta tty1 vyatta login: vyatta Welcome to Vyatta Version: Description: VSE6.7R9T60 Brocade Vyatta 5410 vRouter 6.7 R9T60 Copyright: 2006–2015 Vyatta, Inc. vyatta@vyatta:~\$ vyatta@vyatta:~\$ _

Figure 5.5 View of Vyatta vRouter VM after installing image on local drive

Step 6: Finally configure the interface connected to bridged interface to get IP from DHCP. For the other interface connected to VMnet 3 assign a static IP of 192.168.3.254 and enable DHCP server with default gateway being 192.168.3.254. Also enable NATing on the router setting the bridged interface as outside interface.

5.1.2 ESXi 1

ESXi 1 VM boots using VMware ESXI 5.5 ISO image. This ESXi hypervisor is setup to host the Mirantis OpenStack Fuel master and three child nodes.

Creation and Deployment of ESXi 1

Step 1: Create a new Virtual Machine in VMware Workstation to boot from VMware VSphere 5.5 ISO which can be downloaded from the following link:

https://my.vmware.com/group/vmware/evalcenter?lp=default&p=free-esxi5

Step 2: Go to Virtual Machine Setting and assign 32GB memory to the VM along with 8 processor cores. Also, assign 650GB hard disk space and set Network Adaptor to VMnet3.

	Virtual Mac	hine Settings
Hardware Options Device Memory Processors Hard Disk (SCSI) CD/DVD (IDE) Network Adapter USB Controller Display	Virtual Mac	Mine Settings Memory Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB. Memory for this virtual machine: 32 GB 16 GB 8 GB 16 GB 2 GB 16 GB 2 GB 16 GB 2 GB 16 B 2 GB 16 B 512 MB 256 MB 128 MB 64 MB 32 MB 64 MB 32 MB 4096 MB 16 MB 8 MB 4 MB
	Remove	
		OK Cancel Help

Figure 5.6 ESXI 1 VM setting on VMWare Workstation

Step 3: Power on the Virtual Machine and install ESXi on the local drive.

Step 4: After successful installation assign static IP address of 192.168.3.11 to this ESXi.



Figure 5.7 Set static IP to ESXi 1

	VMware ESXi 5.5.0 (VMKernel Release Build 2068190)		
	VMware, Inc. VMware Virtual Platform		
	2 x Intel(R) Xeon(R) CPU E5-2407 v2 @ 2.40GHz 32 GiB Menoru		
	ac and nerror g		
	Download tools to wanage this host from: http://192.160.3.11/ (STATIC) http://fcR0:200:29FC:sec:fcFa1/ (STATIC)		
(F2)	Custonize System/View Logs	(F12) Shut Down/Restart	

Figure 5.8 View of ESXI 1 on VMWare workstation

5.1.3 ESXi 2

Similar to ESXI 1, ESXi 2 VM boots using VMware ESXI 5.5 ISO image. All the instances created using OpenStack are hosted in this ESXi 2 hypervisor.

Creation and Deployment of ESXi 2

All the steps in creation of ESXi 2 are similar to ESXi 1 except ESXi 2 is assigned with only 20GB memory and 200GB hard disk space.

Figure 5.9 ESXI 2 VM setting on VMWare Workstation

Also, ESXi 2 is assigned with static IP 192.168.3.12



Figure 5.10 View of ESXI 2 on VMWare workstation

5.1.4 vCenter Server

vCenter Server VM boots using VMware-vCenter-Server-Appliance-5.5 OVF file. Both hypervisor ESXi 1 and ESXi 2 are managed using vCenter Server.

Creation and Deployment of vCenter Server

Step 1: Create a new Virtual Machine in VMware Workstation to boot from VMware-vCenter-Server-Appliance-5.5 ovf file.

Step 2: Go to virtual machine setting and set Network Adapter to VMnet 3

	Virtual Mac	hine Settings				
Hardware Options Device Procesors Hard Disk (SCS1) Hard Disk (SCS1) (CC)/VO (IDE) CO/VO (IDE) Display	Virtual Mac	Memory Memory Specify the anount of memory allocated to this virtual medine. The memory size must be a multiple of 4 MB. Memory for this virtual machine: 32 G8 4 G8 16 G8 2 G8 16 G8 16 G8 16 G8 17 C8 18 G8 12 M8 12 M8 12 M8 12 M8 12 M8 12 M8				
Coloro (uc.)	Auto detect	32 GB - ◄ 16 GB - ■ 8 GB - ■ 4 GB - ■ 2 GB - 0 Carc Deyrond this size.) 1 GB - ■ 32 768 MB ■ 512 MB - ■ 256 MB - ■ 768 MB ■ 64 MB - ■ 16 MB - \$12 MB 16 MB - \$12 MB 8 MB - 4 MB -				
Add Remove						

Figure 5.11 VCenter Server VM setting on VMWare Workstation

Step 3: Power on the virtual machine and install the vCenter Server.



Figure 5.12View of VCenter Server on VMWare workstation

Step 4: After successful installation of vCenter Server on VMware workstation, open a web browser and navigate to vCenter Server Appliance webpage <u>https://192.168.3.10:5480</u> (as shown in vCenter server console) and login using default username **root** and password **vmware**.

VMware vCenter Server Applia × +							٥	x
A https://192.168.3.10:5480/#core.Login	C	Q. Search	☆ €	÷	⋒	Ø	0	≡
👺 Fuel Dashboard - MIN 💶 Login - Mirantis Open 🔞 How to Configure Vya 🗧	🕘 Firewall.book - Vyatta 🍈 Brocade 💶 #4 VyOS/Vyatta VL	ANs 🛞 Setting up a configura						
vog ^o VMware vCente	er Server Appliance							
Login								
	User name: root							
	Password:							
	Login							
vm ware [,]	Copyright © 1998-2013 VMware, Inc. All rights reserved.	Powered by VMware Stu	dio					

Figure 5.13 vCenter Server Setup login screen

Step 5: Start vCenter Server Setup. Select Set custom configuration as the configuration options, database type embedded, assign new password for user administrator@vSphere.local and set time synchronization as VMware Tools synchronization

vCenter Server Setup		vCenter Server Setup				vCenter Server Setup		
Contex Server Setup Accept FULA Configure Options Database settings SSO settings Active Directory settings Time synchronization Review configuration Configure	To contigure this which a postance with a static IP address, you must first contigure the hostwarm. To only the cancel the weak of only the reflection address settings and effert water. The hostwarm is calculated with the reflection address settings. The hostwarm is advected on the settings of the reflection address settings. Configure with reflecti settings: Upgrade transmission remains. Upgrade transmission Reflection. Host file setterd.	Accept EULA Configure Options Databases settings SSD settings Active Directory settings Trimes synchronization Review configuration Configure	Ostabase type: Server: Port: wistance name: Login: Password:	embedded		vCenter Server Serve Accept DUA Configure Options Database settings SSO Settings Active Directory settings Time synchronicultion Review configuration Configure	SBO deployment type: Embedded SBO requires choos the user administrator gassword. Account with right to register vCe Usemanne: Password. Account that will be assigned as Name. Lookup service location:	embedded v ng a password for local inter with the SOD server Center administratic
vCenter Server Setup Accept EULA Configure Options	Bet custom configuration Cancel + Prov Next > Adve Directory Enabled Donaix	vCenter Server Setup Accept EULA Configure Options	C Ne synchronization	Cancel	< Prev Next>	VCenter Server Setup Accept EULA Configure Options	Certificate status: vCenter Database: Type:	Cancel «Prev Need»
Latabase settings 550 settings Active Directory settings Time synchronization Review configuration Configure	Administrator past revol	Database settings SSD settings Active Directory settings Time synchronization Review configuration Configure	NTP synchronization: NTP servers: Options: The servers field accepts The servers field accepts	a comma separated list of sen PPD recordiced options to be us allon callon callon is actuated by joining a v thodis cannot be used at the sa	ers ed for each server. Indows domain. me time with it.	Database settings SSO settings Active Directory settings Time synchronization Review configuration Configure	Host Port Instance: Logn: DB Reset SSD: Deployment type: Administrator account Is a group: Lookup senice: Lookup senice:	RO a ambeddad
	Cancel + Prev Next +			Cancel	< Prev Next >		a so reasonat: Type Host Port Instance	Cancel < Prev Start

Figure 5.14 step by step vCenter Server Appliance Setup

5.2 VCENTER INFRASTRUCTURE SETUP

Step 1: Open VMware vSphere Client. Enter the vCenter IP address, username and password and click on login.

Ø	VMware vSphere Client
vmware VMware vSphere Client	
All vSphere feat available only th vSphere Client v feature set as v To directly manage a To manage multiple ho vCenter Server.	ures introduced in vSphere 5.5 and beyond are rough the vSphere Web Client. The traditional vill continue to operate, supporting the same Sphere 5.0. single host, enter the IP address or host name. osts, enter the IP address or name of a
IP address / Nam User name: Password:	e: 192.168.3.10 administrator@vsphere.local ******** Use Windows session credentials Login Close

Figure 5.15 VMware vSphere client login

Step 2: Create a new datacenter and name it Datacenter.

Step 3: Create two new clusters OpenStack and NovaCompute and turn on vSphere DRS feature on both the clusters.



Figure 5.16 Creating clusters OpenStack and NovaCompute

Step 4: Add host ESXi 1 i.e. 192.168.3.11 to OpenStack cluster and ESXi 2 i.e. 192.168.3.12 to NovaCompute Cluster.



Figure 5.17 view of vCenter inventory after adding hosts to clusters

5.3 DEPLOYING MIRANTIS OPENSTACK 5.0.1 VAPP

A VMware Virtual Appliance (VApp) can be download from Mirantis website and imported to vCenter. This VApp offers a pre-configured deployment of virtual machines for Fuel, and OpenStack. The steps taken to deploy this VApp is described below:

5.3.1 Setting up network

Step 1: Create Virtual Machine Port Groups for deploying Mirantis OpenStack VApp on ESXi 1 and assign them with VLAN IDs as listed below:

Network	VLAN ID
Admin PXE	100
Management	101
Storage	102
Private	103
Public	0 (None)
Default	0 (None)

Configuration	Summary
T vSwitch	120 Ports
👳 Public	Virtual Machine
🧙 Management	Virtual Machine
👳 Private	Virtual Machine
😒 Storage	Virtual Machine
Admin PXE	Virtual Machine
👳 Default	Virtual Machine
Management N	et vMotion and IP

Figure 5.18 Virtual Machine Port Group created to deploy Mirantis OpenStack

Step 2: Configure all the port groups to accept Promiscuous mode, MAC Address Changes and Forged Transmits.

Ø	Public Properties X					
General Security Traffic Shap	ng NIC Teaming					
Promiscuous Mode: MAC Address Changes: Forged Transmits:	Iv Accept ▼ Iv Accept ▼ Iv Accept ▼					

Figure 5.19 VM switch ports security setting

5.3.2 Importing Mirantis OpenStack

Step 1: Download VApp for Mirantis OpenStack from the following link: <u>https://content.mirantis.com/vapp-mirantis-OpenStack-landing-page.html</u>

Step 2: Select host 192.168.3.11 (ESXi 1) and from the file menu click on Deploy OVF template.

Step 3: Specify the location of Mirantis OpenStack VApp ova file and Map the network as shown in the snapshot below:

0	Deploy OVF Template - C	Ø		Deploy OVF T	emplate – 🗖	×
Source Select the source location.		Ne	twork Happing What networks should t	he deployed template use?		
Source OVF Template Details		Sec. DV National Sec.	rce F Template Details me and Location	Map the networks used in this OVF ten	plate to networks in your inventory	
Storage		Sta	/ ace	Source Networks	Destination Networks	
Disk Format	And Annual and	Qs	k Format	Admin PXE	Admin PXE	
Ready to Complete	Deploy from a file of DRL	Re	adv to Complete	Default	Default	
	100 Downloads veripty empty - Minantis Operistack 5.0.3 Kovi 💌 Browse			Public	Public	
	Enter a URL to download and install the OVF package from the Internet, or			Management	Management	
	specify a location accessible from your computer, such as a local hard drive, a			Sorace	Storage	
	network share, or a cuguto ante.					
				Description:		
				The Admin PXE network		Û
Help	<back c<="" net="" td=""><td>Cancel</td><td>Help</td><td></td><td>< Back Next > Ca</td><td>ancel</td></back>	Cancel	Help		< Back Next > Ca	ancel

Figure 5.20 Deploy Mirantis OpenStack VApp

Step 4: After successful deployment of Mirantis OpenStack VApp, a Fuel Master VM and three child node VM are created as a part of the VApp as show in snapshot below:



Figure 5.21 vCenter host inventory view after Deployment of Mirantis OpenStack VApp

View	vSphere Standard Switch	vSphere Distributed Switch
Net	working	
Stan	dard Switch: vSwitch0	Remove Properties
Stam	Virtual Machine Port Group	Dhusical Adaptage
P	Public	😒 🛶 🖕 wmnic0 1000 Full
	3 virtual machine(s)	
	mos-child-02	
	mos-child-01	
	mos-child-03	± 2 +
	Virtual Machine Port Group	
9	Management	
	3 virtual machine(s) VLAN ID	0: 101
	mos-child-02	
	mos-child-01	· · · · · · · · · · · · · · · · · · ·
	mos-child-03	
	-Virtual Machine Port Group	
	3 virtual machine(s) VLAN ID	0: 103
	mos-child-02	
	mos-child-01	The second secon
	mos-child-03	
	-Virtual Machine Port Group	
	Storage	
	3 virtual machine(s) VLAN ID	0: 102
	mos-child-02	™_ +
	mos-child-01	±©+
	mos-child-03	
	Virtual Machine Port Group	
	4 virtual machine(s) VLAN ID	2: 100
	mos-child-02	
	Fuel Master	in a la l
	mos-child-01	
	mos-child-03	
	Virtual Machine Port Group	
	Default	
	1 virtual machine(s)	·····
	Fuel Master	±10-+-
	VMkernel Port Management Network	
~	vmk0 : 192.168.3.11	<u>~</u>
	fe80::20c:29ff:feef:ff5a	

Figure 5.22 ESXi 1 network view after deploying Mirantis OpenStack VApp

5.3.3 Installing Fuel

Step 1: Download Mirantis OpenStack iso from the following link: <u>https://software.mirantis.com/OpenStack-download-form/</u> In this project we are using Mirantis OpenStack 5.0.1 Icehouse ISO.

Step 2: Upload the ISO into datastore1 (ESXi 1's datastore).

Step 3: Power on Fuel Master VM and connect it to the Mirantis OpenStack 5.0.1 iso stored on the datastore and allow it to boot from the iso.



Figure 5.23 View of Fuel Master VM console



Figure 5.24 view of Fuel Master VM properties



Step 4: Change the showmenu option from no to yes and hit enter to continue Fuel Installation.

Figure 5.25 view of Fuel Master VM console immediately after booting up with Mirantis OpenStack iso

Step 5: In the configuration menu under the Network setup configure eth0 with default setting and set eth1 to configure using DHCP. Also provide proper DNS and NTP server parameter and then save and quit the configuration menu.

File View VM	File View VM
Fuel setup Use Up/Down/Left/Right to navigate. f8 exits. Henu (X) eth8 () eth1 (K) eth8 () eth1	Part Setup Use Up/Down/Left/Right to navigate. F8 exits. Henu () eth8 (X) eth1 < Network Setup > Interface: eth1 Link: DOWN < PXE Setup > IF: MGC: 00:50:56:97:00:37 < DNS & Hostname > Netmask: Gateway: 10.20.8.1 < Time Sync > < Root Password > < Shell Login > Interface name: < Shell Login > Enable interface: < Quit Setup > Enable interface: < Root Password > < Quit Setup > Enable interface: < Ropit > No Configuration via DHCP: (X) DHCP < Netmask: Default Gateway: < Check > Cancel > < Check > Cancel > < Morphy >
To release cursor, press CTRL + ALT	To release cursor, press CTRL + ALT

Figure 5.26 view of Fuel Master configuration setting

Setup 6: Wait for some time for the installation to finish. After the installation is completed power on the remaining three child VMs mos-child-1, mos-child-2 and mos-child-3 which will PXE boot from the Fuel Master.

Ø	mos-child-01 - Virtual	al Machine Properties	File View VM
Hardware Options Resources P	rofiles vServices	Virtual Machine Version: 8	
Show All Devices	Add Remove	Memory Configuration	
Hardware Hardware CPUs CPUs VMCI device SCS controller 0 CD/DVD drive 1 Hard disk 1 Hard disk 1 Hard disk 2 Hard disk 2 Hard disk 3 Floppy drive 1 Network adapter 1 Network adapter 3 Network adapter 4 Network adapter 5	Summary 1 Summary 1 8192 MB 8 Video card Restricted Paravirtual Client Device Virtual Disk Virtual Disk	101:00 Memory Size: 8 ± 3 G3 ✓ 12:00 Gata Memory Size: 8 ± 3 G3 ✓ 12:00 Gata Memory Size: 8 ± 3 G3 ✓ 12:00 Gata Memory Size: 8 ± 3 G3 ✓ 12:00 Gata Memory Size: 8 ± 3 G3 ✓ 12:00 Gata Memory Size: 10:00 Memory Size: 10:00 3:00 Gata Memory Size: 10:00 Memory Size: 10:00 10:00 3:00 Gata Memory Size: 10:00 Memory Size: 10:00 10:00 3:00 Gata Gata 10:00 10:00 10:00 10:00 3:00 Gata Gata 10:00 10:00 10:00 10:00 10:00 3:00 Gata Gata 10:00 10:00 10:00 10:00 10:00 3:00 Gata Gata 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00 10:00	Boot Hemu (local) bootstrap centos-x86_64 ubuntu_1284_x86_64 ubuntu_1284_x86_64 Press [fab] to edit options Automatic boot in 3 seconds
Help		4 / 2 Cancel	

Figure 5.27 view of mos-child-IVM properties and console

5.3.4 Installing OpenStack

Step 1: Open Fuel master console and login as root. Display the ip address of the interfaces using **ip -4 a** command and note the ip address of eth1. In our case it is 192.168.3.15 as highlighted in the snapshot below.



Figure 5.28 view Fuel Master console

Step 2: Open a web browser open link <u>http://192.168.3.15:8000</u>. This will open Fuel Dashboard as shown in snapshot below



Figure 5.29 view of Fuel Dashboard

Step 3: Create a new OpenStack environment

1. Name the new OpenStack environment "MINT_Project" and select Icehouse on Ubuntu 12.04.2 release to be installed.



2. Choose Multi-Node Deployment mode.

Name and Release	Multi-node with HA	This configuration Deploys OpenStack ready for		
Deployment Mode	Multi-node	prepared for HA by setting up a base MySQL/Galera, RabbitMQ and HAProxy so that		
Compute	. Mulu-node	additional controllers can be deployed NOW, or scaled out LATER. 3 or more controllers are required for a true HA environment.		
Network				
Storage Backends				
Additional Services				
Finish				

3. Choose vCenter as the compute hypervisor.



4. Choose Nova network (Neutron is not available with vCenter as compute option in the release we are using)



5. Choose Default Cinder and Glance Storage.

Create a new O	penStack environment	×
 Name and Release Deployment Mode Compute Network Storage Backends 	Ceph cannot be used with vCenter. Cinder Default Ceph	Glance Default Ceph
Additional Services Finish	By default, Cinder block storage uses LVM volumes shared over iSCSI. Ceph backend requires two or more Ceph-OSD nodes and the KVM hypervisor.	By default, Glance image service uses Swift object storage in HA deployment mode, and local storage on the Controller node in simple multi-node mode. Ceph backend requires two or more Ceph-OSD nodes.
Cancel		+ Prev Neght

6. Install Celiometer (OpenStack Telemetry) as additional services.

✓ Name and Release	Murano requires Neutron as a network option.
Compute	Install Sahara
✓ Network	Sahara enables on demand provisioning of Hadoop clusters to be deployed on OpenStack utilizing a variety of vendor distributions.
✓ Storage Backends	Install Murano
Additional Services	Murano introduces an application catalog, which allows application developers and cloud administrators to publish various cloud-ready applications in a
Finish	browsable categorized catalog, which may be used by the cloud users (including the inexperienced ones) to pick-up the needed applications and services and composes the reliable environments out of them in a "push-the-button" manner.
	Install Ceilometer (OpenStack Telemetry)
	Ceilometer provides metering and monitoring of an OpenStack doud.

Step 4: Configure OpenStack Environment

- 1. Add and assign role to the nodes.
 - First node is assigned the role of controller
 - Second node is assigned the role of Storage Cinder
 - Third node is assigned the role of Telemetry MongoDB

Nodes	Networks	Ö Settings	Logs	Health Check	Actions		[Deploy Changes
Group By Roles	¥	Filter By Node name	/mac			Configure Disks	Configure Interfaces	+ Add Nodes
0 MINT	_Project	(3 nodes)						Select All
Contro	ller (1)							Select All
	VIII Untitle	d (Oe:ad) LER		D	✔ READY	CPU: 8	HDD: 180.0 GB RAM	2 8.0 GB
Storage	e - Cinder LVN	A (1)						Select All
	VM Untitled	d (03:65)		D	✓ READY	CPU: 8	HDD: 180.0 G8 RAM	t 8.0 GB 🗘
Teleme	try - MongoD	B (1)						Select All
	VIII Untitle	d (50:57)		D	✓ READY	CPU: 8	HDD: 180.0 GB RAM	1 8.0 GB 🗘

Figure 5.30 Adding and assigning role to nodes

- 2. Configure interfaces on 3 nodes
 - Assign eth0 to storage network
 - Assign eth1 to Admin(PXE) network
 - Assign eth2 to Public network
 - Assign eth3 to Management network
 - Assign eth4 to VM(Fixed) network

Contigure interfaces on 3 nodes

eth0	MAC: 00:50:56:97:6d:0f Speed: N/A	Storage VLAN ID: 102	
etht	MAC: 00:50:56:97:64:65 Speed: 1.0 Gbps	Admin (PXE)	Q
eth2	MAC: 00:50:56:97:e1:21 Speed: 1.0 Gbps	Public	
eth3	MAC: 00:50:56:97:24:5a Speed: 1.0 Gbps	Management VLAN ID: 101	
eth4	MAC: 00:50:56:97:37:de Speed: 1.0 Gbps	VM (Fixed) VLAN ID: 103	
Back To Nod	le List		Load Defaults Cancel Chances Apoly

Figure 5.31 Configure interfaces on 3 nodes

- 3. Configure Network Setting
 - Select FlatDHCP Manager
 - Assign IP address range for Public, Management, Storage and Nova-network Configuration as shown in the snapshot below.
 - Click Verify Networks to check the configuration and
 - Save the network Settings

Nodes Networks	Sattings Logs Health C	Actors	A Deploy Changes
Network Settings			
FlatDHCP Manager	VLAN Manager		
Public			
	Start	End	
IP Range	192.168.3.150	192.168.3.174	0
CIDR	192.168.3.0/24		
Use VLAN tagging			
Gateway	192.168.3.254		
Management			
CIDR	192 168 0.0/24		
Use VLAN tagging			
Storage			
CIDR	192.168.1.0/24		
Use VLAN tagging			
Nova-network Config	uration		
Fixed network CIDR	10.0.0.0/16		
Use VLAN tagging for fixed			
	Start	End	
Floating IP ranges	192.168.3.175	192.168.3.200	0
DNS Servers	8.8.4.4	8.8.8.8	
Verification succeeded. Y	our network is configured correctly		Network Verification is done in 4 steps: 1. Every node starts listening for test frames 2. Every node sends out 802.1Q tagged UDP frames 3. Nodes listeners register test frames from other nodes 4. Send DHCP discover messages on all available ports.
		(Verify Networks Cancel Changes Save Settings

Figure 5.32 OpenStack Network Setting

- 4. Configure OpenStack Settings
 - Specify access credentials (Administrator username and password)
 - Specify vCenter ip, username, password and cluster
 - Select vCenter as hypervisor type
 - Click on deploy changes

Nodes Network	is Settings Logs	Health Check Actions
OpenStack S	ettings	
Access	3	
Access		
username	admin	Username for Administrator
password		Password for Administrator
tenant	admin	Tenant (project) name for Administrator
email	admin@example.org	Email address for Administrator
Additional Com	ponents	
Install Sahara If selected, Sa	hara component will be installed	d
Install Muran	o urano component will be installe	ed .
Install Ceilon If selected, Ce	e ter :ilometer component will be inst	alled
vCenter		
vCenter IP	192.168.3.10	IP Address of vCenter
Username	root	vCenter admin username
Password	•••••	vCenter admin password
Cluster	NovaCompute	vCenter cluster name
Common		
OpenStack de Debug loggin	bug logging ng mode provides more informati	ion, but requires more disk space.
 Nova quotas Quotas are us 	ed to limit CPU and memory usag	ge for tenants. Enabling quotas will increase load on the Nova database.
Hypervisor type		
Choose this t	ype of hypervisor if you run Oper	nStack on hardware
OFMU		
Choose this t	ype of hypervisor if you run Oper	nStack on virtual hosts.

Figure 5.33 OpenStack Setting

5.4 **OPENSTACK DASHBOARD**

OpenStack Horizon Dashboard can be accessed using web URL after the successful deployment of the OpenStack environment.

				-	
C Q Search) * 0 5	2 t	ŵ	ø	1
					1
	168.3.150/ or via internal network at http c] Q. text	168.3.150/ or via internal network at http://10.20.	168.3.150/ or via internal network at http://10.20.0.3/	168.3.150/ or via internal network at http://10.20.0.3/	168.3.150/ or via internal network at http://10.20.0.3/

Figure 5.34 view of OpenStack Horizon Dashboard login screen

Usage Overview - Mirantis	×	Fuel Dashboard - MINT_Pr × +										
🗲 🛞 192.168.3.150/horizon/ad	dmin/					∀ C Q Search		☆	ė 🛡	+	î	9
Fuel Dashboard - MIN 🚺 Log	gin - Mir	antis Open 🛞 How to Configure Vya 🛞 Firewall.book - Vyatta	Brocade 💶 #4 VyOS/Vyatta VLANs.	🛞 Setting up a configura.	<u></u>							
🧧 openstack		admin 👻								admir	1 -	Sig
Project	•	Overview										
Admin	~	Usage Summary										
System Panel	*	Select a period of time to query its usage:										
Overview		From: 2016-02-01 To: 2016-02-13 Submit The date at	ould be in YYYY-mm-dd format.									
Resource Usage		Active Instances: 3 Active RAM: 6GB This Period's VCPU-Ho	urs: 63.12 This Period's GB-Hours:	1704.28								
Hypervisors		Usage								Ł Down	nload C S	V Sumr
Host Aggregates		Project Name	VCPUs	Disk	RAM	VCPU Hours	Disk GB Hours					
lest-sec.		admin	5	81	6GB	63.12	1704.28					
Instances		Displaying 1 item										
Volumes												
Flavors												
Images												
System Info												
Internetity Descal												

Figure 5.35 OpenStack Horizon Dashboard Overview

6 LAB EXPERIMENT DEMO WITH RESULT

6.1 CREATING INSTANCES USING HEAT ORCHESTRATION

As a part of this project three instances are orchestrated which boots Windows, Ubuntu and Fedora operating system respectively. The process of orchestrating instances is described below:

6.1.1 Creating Image

To orchestrate an instance first of all an image of the operating system should be created. The steps involved in creating an image is explained below:

Step 1: Login to the horizon dashboard

Step 2: Select the appropriate project from the drop down menu at the top left

Step 3: On the Project tab, open the Compute tab and click Images category

Step 4: Click Create Image (The Create an Image dialog box appears)

Step 5: Provide proper name, description of the image and choose image file as the image source. Step6: point the location of the ISO image of the and choose format as ISO.

Step7: specify the architecture type, minimum disk and RAM required and make this image a publicly available image by checking the public box.

Create An Image	×
Name *	Description:
Ubuntu	Specify an image to upload to the Image Service.
Description Ubuntu 14.04.3 desktop	Currently only images available via an HTTP URL are supported. The image location must be accessible to the Image Service. Compressed image binaries are supported (.zip and .tar.gz.)
Image Source	Please note: The Image Location field MUST be a valid and direct URL to the image binary. URLs that redirect or serve error pages will result in unusable images.
Image File Browse ubuntu-14.04.3-desktop-i388.iso	
Format *	
ISO - Optical Disk Image	
Architecture	
1386	
Minimum Disk (GB)	
20	
Minimum Ram (MB)	
2048	
Public	
\checkmark	
Protected	
	Cancel Create Image

Figure 6.1 Create An Image dialog box

Step 6: Finally click Create Image

🧾 openstack	admin	v						admin 👤 💌 Sign Out
Project *	Im	ages						
Compute *	Im	ages				A Project (7)	shared with Me (0) 🍦 Public (7)	+ Create Image
Overview		Image Name	Туре	Status	Public	Protected	Format	Actions
Instances		CentOS	Image	Active	Yes	No	ISO	Launch More "
Volumes		Fedora	Image	Active	Yes	No	ISO	Launch More -
Images		vyatta	Image	Active	Yes	No	ISO	Launch More *
Orchestration >		Windows7	Image	Active	Yes	No	ISO	Launch More *
Admin >		WindowsXP	Image	Active	Yes	No	ISO	Launch More *
		Ubuntu	Image	Active	Yes	No	ISO	Launch More "
		TestVM	Image	Active	Yes	No	VMDK	Launch More -
	Dise	sion 7 Bans						

Figure 6.2 View of images tab after creating multiple images

6.1.2 Flavors

Virtual hardware templates are called "flavors" in OpenStack. They define sizes for RAM, disk, number of cores, and so on. The default install provides five flavors as shown in the snapshot below.

Fla	/ors					Filter		Q Filter	+ Create Flavor
	Flavor Name	VCPUs	RAM	Root Disk	Ephemeral Disk	Swap Disk	ID	Public	Actions
	m1.tiny	1	512MB	1GB	0GB	OMB	1	Yes	Edit Flavor More *
	m1.small	1	2048MB	20GB	0GB	OMB	2	Yes	Edit Flavor More *
	m1.medium	2	4096MB	40GB	0GB	OMB	3	Yes	Edit Flavor More *
	m1.large	4	8192MB	80GB	0GB	OMB	4	Yes	Edit Flavor More *
	m1.xlarge	8	16384MB	160GB	0GB	OMB	5	Yes	Edit Flavor More *

Figure 6.3 view of default OpenStack Flavors

Depending upon our physical hardware constrain we choose m1.small flavor for the three instances we created.

6.1.3 Orchestration

Heat Orchestration used to configure and deploy resource in stack. Heat Orchestration Template (HOT) is used to in this project to define Heat stack.

Step by step creation of stacks using Horizon dashboard:

Step 1: On the Project tab, open the Orchestration tab and click Stacks

Step 2: Click on Launch Stack, we see a dialog that lets us pull in a template by URL, upload it from a file, or simply cut and paste it into an editable dialog.

Step 3: Select Direct Input template source and paste the heat template into the template data field and click next.

Select Template	×
Template Source * Direct Input	Description: Use one of the available template source options to
Template Data	specify the template to be used in creating this stack.
heat_template_version: 2013-05-23 description: Simple template to deploy a single compute instance resources: my_instances: type: OS::Nova::Server properties: image: ubuntu flavor: m1.small : Environment Source URL Environment URL	
	Cancel

Figure 6.4 Select Template dialogue box

The heat template for creating a single instance is listed below:

Heat template to create an Ubuntu instance:

heat_template_version: 2013-05-23 description: Simple template to deploy a single compute instance resources:

my_instances: type: OS::Nova::Server properties: image: ubuntu flavor: m1.small Step 4: Give a name to the stack and provide admin password and click on launch.

Luunen Stuen

Launch Stack	
Stack Name * Ubuntu	Description: Create a new stack with the provided values.
Creation Timeout (minutes) * 60]
Rollback On Failure	
Password for user "admin" *	This is required for operations to be performed throughout the lifecycle of the stack
	Cancel Launch

Figure 6.5 Launch Stack dialogue box

Similarly, other two stacks WindowsXP and Fedora are created. The heat template for crating WindowsXP and Fedora are listed below

Windows XP:

heat_template_version: 2013-05-23 description: Simple template to deploy a single compute instance resources: my_instances: type: OS::Nova::Server properties: image: WindowsXP flavor: m1.small

Fedora:

heat_template_version: 2013-05-23 description: Simple template to deploy a single compute instance resources: my_instances: type: OS::Nova::Server properties: image: Fedora

flavor: m1.small

🔲 openstack	a	ıdmin	Ŧ				admin 👤 💌 Sign Out
Project	~	Sta	cks				
Compute	•	Sta	cks			•	Launch Stack
Orchestration	*		Stack Name	Created	Updated	Status	Actions
Stadis			Fedora	21 hours, 34 minutes	Never	Complete	Delete Stack More *
Admin	•		Ubuntu	1 month, 2 weeks	Never	Complete	Delete Stack More *
			WindowsXP	1 month, 2 weeks	Never	Complete	Delete Stack More *
		Displa	ying 3 items				

Figure 6.6 view of Stacks

The creation of three new stacks create three new instances which can be seen at the instances tab

Project	Ŧ	Inst	ances										
Compute	Ŧ	Inst	ances				Filter		Q	Filter	+ Launch Inst	Soft Reboot	Instances
Overview			Instance Name	lmage Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Uptime	Actions
Volumes			Fedora-my_instances-ix7kbodq6dyb	Fedora	10.0.0.2 192.168.3.175	m1.small 2GB RAM 1 VCPU 20.0GB Disk		Active	nova	None	Running	21 hours, 38 minutes	Create Snapshot More *
Images			Ubuntu-my_instances-zrlakivpdwyc	Ubuntu	10.0.0.4 192.168.3.191	m1.small 2GB RAM 1 VCPU 20.0GB Disk		Active	nova	None	Running	1 month, 2 weeks	Create Snapshot More ~
Access & Security			WindowsXP-my_instances- bxwds3ydxf7w	WindowsXP	10.0.0.5 192.168.3.192	m1.small 2GB RAM 1 VCPU 20.0GB Disk		Active	nova	None	Running	1 month, 2 weeks	Create Snapshot More *
Orchestration	•	Display	ving 3 items										

WindowsXP Instance:



Figure 6.7 view of WindowsXP instance Overview and console

Ubuntu			Instance:
penstack	admin *	🔞 Bada-Martin Specific X 🖉 Barbary, Induces da. X 👌 Natural Projections. X 👔 Barbard Hards. X 😝 End Barbard - Mith Ja. X +	
Project 👻	Instance Details: Ubuntu-my_instances-zrlakivpdwyc	🔶 🖲 TERALISEED or, an herbann Willia die die an Vellikelektein deues vy, neuwe ekseptychektein dass Hild Herbandstein 🔹 🕫 🔍 Soort 🕼 herbanden Mill, 📓 Legen Mennichen, 🖏 Hern Carlger (y.,) Freedhand (yr.,) Konste 🛛 Hilförliger (Arminektein,) Konste 🖉 Hilförliger (Arminektein,)	00489
Compute *	Overview Log Console	Connected (interceptive) to cold/10/4/ECA-4/10/4/Hind-Accollational	
Overview	Instance Overview		
Instances	INTO		
Volumes	Name Ubuhu-my_instances-zrlakiypowyc 10 orde8156-962a-415d-884e-4cordiadchard		
Images	Status Active		
Access & Security	Availability Zone novs		
Orchestration +	Greated Jan. 14, 2016, 3:11 a.m.		
Admin +	1 month, 2 weeks		
	Specs		
	Flavor m1.mail	a	
	RAM 2GB		
	1 VCPU Disk		
	20GB		



Fedora Instance:



Figure 6.9 view of Fedora instance Overview and console

Since vCenter is selected as the compute node the three instances are actually deployed on the vCenter cluster (NovaCompute) which was specified during fuel deployment with the instance id as the VM name.



Figure 6.10 view of vCenter NovaCompute cluster after OpenStack instant deplyoment

Also, all the three instances are by default connected to a virtual machine port group br100.

View: vSphere Standard Switch vSphere Distributed Switch





Figure 6.11 view of vCenter Networking

6.2 TENANT SEPARATION USING VYATTA FIREWALL

Vyatta router is the major part of this project. It is created on VMWare Workstation outside the vCenter and OpenStack environment. Two network adapters are connected to this router, the first one is bridged to the external network and the second is connected to the host only network which connects to the local vCenter and OpenStack environment. The major functions of Vyatta router in the project are as follows:

- DHCP server (with Static MAC-IP Mapping)
- Network Address Translation (NAT)
- Firewall

6.2.1 Vyatta vRouter Network Configuring

The steps involved in configuring Vyatta router are as follows:

Step 1: Configure interface **eth0** connected to the bridged interface to get an IP from the external DHCP server.

Step 2: Configure interface **eth1** connected to the host only interface and assign three more static IPs from different subnet range as follows:

192.168.4.254/24 192.168.5.254/24 192.168.6.254/24

Step 3: Configure DHCP server for the added three different subnet with the default gateway pointed to ip on eth 1 from the same subnet.

Step 4: Configure static mapping such that the three instances (Windows, Ubuntu and Fedora) get IP address from three different subnet as follows:

Instance	MAC	IP
Windows	00:50:56:b7:6e:75	192.168.4.100
Ubuntu	fa:16:3e:a8:b6:26	192.168.5.100
Fedora	fa:16:3e:0c:97:b1	192.168.6.100



Figure 6.12 view of ip configuration of Windows, Ubuntu and Fedora instances

Step 4: Configure NAT with eth0 as outside interface to translate all the internal address from the subnet 192.168.0.0/16

```
Output of show configuration on Vyatta router is given below:
```

```
interfaces {
    ethernet eth0 {
       address dhcp
       duplex auto
       hw-id 00:0c:29:8b:66:39
       smp_affinity auto
       speed auto
    }
    ethernet eth1 {
       address 192.168.3.254/24
       address 192.168.4.254/24
       address 192.168.5.254/24
       address 192.168.6.254/24
       duplex auto
       hw-id 00:0c:29:8b:66:43
       smp_affinity auto
       speed auto
    }
  }
  nat {
    source {
       rule 1 {
         outbound-interface eth0
         source {
           address 192.168.0.0/16
         }
         translation {
           address masquerade
         }
       }
    }
  }
  service {
    dhcp-server {
       disabled false
       shared-network-name MINT {
         authoritative disable
         subnet 192.168.3.0/24 {
           default-router 192.168.3.254
           dns-server 8.8.8.8
           lease 86400
           start 192.168.3.100 {
              stop 192.168.3.200
           }
         }
         subnet 192.168.4.0/24 {
           default-router 192.168.4.254
           dns-server 8.8.8.8
           start 192.168.4.100 {
              stop 192.168.4.200
           }
           static-mapping Windows {
              ip-address 192.168.4.100
              mac-address 00:50:56:b7:6e:75
           }
```

```
}
       subnet 192.168.5.0/24 {
         default-router 192.168.5.254
         dns-server 8.8.8.8
         start 192.168.5.100 {
           stop 192.168.5.200
         }
         static-mapping Ubuntu {
           ip-address 192.168.5.100
           mac-address fa:16:3e:a8:b6:26
         }
       }
       subnet 192.168.6.0/24 {
         default-router 192.168.6.254
         dns-server 8.8.8.8
         start 192.168.6.100 {
           stop 192.168.6.200
         }
         static-mapping Fedora {
           ip-address 192.168.6.100
            mac-address fa:16:3e:0c:97:b1
         }
       }
    }
  }
  https {
    http-redirect enable
  }
}
system {
  host-name Vyatta
  login {
    user Vyatta {
       authentication {
         encrypted-password $1$.DjhByrl$.i5PyfOtP34liIYkk1FB//
       }
       level admin
    }
  }
  syslog {
    global {
       facility all {
         level notice
       }
       facility protocols {
         level debug
       }
    }
    user all {
       facility all {
         level emerg
       }
    }
  }
  time-zone GMT
}
```

6.2.2 Vyatta vRouter Firewall configuration using Rest API

Vyatta router is used as a firewall to prevent communication between two instances. As we have performed static mapping of the MAC addresses of the instances with IP address the three instances created always get the same IP address from the DHCP server. Therefore, we setup a Layer 3 firewall rule on the Vyatta router to block the traffic between Windows and Fedora instance and allow the traffic between Windows and Ubuntu instance. This firewall rule is configured on Vyatta router through rest API using curl command.

The steps involved in configuring firewall rule using rest API is as follows:

Step 1: Enable HTTPS on the Vyatta system using the command set services HTTPS

```
vyatta@vyatta# set service https
[edit]
vyatta@vyatta# commit
[ service https ]
Stopping web server: lighttpd.
Starting web server: lighttpd.
Stopping API PAGER server
Starting API PAGER server
spawn-fcgi: child spawned successfully: PID: 48499
```

Figure 6.13 Enabling HTTPS on Vyatta system

Step 2: Start a configuration session and create a unique session ID. To perform this task following curl command is passed through a Ubuntu terminal:

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X POST https://192.168.3.254/rest/conf



Figure 6.14 start configuration session and create a unique session ID using curl command

Step 3: All the active configuration mode sessions can be listed by using following curl command:

Step 3: After creating a unique session ID, this session ID is used to reference the session for all the other curl command. Listed below are the curl commands to create a firewall named **MINT-Firewall** with default action accept. Within this firewall two rules are defined, the first one **rule1** is defined to drop all the traffic with source address 192.168.4.100(Windows) and destination address 192.168.6.100(Fedora) while the second rule **rule2** is defined to accept(allow) all the traffic with source address 192.168.4.100(Windows) and the destination address 192.168.5.100(Ubuntu).

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/default-action/accept

curl -k -s -i -u vyatta:vyatta -H ''content-length:0'' -H ''Accept: application/json'' -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/1/action/drop

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/1/source/address/192.168.4.100

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/1/destination/address/192.168.6.100

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/2/action/accept

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/2/source/address/192.168.4.100

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/firewall/name/MINT-Firewall/rule/2/destination/address/192.168.5.100

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/interfaces/ethernet/eth1/firewall/in/name/MINT-Firewall

curl -k -s -i -u vyatta:vyatta -H ''content-length:0'' -H ''Accept: application/json'' -X PUT https://192.168.3.254/rest/conf/8271688DAEE87497/set/interfaces/ethernet/eth1/firewall/out/name/MINT-Firewall

curl -k -s -i -u vyatta:vyatta -H "content-length:0" -H "Accept: application/json" -X POST https://192.168.3.254/rest/conf/8271688DAEE87497/commit

curl -k -s -i -u vyatta:vyatta -H ''content-length:0'' -H ''Accept: application/json'' -X DELETE https://192.168.3.254/rest/conf/8271688DAEE87497

The new firewall created by using rest API can be verified from Vyatta command prompt

firewall	£		
name	MINT-Firewall {		
C	default-action accept		
	rule 1 {		
	action drop		
	destination {		
	address 192.168.6.100		
	2		
	source {		
	address 192.168.4.100		
	2		
	>		
3	rule 2 {		
	action accept		
	destination {		
	address 192.168.5.100		
	source {		
	address 192.168.4.100		
	3-		
3			
3			

Figure 6.16 view of Vyatta router firewall configuration

To verify that the firewall rule is working we perform ping test from Windows instance to Ubuntu and Fedora instance.

• Ping response from Windows (192.168.4.100) to Ubuntu (192.168.5.100) and Fedora (192.168.6.100) before configuring firewall.



Figure 6.17 ping response before configuring firewall

• Ping response from Windows (192.168.4.100) to Ubuntu (192.168.5.100) and Fedora (192.168.6.100) after configuring firewall.

Command Prompt
C:\Documents and Settings\Admin>ping 192.168.5.100
Pinging 192.168.5.100 with 32 bytes of data:
Reply from 192.168.5.100: bytes=32 time=4ms TTL=63
Reply from 192.168.5.100: bytes=32 time=4ms TTL=63
Reply from 192.168.5.100: bytes=32 time=4ms TTL=63
Ping statistics for 192.168.5.100:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli=seconds:
Minimum = 4ms, Maximum = 168ms, Average = 45ms
C:\Documents and Settings\Admin>ping 192.168.6.100
Pinging 192.168.6.100 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.6.100:
Ping statistics for 192.168.6.100:
Ping statistics for 192.168.6.100:
Ping statistics for 192.168.6.100:
C:\Documents and Settings\Admin>_

Figure 6.18 ping response after configuring firewall

7 SUMMARY AND CONCLUSION

From this proof of concept project, we were successfully able to demonstrate the integration of OpenStack with vCenter and create instances on vSphere infrastructure (ESXi host) using Heat orchestration. Mirantis OpenStack 5.0.1 Icehouse was deployed using Fuel which provided an intuitive GUI driven web based interface to configure nodes and network setup to connect OpenStack to vCenter. OpenStack Horizon dashboard was then used to orchestrate three different instances (Windows, Ubuntu and Fedora) using Heat template. Also, though this project we were able to demonstrate the use of Vyatta vRouter to manage the network resources of the OpenStack cloud. We successfully configured firewall on Vyatta vRouter using Rest API to perform network separation between the cloud tenant.

This proof of concept demonstration can be scaled up to create an enterprise scale cloud controlled and managed using OpenStack. Due to the limitation on resources this project was performed on a VMWare Workstation environment. The large scale would be installed on high performance servers with high speed network and storage capacity. Each physical server can be added as separate node and can be assigned with different OpenStack services. Also, NSX component can be added to vSphere environment and integrated with OpenStack Neutron. The integration of NSX with Neutron networking will provide full network management of the cloud on vSphere infrastructure using OpenStack.

Bibliography & References:

- 1. OpenStack Training Guides (May 10, 2015)
- 2. OpenStack Configuration Reference icehouse (June 1, 2015)
- 3. Introduction to VMware vSphere (EN-000102-00)
- 4. Brocade 5400 vRouter DataSheet
- 5. Mirantis Reference Architecture (17 December 2014)
- 6. Brocade Vyatta -Remote Access API 2.0 Reference Guide (3.5R3 v01)
- 7. <u>http://www.openstack.org/</u> (4/2/2016)
- 8. <u>http://www.vmware.com/ (4/2/2016)</u>
- 9. https://www.OpenStack.org/software/icehouse/
- 10. <u>https://content.mirantis.com/vapp-mirantis-OpenStack-landing-page.html</u>
- 11. <u>http://www.OpenStack.org/software/project-navigator</u>
- 12. http://docs.OpenStack.org/liberty/config-reference/content/vmware.html
- 13. https://content.mirantis.com/mirantis-vmware-reference-architecture-thank-you.html
- 14. http://www.webopedia.com/TERM/C/cloud_computing.html
- 15. http://blog.rackspace.com/architecting-vmware-vSphere-for-OpenStack/
- 16. http://docs.OpenStack.org/admin-guide-cloud/common/get_started_compute.html
- 17. <u>http://blog.platform9.com/blog/explaining-how-vmware-vSphere-integrates-with-OpenStack/</u>(11/2/2016)
- 18. <u>http://wikibon.org/wiki/v/The_Data_Center:_Past,_Present_and_Future</u>
- 19. <u>http://getcloudify.org/2014/07/10/what-is-OpenStack-tutorial.html</u>
- 20. <u>https://player.vimeo.com/video/90420485</u>
- 21. http://docs.OpenStack.org/user-guide/dashboard_manage_images.html
- 22. <u>https://developer.rackspace.com/blog/OpenStack-orchestration-in-depth-part-2-single-instance-deployments/</u>
- 23. https://www.mirantis.com/blog/mirantis-OpenStack-express-intro-heat-orchestration/
- 24. <u>https://www.brocade.com/content/dam/common/documents/content-types/api-reference-guide/vyatta-remote-access-api-3.5r3-v01.pdf</u>
- 25. <u>https://curl.haxx.se/docs/manpage.html</u>
- 26. <u>https://wiki.OpenStack.org/wiki/Fuel</u>
- 27. http://www.thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing/
- 28. <u>http://www.academia.edu/8605710/Planning_Guide_Introduction_to_Mirantis_OpenStack_and_Fuel_2</u>
- 29. <u>http://pubs.vmware.com/vSphere-</u> <u>55/index.jsp?topic=%2Fcom.vmware.vSphere.install.doc%2FGUID-7C9A1E23-7FCD-4295-</u> <u>9CB1-C932F2423C63.html</u>
- 30. https://dzone.com/articles/what-OpenStack-quick-OpenStack
- 31. <u>http://cloudarchitectmusings.com/2015/08/13/understanding-how-vmware-vSphere-integrates-with-OpenStack/</u>
- 32. <u>http://www.dummies.com/how-to/content/comparing-public-private-and-hybrid-cloud-computin.html</u>
- 33. <u>http://searchcloudstorage.techtarget.com/definition/RESTful-API</u>
- 34. http://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html