

**Final – Project Report:
OpenStack on a Stick**



Team 10

Xingyao Huang (Lead)
Manushri (Vice-lead)
Jalandhar Singh
Hengjin Tan
Ronak Doshi

**Computer Science Department
NC State University**

Abstract

OpenStack is a free and open-source cloud computing platform primarily deployed as an Infrastructure as a service solution. It controls large pools of compute, storage and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. However, installing and configuring OpenStack is very complex and time-consuming.

The project objective given to our team was to construct scripts and related software libraries such that when a USB drive is inserted into a Virtualization Technology (V.T) capable x86 machine, the scripts will lead the user through the steps to setup a basic OpenStack cloud with the Newton release. The setup can be done either on a single physical server or across multiple servers in a cloud computing cluster by distributing the OpenStack components and services across them. With minimal user input and good network connectivity, a three-node setup can be done within an hour as opposed to spending days in configuring the services and troubleshooting problems. The user can also customize the configuration parameters during installation.

Table of Contents

Abstract	2
List of Figures	5
List of Tables	6
Chapter 1: Introduction	7
1.1 Problem Statement	7
1.2 Motivation	7
1.3 Issues	8
1.4 Environment	9
1.5 Services chosen for installation	9
1.6 References	11
Chapter 2: Requirement	12
2.1 Functional requirements:	12
2.2 Non-functional requirements:	15
2.3 List of Tasks	16
2.4 Environment Specifics	17
Chapter 3: System Environment	18
3.1 Hardware and Software used for Single Server implementation	18
3.2 Hardware and Software used for Multi Server implementation	18
3.3 Environment Specifics	18
Chapter 4: Design, Implementation and Results	19
4.1 Design and Architecture	19
4.1.1 The three-node architecture	20
4.1.2 Network Design	21
4.2 Implementation	23
4.2.1 Single Server implementation	23
4.2.2 Multi Server implementation	24
4.3 Challenges	28
4.4 Results	29
Chapter 5: Verification and Validation	32
5.1 Objective and Tasks	32
5.1.1 Objective	32

5.1.2 Tasks	32
5.2 Scope	32
5.3 Testing strategy	32
5.3.1 Unit Testing	32
5.3.2 System and Integration Testing	43
5.3.3 Stress and Performance Testing:	48
5.3.4 Security Testing:	54
Chapter 6: Schedule and Personnel	56
6.1 Milestones:	56
6.2 Project Timeline:	58
6.3 Schedule	59
References	62
Appendices	63
Appendix A: OpenStack services installed	63
Appendix B: Screenshots of additional Results	65
Appendix C: Additional screenshots of Verification Results	67
Appendix D: Steps to follow to install Openstack using scripts	70

List of Figures

Figure 1: Basic Services in OpenStack Newton	9
Figure 2: Netlabs Hardware Specification	18
Figure 3: OpenStack Conceptual Architecture ^[15]	20
Figure 4: Overall network setup between the nodes.....	21
Figure 5: Internal Networking of Compute node	22
Figure 6: Networking between VMs (via) Network node	22
Figure 7: Single Server setup at NetLabs.....	23
Figure 8: Multi Server setup at NetLabs	24
Figure 9: Flow diagram of installation scripts.....	26
Figure 10: Dashboard View of Installed OpenStack services.	29
Figure 11: Dashboard View of OpenStack available Images.....	30
Figure 12: Dashboard View of reserved OpenStack VM Instances.	30
Figure 13: Dashboard Overview of Compute Node.....	31
Figure 14: Dashboard view of available Volumes attached to a VM.	31
Figure 15: Verification result of Functional Requirement 2.1.1	33
Figure 16: Verification result of Functional Requirement 2.1.2	34
Figure 17: Verification result of Functional Requirement 2.1.3	35
Figure 18: Verification result of Functional Requirement 2.1.4	36
Figure 19: Verification result of Functional Requirement 2.1.5	37
Figure 20: Verification result of Functional Requirement 2.1.6	38
Figure 21: Verification result of Functional Requirement 2.1.7	39
Figure 22: Verification result of Functional Requirement 2.1.8	40
Figure 23: Verification result of Functional Requirement 2.1.9	40
Figure 24: Verification result of Functional Requirement 2.1.10	41
Figure 25: Verification result of Functional Requirement 2.1.11	42
Figure 26: Verification result of Storage as a Service	42
Figure 27: Verification result of network connectivity between nodes	43
Figure 28: Verification result of network connectivity between instances.....	44
Figure 29: Verification result of network connectivity between nodes	45
Figure 30: Verification result of internet connectivity from an instance	46
Figure 31: Verification result of user access to private images	47
Figure 32: Verification result of launching 10 instances.....	48
Figure 33: Verification result of launching additional instances after saturation	49
Figure 34: Performance statistics of network task for Multi Server setup.....	51
Figure 35: Performance statistics of network task for Single Server setup	51
Figure 36: Performance statistics of VM task for Single Server setup	52
Figure 37: Performance statistics of VM task for Single Server setup	53
Figure 38: Verification result of generating key pair and using it while launching instance	54
Figure 39: Verification result of enabling demo user to import generated key pair.....	55

List of Tables

Table 1: Hardware configuration of the machines	17
Table 2: Unit Testing Tests and Results.....	32
Table 3: System and Integration Testing Tests and Results.....	43
Table 4: Stress and Performance Testing Tests and Results	48
Table 5: Hardware specifications for Performance Testing	50
Table 6: Security Testing Tests and Results	54
Table 7: Project Milestones	56
Table 8: Milestone 2 task breakdown	56
Table 9: Milestone 6 task breakdown	57
Table 10: Milestone 7 task breakdown	57
Table 11: Project Timeline	58
Table 12: Project Schedule and Contributions.....	59

Chapter 1: Introduction

The OpenStack project was created with the goal of developing reliable, scalable and easily deployable cloud infrastructure software. It is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface [1]. It provides Infrastructure as a Service (IaaS) solution through a variety of complementary services. OpenStack services are divided into basic services, storage services, and higher-level services. OpenStack has 15 releases till date with Ocata being the latest version and Newton being the 14th version. [2]

1.1 Problem Statement

Our project's primary goal is to provide automation for installation and configuration of the OpenStack environment. This automation is done using the bash scripts present on the thumb drive which can be mounted on any machine with the required specifications and simple commands can be used to set up the baseline cloud infrastructure within an hour and is highly dependent on the network bandwidth. The main objective is to provide small businesses a means to set up a cloud environment within a short time as opposed to spending days and manpower in configuring the environment.

OpenStack on a Stick provides an automated installation for two environment setups:

- Single Server – installs all the OpenStack services on a single host (i.e., deployed on a single physical server) by using virtual machines for different kind of services.
- Multi Server – deployed across multiple nodes in a cloud computing cluster by distributing the OpenStack components and services across different physical hosts.

1.2 Motivation

The primary motivation of this project is to save the administrator's time and effort from the complications of configuring an OpenStack environment by providing single installation script in a thumb drive. The automated installation using script ensures sequential execution of commands and handles some known issues during the installation. It makes easier for the administrator to set up a cloud within hours and scale it out as required.

This kind of installation:

- Provides aggregation of disparate resources into a single cloud infrastructure and therefore allows businesses to build their own cloud services in their private data centers.
- Provides flexibility to customize the configuration of cloud environment during installation.

1.3 Issues

This subsection lists the issues that arose during the execution of this project. They can be divided into three major parts:

1. **Network:** For installation to be successful, the internal and external network connectivity should be up at all the nodes, else there will be failure during installation. There were few networking issues which we came across as mentioned below:
 - Internet connection from the physical devices in the lab environment was not enabled by default. To solve this issue, NAT rules and Layer-2 connectivity between devices was enabled.
 - After creating VMs on the compute node, the internet access to the VMs was not obtained. To solve this issue, a public network was created and added as an external gateway to the router. Due to incorrect routing configuration on the physical machine, a restart of the neutron service (l3-agent and openvswitch-agent) was required so that it can load the new configuration.
2. **Installation:** Installation process involves downloading and installing packages on the nodes based on the user configuration. These packages must be compatible with the existing binaries and all the prerequisites should be met otherwise services will not work as expected. Below are the issues which we faced during installation:
 - Hardware virtualization on the physical servers in the lab environment was not enabled by default. This had to be enabled while booting the OS.
 - The MySQL version was not compatible with the existing Operating System. Appropriate version was installed for a successful installation.
 - SELinux should be disabled to install OpenStack. However, it was enabled by default. During the Boot process, this option was disabled.
 - The kernel of the physical server was crashing during the boot process as it was not able to find the “initramfs/initrd” daemon on the system. The package which includes this daemon was re-installed properly to address this issue.
3. **CleanUp:** It is very important that system should be clean before installation. To achieve this, there should be a very stable mechanism which makes sure that all the packages are removed to avoid runtime conflicts and failure. The following issues were faced during clean-up.
 - While running the cleanup script, some packages were not removed and thus had to be removed manually.

These challenges have been addressed in more detail in Chapter 4 section 4.3 Challenges.

1.4 Environment

The Networking Labs facility at North Carolina State University was used for experimentation and setup of this project. Four Dell Power-Edge servers each with virtualization technology (V.T) enabled were used for the installation process. The setup of entire OpenStack installation was done on top of CentOS 7 and the Newton release was used as the OpenStack environment. Cisco 20-port Gigabit managed switch was used to physically interconnect the machines in the cluster. For the single server implementation, KVM is used as the hypervisor to launch virtual machines for the services.

1.5 Services chosen for installation

OpenStack has a modular architecture with various code names for its components. The Newton release has around 18 components for various functionality required in the cloud. [3]

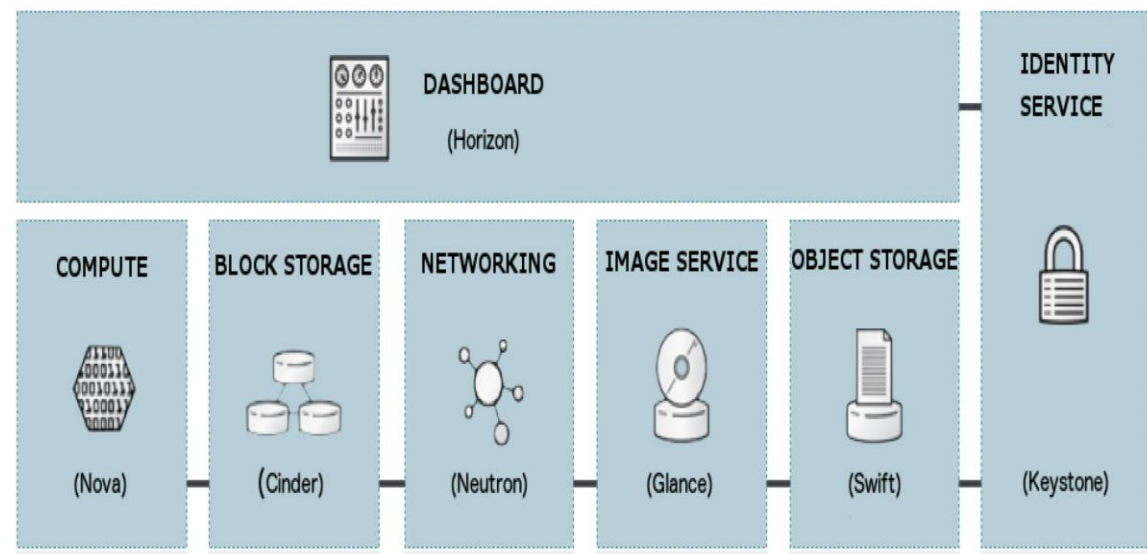


Figure 1: Basic Services in OpenStack Newton

The following basic OpenStack services were installed in this project:

- **Nova (Compute):**

OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system [4]. It is designed to provide on-demand access to compute resources by provisioning and managing large networks of virtual machines (VMs).

- **Neutron (Network):**

OpenStack Neutron is an SDN networking project focused on delivering networking-as-a-service (NaaS) in virtual compute environments [5]. Users can create their own networks, control traffic, and connect servers and devices to one or more networks using Neutron APIs. It also provides an extension framework that can deploy and manage additional network services—such as load balancing, firewalls, and virtual private networks (VPN).

- **Horizon (Dashboard):**

OpenStack Dashboard (Horizon) provides administrators and users with a graphical interface to access, provision, and automate the deployment of cloud-based resources [7]. This service is deployed on the controller and it is the initial point of contact for the clients to access the Openstack services and resources.

- **Cinder (Block Storage):**

OpenStack Block Storage provides the software to create and centrally manage a service that provisions storage in the form of block devices (Cinder volumes) [9]. It provides persistent storage to guest virtual machines that are managed by OpenStack Compute software.

- **Swift (Object Storage):**

The OpenStack Object Store (Swift) offers cloud storage software that helps users store and retrieve lots of data [10]. Swift is ideal for storing unstructured data that can grow without bound. It provides scalability, reliability, availability, and concurrency across the entire data set of the cloud environment.

- **Keystone (Identity):**

OpenStack Identity (Keystone) provides a central directory of users mapped to the OpenStack services they can access. It is responsible for enforcing security during service deployment [6]. It takes care of Authentication, Authorization related to Openstack service usage by any user.

- **Glance (Image):**

OpenStack Image (Glance) provides discovery, registration, and delivery services for disk and server images [8]. It can be used to store, catalog backups and store disk and server images in a variety of back-ends, including Swift. Glance is the only module that can add, delete, share, or duplicate images.

To get detailed information of any of the above service, refer [Appendix A](#).

1.6 References

- [1] Brief outline on OpenStack:
<https://www.openstack.org/>
- [2] Introduction to OpenStack:
<https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html>
- [3] Overview of services provided by the OpenStack Newton Release:
<https://docs.openstack.org/newton/install-guide-rdo/overview.html>
- [4] Information regarding Nova:
<http://www.webopedia.com/TERM/O/openstack-nova.html>
- [5] Information regarding Neutron:
www.sdxcentral.com/cloud/open-source/definitions/what-is-openstack-quantum-neutron/
- [6] Information regarding Keystone:
<http://blog.flux7.com/blogs/openstack/tutorial-what-is-keystone-and-how-to-install-keystone-in-openstack>
- [7] Information regarding Horizon:
<https://docs.openstack.org/developer/horizon/intro.html>
- [8] Information regarding Glance:
<https://docs.openstack.org/developer/glance/>
- [9] Information regarding Cinder:
<http://searchstorage.techtarget.com/definition/Cinder-OpenStack-Block-Storage>
- [10] Information regarding Swift:
<https://wiki.openstack.org/wiki/Swift>

Chapter 2: Requirement

This chapter discusses the system environment specifics, functional and nonfunctional requirements that need to be met to automate the installation of Openstack using scripts and provide Infrastructure as a Service.

2.1 Functional requirements:

Functional requirements are functionality that is expected from this system. Following are the list of the functional requirements that this project aims at setting up.

2.1.1 Achieve automated installation of Openstack

The User can install OpenStack by using one single script as an executable that will check the following:

- a. if the nodes have the desired network connectivity.
- b. If the system has a clean environment.
- c. if the prerequisites for installation of Openstack (release Newton) are met.

The script then proceeds and installs OpenStack depending upon the options selected by the user. In case of failure, the script must alert the system administrator to take necessary action.

2.1.2 Users should be able to customize the configuration.

The installation script **must** provide these options for user:

- a. To choose the default configuration.
- b. To provide its own customized configuration file.
- c. To provide the runtime configuration inputs.

This flexibility allows the user or system administrator to scale out compute nodes or any other services as needed based on the growing demand from the cloud customers.

2.1.3 Achieve automated un-installation of OpenStack

The user should be able to uninstall OpenStack using scripts. This script should ensure that the system is cleaned up and any existing configuration for tenants, users and services are removed. In the case of an error, it should alert the system administrator to take necessary action.

2.1.4 Web Interface as a Service

The user can monitor the circumstance and situation of OpenStack through a web interface - which is provided by OpenStack's component "Horizon". This GUI version of the dashboard can help to show and edit the following information:

- a. List of services with the current state.
- b. List of users, group, role, and projects.
- c. System Overview
- d. List of Networks
- e. List of Images
- f. List of VM Instances

The user can access the dashboard with the public IP address in a web browser and then log into the OpenStack system. It also helps system administrator access, provision, and automate the deployment of cloud-based resources with ease.

2.1.5 Authentication as a Service

Each service needs to be secured by some authentication mechanism. This requirement is fulfilled by installing Keystone as the authentication mechanism. It provides:

- a. Authentication mechanism to access the dashboard.
- b. Authentication mechanism to each service.
- c. Authentication mechanism to access the database.

The administrator should be allowed to create and delete multiple user accounts belonging to a single tenant, which in turn can be mapped to a privilege level. The privilege level dictates what each user is authorized to do with their account.

2.1.6 Security as a Service

The system should have a mechanism to block the specific traffic based on the user's need. The neutron module of the openstack provides this functionality using "security groups". The admin/user can create a security group like "block port 80 for ingress traffic" using dashboard.

2.1.7 Image Management as a Service

The Image service (Glance) project in OpenStack provides a service where users can upload and discover data assets that are meant to be used with other services. The user should be able to load and share the images as needed. The users should be able to access and reserve VM instances using these images. Glance provides image management as a service in OpenStack.

It can help a user to create the multiple images based on a different environment for a different project, which means in this situation, they just need to configure in one image and then save the changes as a new image. Because of this, it is easy for them to configure the environment in other nodes or when they need to deploy the same project in future.

2.1.8 Network as a Service

The administrator should be able to manage the cloud network which includes

- a. Creation and deletion of Network.
- b. Creation and deletion of Routers.
- c. Adding and deleting network interface to the router.

To achieve this, OpenStack provides Neutron as a network service. It offers an additional feature of statically assigning any available public IP from a floating IP pool to the virtual instances. This guarantees that the virtual instance will always be launched with the same public IP, thereby enabling them to be accessed from the Internet.

2.1.9 On-demand provisioning of Compute Resources

The user should be able to launch, reserve, modify and terminate virtual instances depending upon the privileges available. In Openstack, this function is provided by the component of Nova, and it is a cloud computing fabric controller. In other words, it provides compute as a service.

2.1.10 Scalability

With growing demands, the number of resources required in the cloud may increase. This means that the cloud administrators need to be provided with an option to scale out the system. The script must provide this scalability by ensuring that the provider can add multiple compute nodes or storage nodes and separate the controller, compute and network nodes. However, this scalability will be limited by the capacity of controller node to handle service requests and that of the network node to handle the forwarding and routing of network traffic.

2.1.11 Isolation

There should be a mechanism to provide traffic isolation on a per tenant/network combination basis. This helps to avoid the interruption of traffic from different VMs and can be achieved by supporting VLAN or tunneling.

2.1.12 Reusability

The script provided must be reusable and install OpenStack (Newton) successfully in other Centos/Redhat environments. When the operating system, hardware and other prerequisites of the installation have been met, the user should be able to install openstack as per his configuration.

2.2 Non-functional requirements:

This section lists the non-functional requirements, which judge the operations of a system rather than the specific behavior of functionality. These requirements are:

2.2.1 Availability

Infrastructure as a Service should be available to the end-user without any interruptions. This can be provided by avoiding a single point of failure in our system.

Note: The development environment is a limitation to go ahead with this design and the general practice of having a single controller and network node was adopted. Although these nodes might prove to be a single point of failure for the OpenStack services, regular backup of the OpenStack database and the image repository would help recover from any failures.

2.2.2 Reliability and Serviceability (RAS)

The system must be robust and a crash/malfunction on one service should not affect other services. If any virtual instance on a compute node crashes due to a malfunction of some process running on it, the impact of this should be confined to that virtual instance alone. Similarly, software or hardware failures of any of the compute nodes would not hinder the operation or deteriorate the performance of virtual instances running on other compute nodes on the same physical network.

2.2.3 Ease of installation

Installing OpenStack using manual procedure requires the execution of multiple commands and hence is difficult and prone to errors. The OpenStack on a Stick appliance must automate this process and simplifies the installation by running just a single command as an executable.

2.3 List of Tasks

This section lists the tasks performed to achieve the objective of the project which includes understanding OpenStack, manual installation and configuration, compiling the manual steps into scripts for automated installation.

- Understanding basics and installation process of OpenStack
- Manually install and configure OpenStack in a Single server setup:
 - Installing KVM/QEMU related packages on the Host CentOS 7.
 - Creating VMs using KVM and provide the config parameters using kickstart file.
 - Configure network and internet access across VMs.
 - Installation of OpenStack (Newton) across VMs.
 - Configure the OpenStack services.
- Manually install and configure OpenStack in a Multi-server setup:
 - Install CentOS 7 on all the physical machines.
 - Provide Layer-2 connectivity between the machines using the switch.
 - Configure network and internet access across physical machines.
 - Enable hardware virtualization on all the machines.
 - Install KVM/QEMU related packages on the compute Node.
 - Install OpenStack Services across different nodes based on the user requirement.
 - Configure OpenStack service across multiple nodes
- Automation of OpenStack installation: The installation script will perform the following tasks:
 - Read user input and generate a configuration file.
 - Based on user configuration, check the network connectivity.
 - Clean the system before starting the installation.
 - Check for prerequisites condition for Openstack services.
 - Install Openstack using Packstack based on the configuration file.
 - a. On successful installation, provide the URL to access the Dashboard.
 - b. On failure, exit with an appropriate error message.

Note: Provide the installation script to create the virtual machine for the single Server setup.

- Test whether the script can install OpenStack service correctly by create VMs in the web interface and run some application like iperf on the VMs.
- OpenStack un-installation: provide the script to uninstall the OpenStack on all the nodes.

2.4 Environment Specifics

The basic software and hardware requirements for OpenStack installation mandated by the Newton installation guide are as follows:

1. **Operating System:** CentOS 7 stable version

Following are the reasons for choosing Centos:

- Enterprise level operating system.
- Feature complete.
- No licensing issues associated with its usage.
- Good support from the open-source community.

2. **Hypervisor:** KVM

Following are the reasons for choosing QEMU/KVM as the hypervisor:

- Feature complete – Has a good mix of basic and advanced functionalities that are needed for the OpenStack environment.
- No hassle of license purchase and management, which is critical in a multi-tenant environment.
- Good support and documentation from the Open-source community.

3. **Hardware:** The hardware machine should have hardware virtualization enabled. As per the OpenStack documentation [15], the minimum hardware requirements for each node is as mentioned in the Table 1.

Node	Minimum number of CPU	Minimum RAM requirement	Storage of Disk
Network Node	1	1 GB	5 GB
Computing Node	2	2 GB	10 GB
Controller Node	1	2 GB	5 GB

Table 1: Hardware configuration of the machines

Controller node: The controller service should run on a machine with minimum of 1 CPU, 2GB RAM and 5 GB of storage. The RAM and storage are important for controller because of the number of processes that will be running simultaneously on the controller at any point of time.

Compute node: The compute service should run on a machine with minimum of 2 CPUs, 2GB RAM and 10 GB of storage. This is important as it is the compute node on which the virtual machines are created.

Network node: The network service should run on a machine with minimum of 1 CPU, 1GB RAM and 5 GB of storage. As the network node is only used for communication and not for high end processing, the above mentioned minimum requirements should suffice.

Chapter 3: System Environment

This chapter describes the hardware-software setup that was used in the NCSU Network laboratory during implementation of this project.

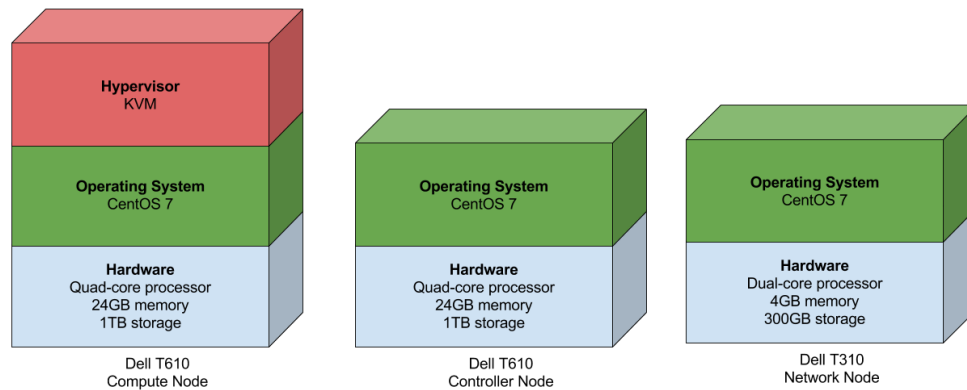


Figure 2: Netlabs Hardware Specification

3.1 Hardware and Software used for Single Server implementation

- Host machine – Dell Power-Edge T610 with Intel Virtualization technology (supports nested virtualization)
- Host OS – Centos 7
- Cisco SG 300 20-port Gigabit managed switch
- TP-LINK TL-WN722N Wireless USB adapter
- Hypervisor – KVM

3.2 Hardware and Software used for Multi Server implementation

- Host machines – 2 Dell PowerEdge T310 and T610 with Intel Virtualization technology
- Host OS – Centos 7
- TP-LINK TL-WN722N Wireless USB adapter
- Cisco SG 300 20-port Gigabit managed switch

3.3 Environment Specifics

- 1) Operating System: CentOS 7
- 2) Hardware configuration details:
 - Dell T310: Dual-core processor, 4GB memory, and 300GB storage
 - Dell T610: Quad-core processor, 24GB memory, and 1TB storage

Chapter 4: Design, Implementation and Results

The chapter covers the design approaches that were considered and chosen to be implemented based on their advantages and disadvantages.

4.1 Design and Architecture

OpenStack offers flexibility to the user to distribute services and components across multiple nodes based on the available resources and user requirements. Using the OpenStack on a Stick appliance, the user will be able to customize configuration and installs the services across different nodes.

According to the OpenStack Architecture Design Guide [14], there are total five basic services Compute, Controller, Network, Block and Object Storage. There could be multiple combinations in which an administrator can install these services on a single Server or multiple servers. However, as this project did not have a lot of data to store, the Block and Object Storage nodes were installed on the same physical server as the Controller Node. This results in three basic services Controller, Network and Compute.

The basic services can be installed either in a Two-node (controller, network on one node and compute on a separate node) or a Three-node architecture. Three-node architecture was selected for implementing this appliance due to the following reasons:

- The architecture becomes more scalable, reliable and available as the services are physically separated.
- A significant performance enhancement is possible as decoupling of services on different nodes reduces the workload on physical hardware.
- Dedicated High-end Network Service offerings (like a firewall etc.) can be implemented using Neutron. Nowadays, these service offerings are no longer optional but it is mandatory.
- It is always good to install the network service on a separate node as it provides better isolation and security.

As per above the evaluation, the three-node architecture diagram is as shown below in Figure 3.

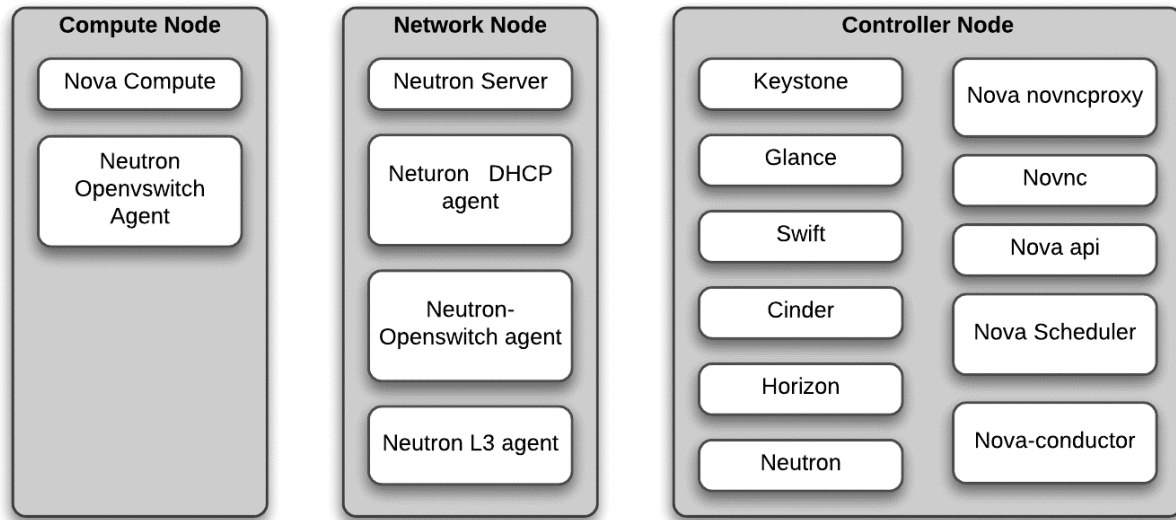


Figure 3: OpenStack Conceptual Architecture^[15]

4.1.1 The three-node architecture

According to the installation guide for OpenStack Newton, services were installed on the nodes as depicted in Figure 3.

- **Controller:** It handles most of the services in OpenStack architecture. Keystone, Glance, Nova API, Neutron API are all installed on the controller node. Supporting services like database and messaging are also installed on the controller node. If object storage is not installed and configured, then the Glance service will use local storage to store the images used to launch virtual machines.
- **Network:** This node handles all incoming and outgoing traffic between internet, OpenStack components and virtual instances across physical compute nodes. Thus, providing a level of isolation for the environment. This node also has the neutron DHCP agent installed on it. This DHCP agent renders public IPs to the spun-up instances. The ML2 plugin allows OpenStack networking to use various complex layer 2 networking technologies.
- **Compute:** This node handles provisioning of VMs on demand. Installation of block/object storage was skipped to keep the architecture simple. Direct Attached Storage (DAS) available on the compute node has been used instead.

4.1.2 Network Design

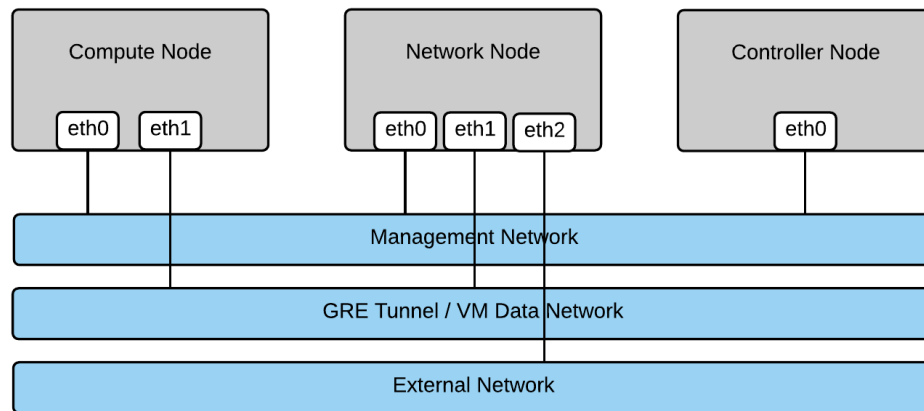


Figure 4: Overall network setup between the nodes

The cloud architecture has three different networks for different traffic purposes:

- Management Network:** This interconnects the three (or more) primary nodes - Network, Controller and Compute nodes over a common subnet 192.168.0.0/24. The subnet can be configured using a configuration file. This network is dedicated to carry informational messages that are involved during the start and stop of OpenStack services, spawning a new virtual instance, authentication, modifying the existing hardware configuration of any instance, etc. In this design, controller, network and compute nodes are assigned with IPs 192.168.0.42/24, 192.168.0.22/24 and 192.168.0.32/24 respectively. These default configurations can be changed by modifying the configuration file as specified.
- GRE Tunnel / VM Data Network:** A Generic Routing Encapsulation (GRE) based tunnel is set between the network and compute nodes. Each compute node that gets added to the cluster would need to establish an independent GRE tunnel with the network node. This is a close private network that is assigned an IP from the network 10.0.1.0/24. The primary objective of this tunnel is to carry network traffic that originates from or is destined to virtual instances hosted by compute node. For example, it is used to carry the inter-VM traffic between compute nodes and traffic between virtual instance and the network node that will further be routed towards the internet using the external network. The cardinal reason for using GRE tunneling is to provide multi-tenant traffic isolation.
- External Network:** It is configured on the network node and is primarily used for connectivity of VMs to the internet. A public IP is assigned to the Network Node upon insertion of the WIFI Modem/Adapter to the machine. NAT and networking configuration was done on this physical server as every node needs access the internet.

The below sections provide in-depth detail of the networking that was configured on each physical server.

- 1) **Compute Node:** An OpenVSwitch runs on the compute node. A default Linux Bridge gets created during installation with default configuration like virbr0 etc. This Linux bridge is attached with one of the network interfaces like eth0, eth1 etc. When nova-compute starts spawning a new VM, it first creates a Linux Tap or port (vport1 or vport2) and then attaches it with the Linux Bridge. And on the other end, this Linux Tap port will be connected to one of the interfaces of the VM.

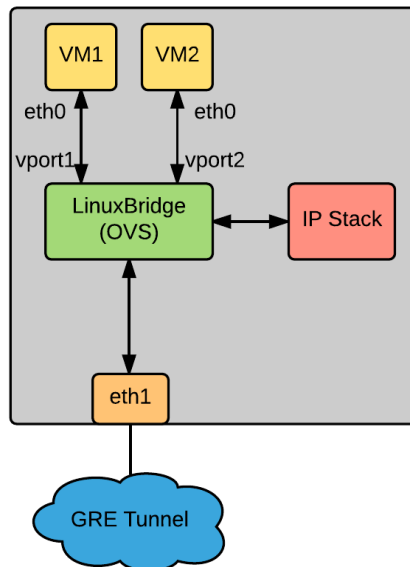


Figure 5: Internal Networking of Compute node

- 2) **Network Node:** A virtual router runs on the network node. When a VM from one subnet, wants to communicate to another VM running in a different subnet, this virtual device provides the required routing rules.

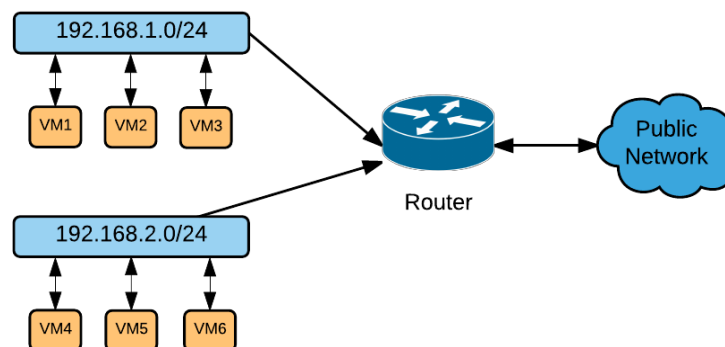


Figure 6: Networking between VMs (via) Network node

Note: Whenever user creates a network, he/she should attach it to the interface of the router as well, to update an entry in the routing table.

4.2 Implementation

The architecture explained in section 4.1.1 has been implemented in two flavors:

- Single Server implementation
- Multi Server implementation

4.2.1 Single Server implementation

A single physical machine is used to host the entire cloud stack where three primary nodes – network, compute and controller nodes are installed or running as separate VMs. A Hypervisor is required to run VMs on the physical machine. OpenStack supports a lot of hypervisors like XEN, KVM etc. For the implementation of OpenStack in this appliance, QEMU/KVM was selected as the hypervisor to create and run these VMs.

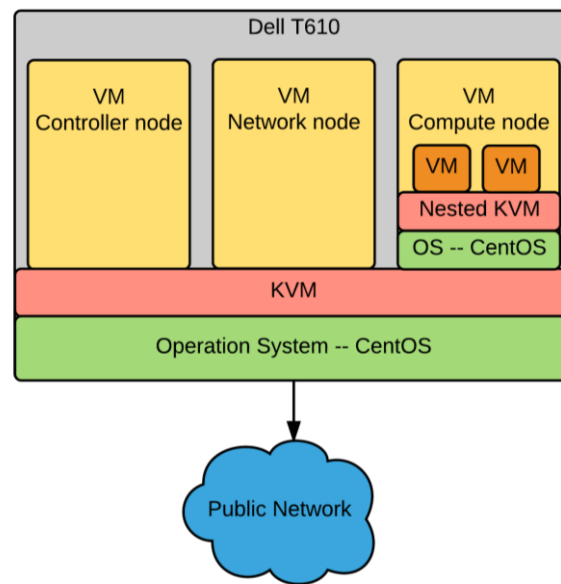


Figure 7: Single Server setup at NetLabs

Advantages:

- There would be faster communication between the services as compared to the Multi Server Implementation because all the VMs are running on the same physical server and the communication is internal to the server.
- It provides ability to capture periodic “snapshots” of the entire cloud configuration throughout the installation process.
- Easier to roll back to the previous configuration setting using the snapshots that were created during the OpenStack installation.
- During installation, in case of a failure of any particular node, the effects will be constrained to that particular VM. Following this, the defunct VM can be shut down and a new virtual instance can immediately be spun up.

Disadvantages:

- The major disadvantage of the single Server installation is performance as compared to multi server because each node is running as a VM instance. Now when a user launches instances on compute node, the instances will run as Nested VMs. So, there will be multiple context switches (2) for each resource request by a running instance where as in multi-server, there will be only one context switch.
- A single server installation of OpenStack will provide limited scalability when compared to multi-server installation due to hardware constraints.
- The services in a single-server OpenStack installation are spawned on VMs in the same server and hence the isolation between these services is not as good as that provided in a multi-server setup.

4.2.2 Multi Server implementation

In our multi server implementation, three physical computers act as the network, compute and controller nodes. These three blades are connected by a switch management network. This kind of implementation is more widely used in real life.

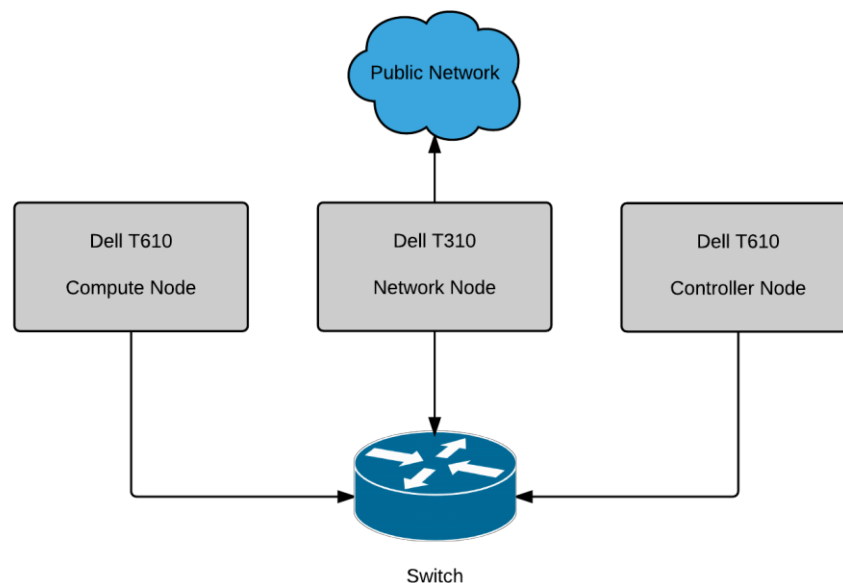


Figure 8: Multi Server setup at NetLabs

Advantages:

- As the compute node is installed on a different physical machine, the VMs created on top of the compute node are not "nested". This results in a better performance (of an application running on the VM) as there are fewer number of context switches as compare to single server implementation.
- Better scalability can be achieved in a Multi-Server setup as compared to Single Server.
- Better Isolation and security is provided in Multi-Server setup as services are running on separate physical server.

Disadvantages:

- Installation and maintenance cost would be more as compare to single server because of increase in number of physical machines.

4.2.3 Scripts for installing OpenStack

The below flowchart depicts the entire logic of how the scripts will be executed.

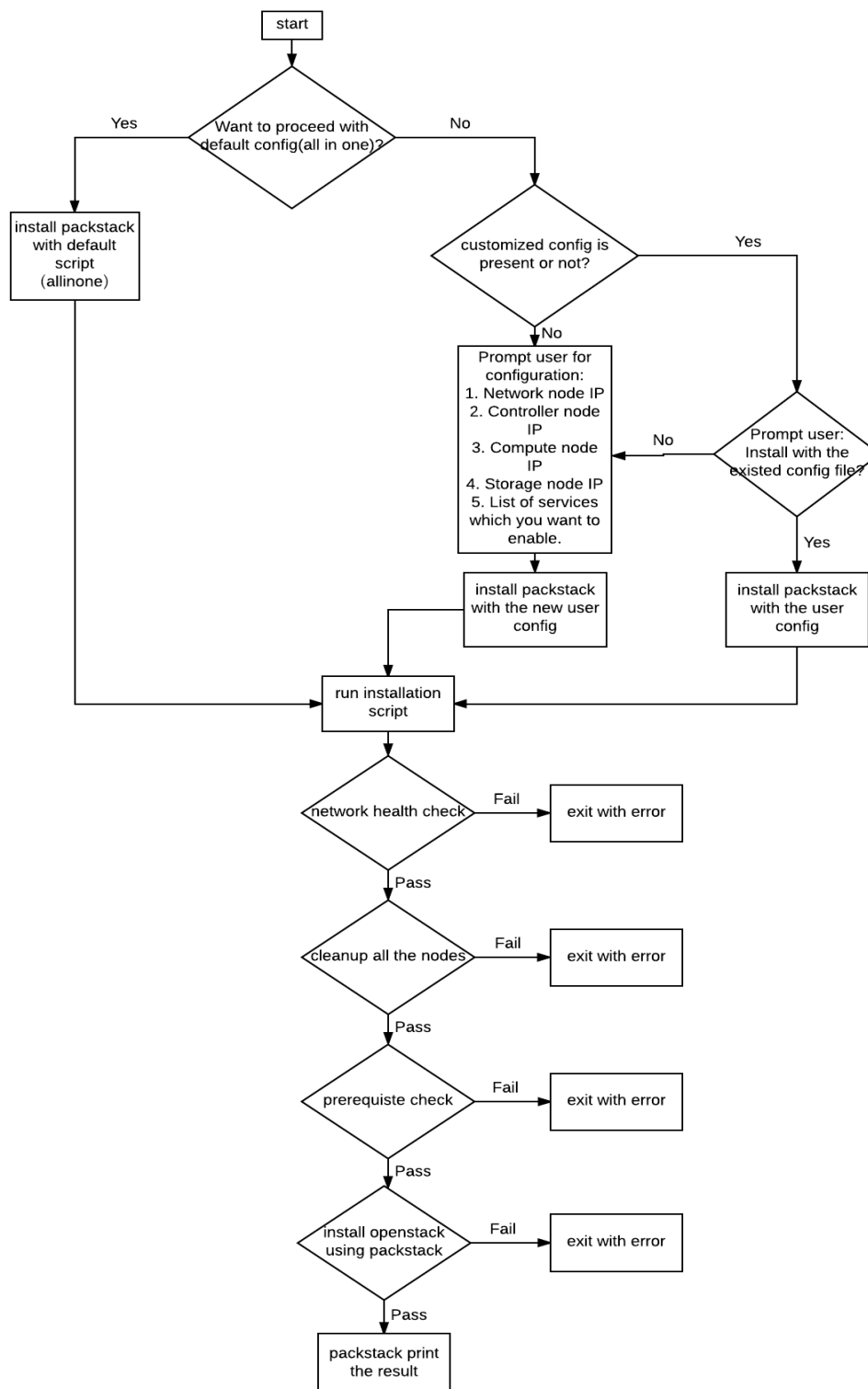


Figure 9: Flow diagram of installation scripts

The flow diagram in Figure 9 has been executed in multiple steps using scripts as mentioned below:

1) User Configuration Step (user_config.sh): When user executes the installation script, it will prompt the user for configuration input.

```

"Do you want to proceed with default configuration? [y/n]"
→ If user input is "no" then first check with the customized config file is present or not.

    → If customize config file is present then prompt the user.
        "Do you really want to proceed with config file (PATH) ?"
        → If yes then
            proceed the installation with the customized file.
        → else
            prompt the user for input (configuration)
    → else
        script should prompt for user input for configuration
        For ex:
        • Please enter the controller node IP address.
        • Do you want to enable cinder service? [y/n].
    → else
        install the openstack with default configuration.

```

Note: Installation with default configuration implies all modules will get installed on single operating system using Packstack with "all-in-one" option.

2) Network Health Checkup Step(network_health_checkup.sh): This script will verify that network health is proper or not. As part of this, it will perform a set of tasks such as:

- a. Network Service should be up and running on all the nodes.
- b. Network connectivity should be up and running between all nodes
- c. All the nodes should be able to access internet

Note: If nodes are not able to access internet, then the script will configure network configuration (NAT) using iptables so that they can access the internet.

3) Cleanup Step (clean_up.sh): This step is executed after the "Network Health checkup". This script ensures that the system is cleaned before running the Openstack installation. This step is very important because if the system is not cleaned then it may result in failure during installation.

4) Prerequisite Step (prerequisites.sh): Ideally, before starting any installation, the script should

verify that all the prerequisite condition mentioned in the OpenStack documentation is verified. For ex: rabbitmq-server (messaging broker): it requires port number 5672 for inter node communication. This step will catch the missing mandatory requirements for the OpenStack installation at an earlier stage.

5) Installation Step (install_openstack.sh): This step will install the OpenStack using Packstack. Packstack will take the configuration file as input. So this step will make sure that it is using the correct user configuration file.

At the end of this step, it will provide the information to the user like:

- On successful installation: Dashboard URL path and other configuration file paths.
- On failure: It will print the error message on the console and provide the path to the log file for the detailed information about the error.

Note: For more information on how to install OpenStack, refer [Appendix D](#).

4.3 Challenges

While setting up OpenStack environment and executing test cases several challenges were faced. The significant challenges and learnings are documented in this section.

- Packstack installation requires that every node has Internet access. However, the hardware provided in lab had only one wifi adapter. Therefore, for successful internet connection NAT rules were configured at the node where wifi adapter was connected using iptables. To make the iptables rules persistent, the rules were saved after insertion in a file so that a restart does not affect the network connectivity.
- During the installation, a failure occurred when the installer was trying to install/start rabbitmq-server via the puppet module amqp.pp because the port 5672 required by the rabbitmq-server was not free. This was one of the prerequisites condition mentioned in the installation guide of rabbitmq server. Similarly, it could be possible that some prerequisites for OpenStack services are not satisfied in an environment. To resolve this, a script that checks the necessary prerequisites for all the modules was created. This verifies that system is meeting these conditions before installation. If the environment is missing something then it will exit with an error message. This also saves a lot of time by catching bugs at the earlier stage.
- Although Packstack provides the administrator an easy way to install the OpenStack, it is very hard to uninstall OpenStack completely and correctly. RDO (RedHat Packet Manager Distribution of OpenStack [16]) provides a script to uninstall Packstack modules but this script does not uninstall all the Openstack related files/rpms. The script provided in the GitHub repository has a wrapper script on top of the Packstack un-installation script, to clean up the system fully.

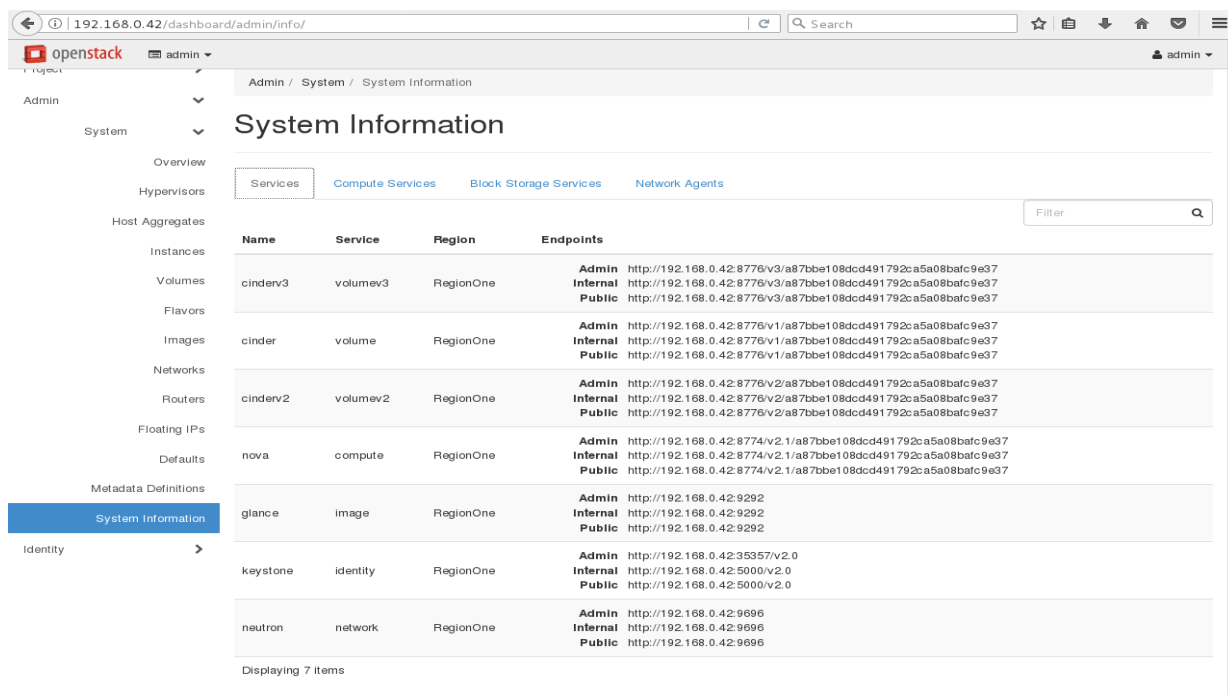
- In the compute node, there was an error while trying to spawn a VM instance because hardware virtualization was not enabled in the BIOS setting. To avoid this failure, the prerequisite script to verifies that hardware virtualization on the compute node is enabled before the installation. If it is not enabled then the script will exit with an error message.

4.4 Results

According to the list of tasks, the first step was to manually create the OpenStack Environment. This was followed by automation and testing of OpenStack single server and multi-server environments with basic service installation and configuration. The installation of OpenStack Newton environment has been automated via bash scripts. Installation of OpenStack is done with minimum user interaction both in single server and multi-server environment.

After the automated installation of OpenStack environment, the dashboard could be accessed using the IP address of the controller node. After logging in as admin and navigating to Admin -> System -> System Information, the following information is obtained:

1. List of all the installed services



Name	Service	Region	Endpoints
cinderv3	volumev3	RegionOne	Admin http://192.168.0.42:8776/v3/a87bbe108dcd491792ca5a08bafc9e37
			Internal http://192.168.0.42:8776/v3/a87bbe108dcd491792ca5a08bafc9e37
			Public http://192.168.0.42:8776/v3/a87bbe108dcd491792ca5a08bafc9e37
cinderv2	volumev2	RegionOne	Admin http://192.168.0.42:8776/v2/a87bbe108dcd491792ca5a08bafc9e37
			Internal http://192.168.0.42:8776/v2/a87bbe108dcd491792ca5a08bafc9e37
			Public http://192.168.0.42:8776/v2/a87bbe108dcd491792ca5a08bafc9e37
nova	compute	RegionOne	Admin http://192.168.0.42:8774/v2.1/a87bbe108dcd491792ca5a08bafc9e37
			Internal http://192.168.0.42:8774/v2.1/a87bbe108dcd491792ca5a08bafc9e37
			Public http://192.168.0.42:8774/v2.1/a87bbe108dcd491792ca5a08bafc9e37
glance	image	RegionOne	Admin http://192.168.0.42:9292
			Internal http://192.168.0.42:9292
			Public http://192.168.0.42:9292
keystone	identity	RegionOne	Admin http://192.168.0.42:35357/v2.0
			Internal http://192.168.0.42:5000/v2.0
			Public http://192.168.0.42:5000/v2.0
neutron	network	RegionOne	Admin http://192.168.0.42:9696
			Internal http://192.168.0.42:9696
			Public http://192.168.0.42:9696

Displaying 7 Items

Figure 10: Dashboard View of Installed OpenStack services.

To view other list of installed services, refer [Appendix B](#).

After listing down the status of all the services, the next step is to launch a new VM or instance. However, an “Image” is needed to spawn the VM. The user can upload the image as per the requirement and customize it using dashboard and then while launching instance, user can choose that image from the list of images.

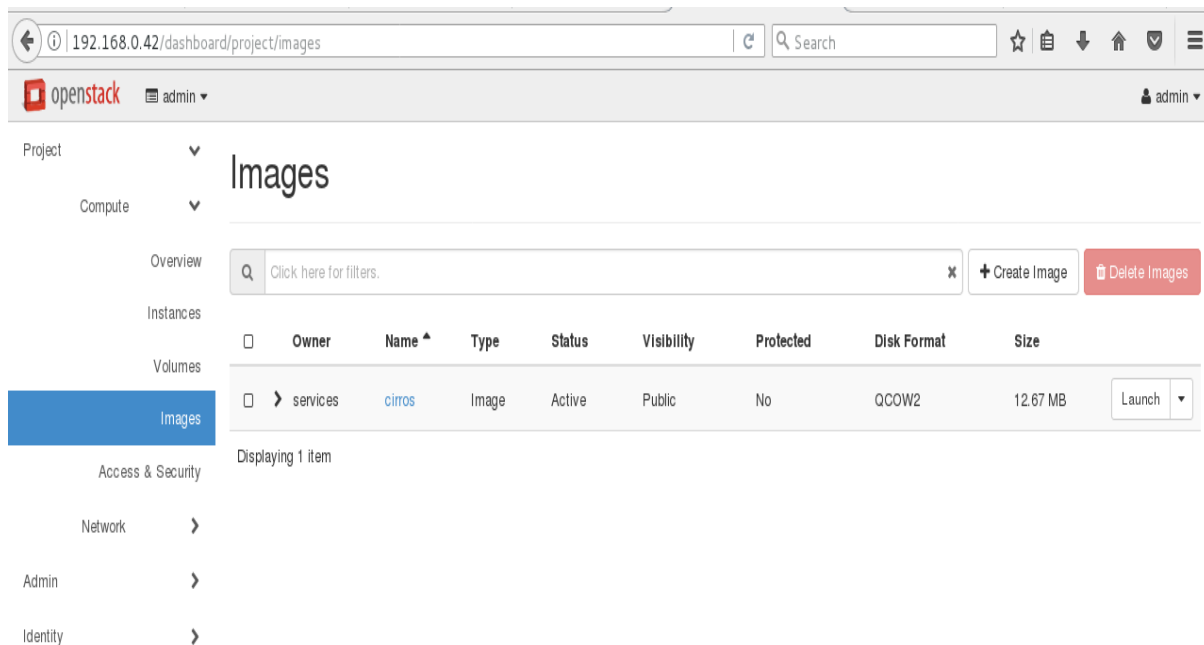


Figure 11: Dashboard View of OpenStack available Images.

To launch a VM, go to Project -> Compute -> Instances and click on the “Launch instance” button.

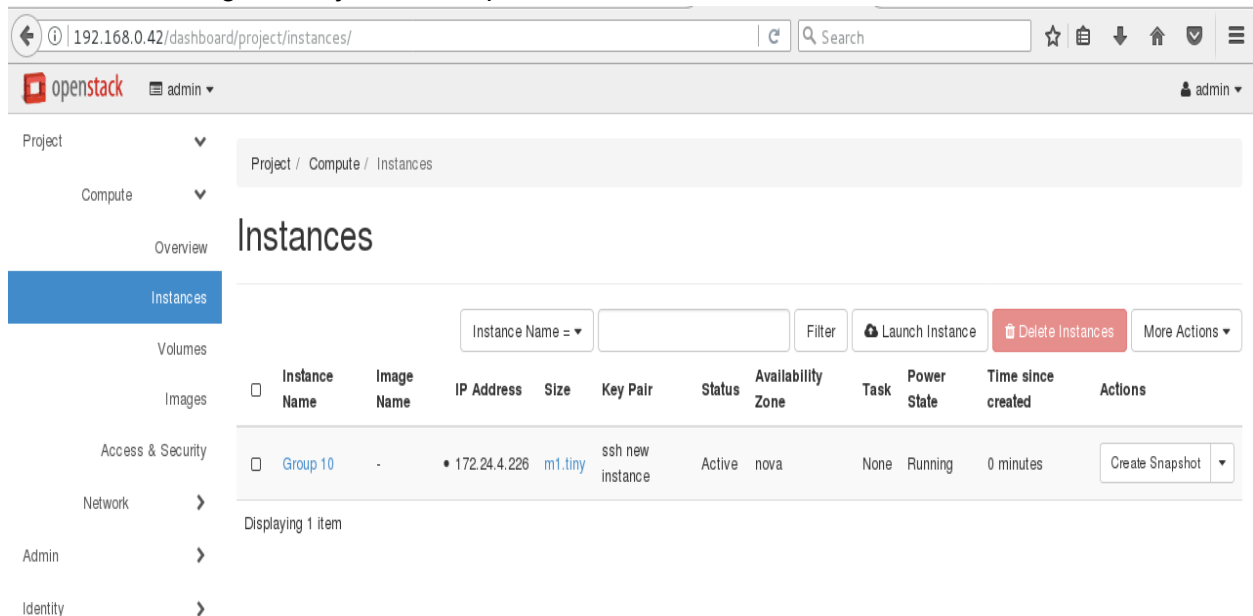


Figure 12: Dashboard View of reserved OpenStack VM Instances.

After launching the VMs, the overview status of the compute node can be viewed by navigating to **Project -> Compute -> Overview**

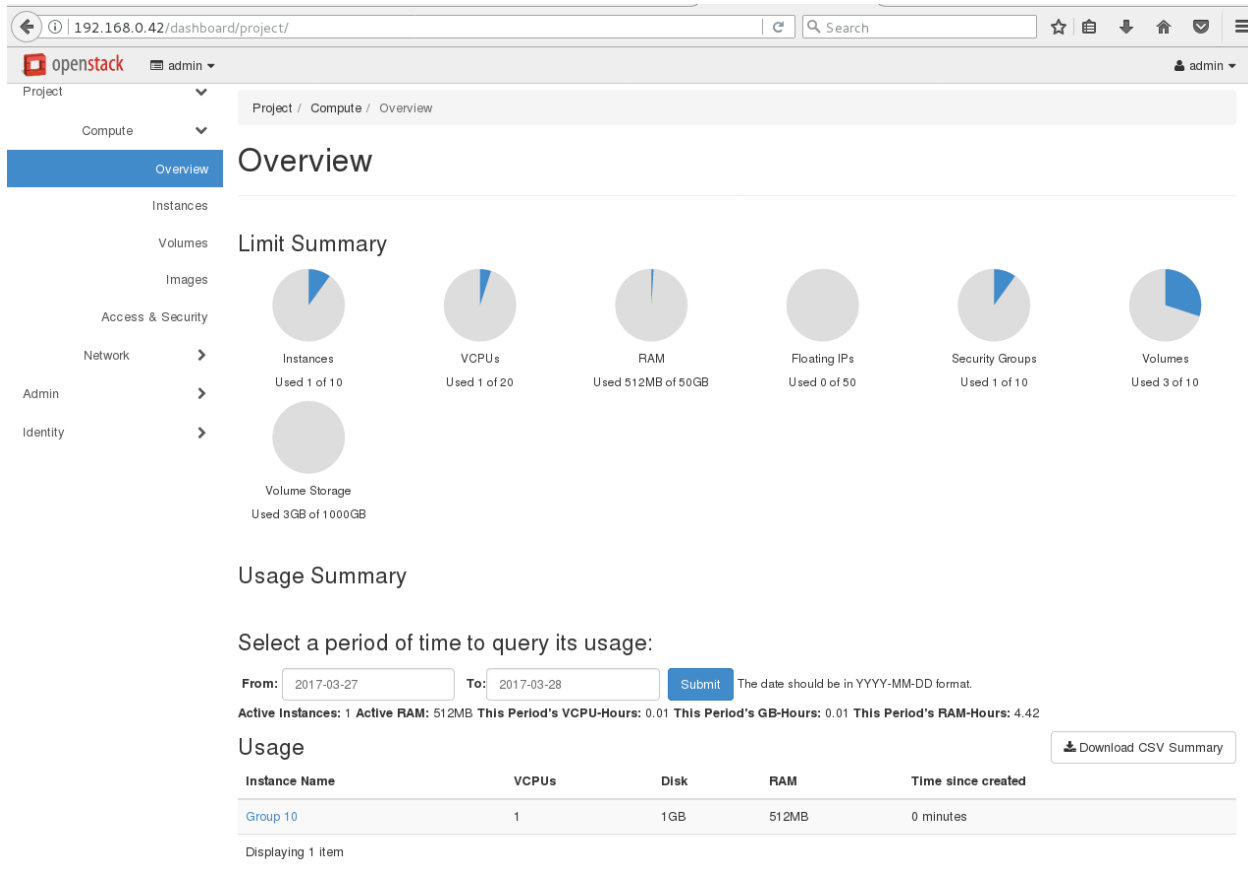


Figure 13: Dashboard Overview of Compute Node.

To check the “list of volumes” attached with a particular VM, the administrator or user can navigate to Project -> Compute -> Volumes.

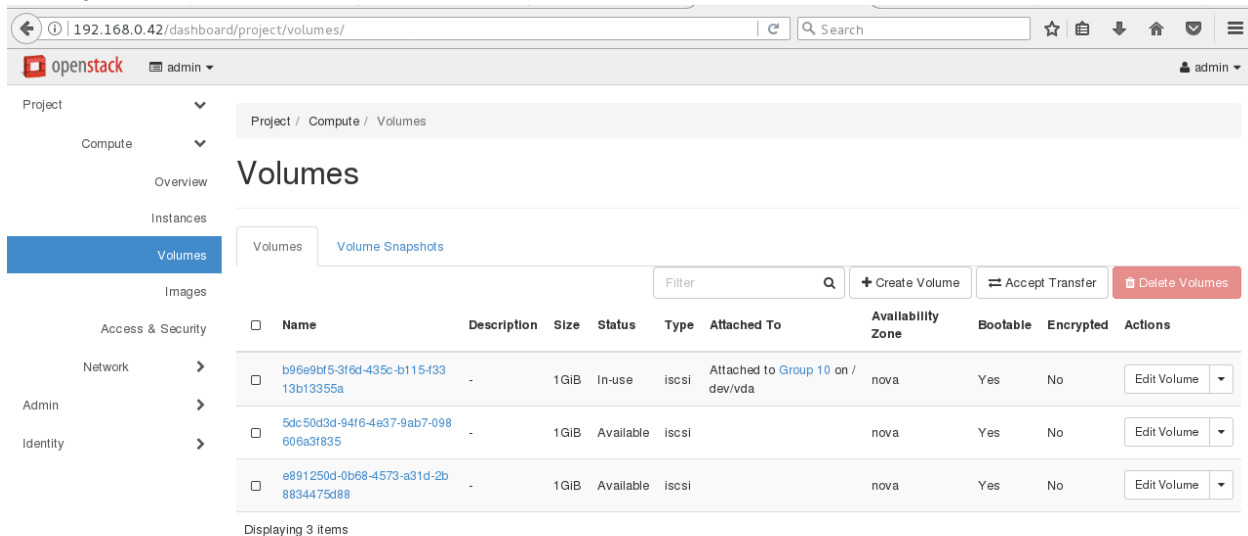


Figure 14: Dashboard view of available Volumes attached to a VM.

Chapter 5: Verification and Validation

This section deals with the verification and validation for all the functional and non-functional requirements along with individual test cases as per the requirement of the project.

5.1 Objective and Tasks

5.1.1 Objective

Verification - To ensure that the OpenStack installation scripts are being built according to the requirements and design specifications. In other words, to ensure that scripts meet their specified requirements.

Validation - To ensure that the installation scripts meets the user's needs, and that the specifications were correct in the first place. In other words, to demonstrate that the scripts fulfill its intended use when placed in its intended environment.

The objective of this test plan is to ensure that final deliverable correctly installs the Newton version of OpenStack.

5.1.2 Tasks

- Unit Testing
- System and Integration Testing
- Stress and Performance Testing
- Security Testing

5.2 Scope

The scope of this test plan only encompasses the installation of OpenStack Newton environment. It covers testing of individual components and, interaction between different components.

5.3 Testing strategy

5.3.1 Unit Testing

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. The unit testing is done to verify that each of the OpenStack components has been successfully installed and behaves as expected.

Tests & Results:

Table 2: Unit Testing Tests and Results

FR - Functional Requirement

Test Case #	Test Case Description and Verification	Result
1)	FR - 2.1.1 Achieve automated installation of Openstack: a. Check if the nodes have the desired network connectivity. b. Check if the system has a clean environment before installation. c. Check if the prerequisites for installation of Openstack are met.	Pass

Expected Result:

Appropriate result should be displayed when Openstack installation script is executed.

- a. All the nodes should have network connectivity
- b. If cloud environment is already setup, it should be cleaned up before installation
- c. Before installing any module, if pre-requisite module is not present, user should be prompted for any action.

In case failure, the script must alert the system administrator to take necessary action.

Verification Steps:

1. Login to the node
2. Execute install_openstack.sh script

Actual Result:

a)

```
[root@bn17-238 ~](vmhost1)# date
Tue Apr 18 21:25:03 EDT 2017
[root@bn17-238 ~](vmhost1)# ./install_openstack.sh

Compute Node is reachable
Network Node is unreachable. exiting !!
```

b)

```
[root@bn17-238 ~](vmhost1)# date
Tue Apr 18 20:59:51 EDT 2017
[root@bn17-238 ~](vmhost1)# ./install_openstack.sh

Destroying VM's and undefining them.

Removing openstack package which are already present.

Removing cinder volumes.

Removing config file.
```

c)

```
[root@bn17-238 ~](vmhost1)# date
Tue Apr 18 20:56:53 EDT 2017
[root@bn17-238 ~](vmhost1)# ./install_openstack.sh

KVM kernel Module is missing.

Do you want me to insert the module ?.[y/n]
y

Installing ipset RPM.

Port 9696 is not free. It is a prerequisite for Neutron Service.

Do you want me to free the port ?.[y/n]
n
[root@bn17-238 ~](vmhost1)# date
Tue Apr 18 20:57:15 EDT 2017
[root@bn17-238 ~](vmhost1)# █
```

Figure 15: Verification result of Functional Requirement 2.1.1

2)	FR - 2.1.2 Users should be able to customize the configuration. a. To choose the default configuration. b. To provide its own customized configuration file. c. To provide the runtime configuration inputs.	Pass
----	--	-------------

Expected Result:

The script should prompt the user to choose default configuration or not, provide its own configuration file and allow the user to give configuration inputs at runtime. In case failure, the script must alert the system administrator to take necessary action.

Verification Steps:

1. Login to the node
2. Execute install_openstack.sh script

Actual Result: The user could provide configuration as per the prompts

```
[VultureMacBook:script xingyao$ date
Tue Apr 18 20:01:26 EDT 2017
[VultureMacBook:script xingyao$ ./install_openstack.sh

Want to proceed with default config -- allinone?(yes/no) no
Checking if the customized config file is existed...
The customized config file is existed.
Install with the existed file?(yes/no) no
Starting user input configuration...
Please enter the network node IP (x.x.x.x): 192.168.0.22
Please enter the compute node IP (x.x.x.x): 192.168.0.32
Please enter the controller node IP (x.x.x.x): 192.168.0.42
Please enter the storage node IP (x.x.x.x): 192.168.0.42
Do you want to enable NOVA?(y/n)y
Do you want to enable NEUTRON?(y/n)y
Do you want to enable GLANCE?(y/n)y
Do you want to enable HORIZON?(y/n)y
Do you want to enable CINDER?(y/n)y
Do you want to enable SWIFT?(y/n)n
```

Figure 16: Verification result of Functional Requirement 2.1.2

3)	FR - 2.1.3 Achieve automated un-installation of OpenStack The user should be able to uninstall OpenStack using scripts.	Pass
----	---	-------------

Expected Result: On executing the un-installation script, the user should be able to uninstall the Openstack.

Verification Steps:

1. Login to the node
2. Check for any Openstack installed modules

- 3. If any module present, execute `uninstall_openstack.sh` script
- 4. Verify if any Openstack modules are still present

Actual Result: There were no Openstack modules present when `uninstall_openstack.sh` script was executed.

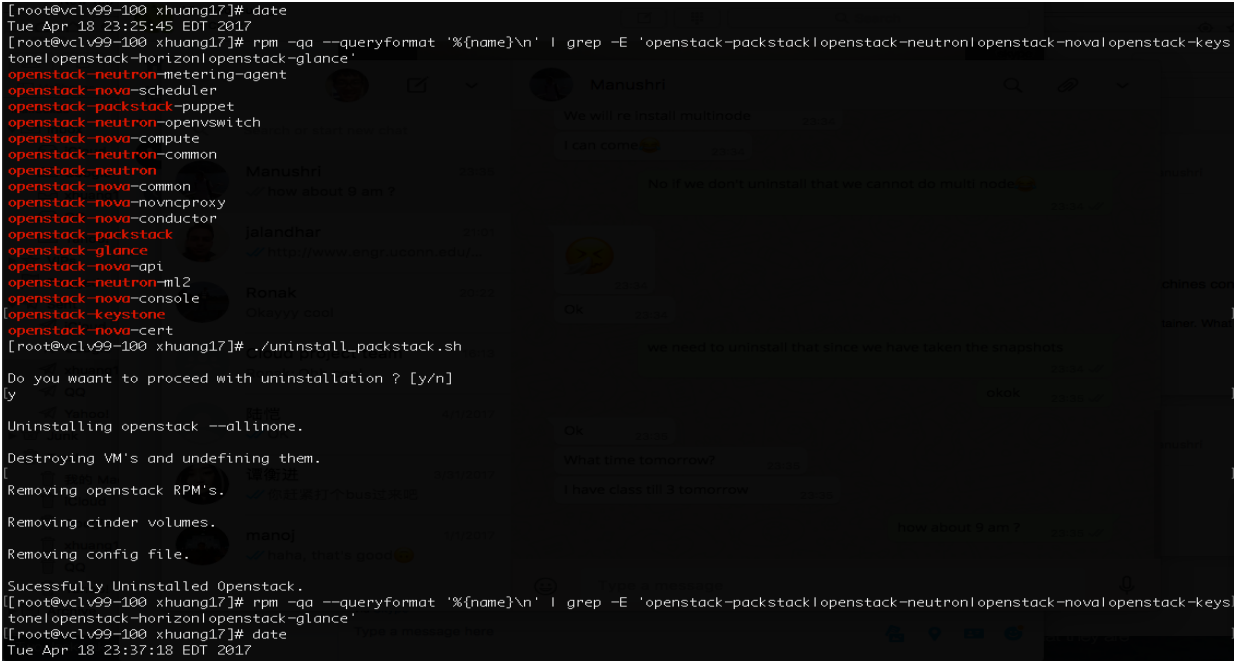


Figure 17: Verification result of Functional Requirement 2.1.3

4)	FR - 2.1.4 Web Interface as a service: The user should be able to access the dashboard with the public IP address and view different sections.	Pass
----	---	------

Expected Result: Dashboard should be accessible

Verification Steps:

- 1. Access the web interface using the IP address
- 2. Log in to the interface using credentials
- 3. Access different tabs of the dashboard

Actual Result:

Dashboard is accessible

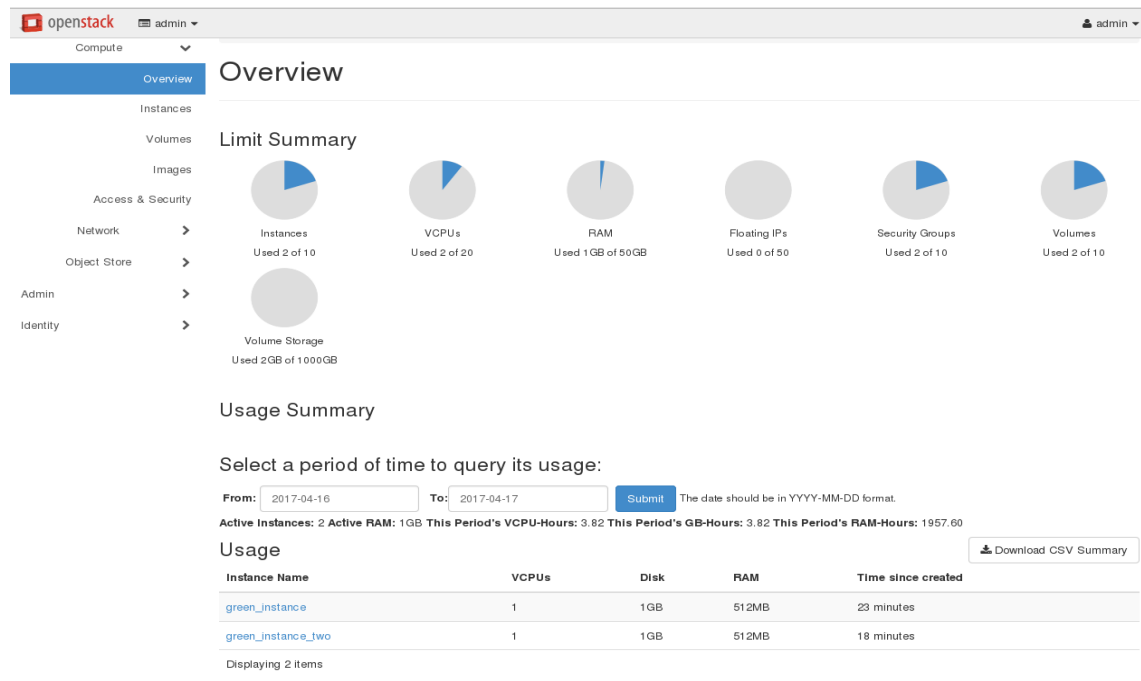


Figure 18: Verification result of Functional Requirement 2.1.4

5)	FR - 2.1.5 Authentication as a Service Only authenticated users should be able to - <ol style="list-style-type: none"> access the dashboard. access each service on the dashboard access the database. 	Pass
----	---	------

Expected Result: Authenticated users should be able to access the dashboard.

Verification Steps:

1. Access the dashboard using the IP address
2. Provide the credentials of the user

Actual Result:

Wrong password was given for the user. Hence, the user was not allowed to log on to the dashboard.

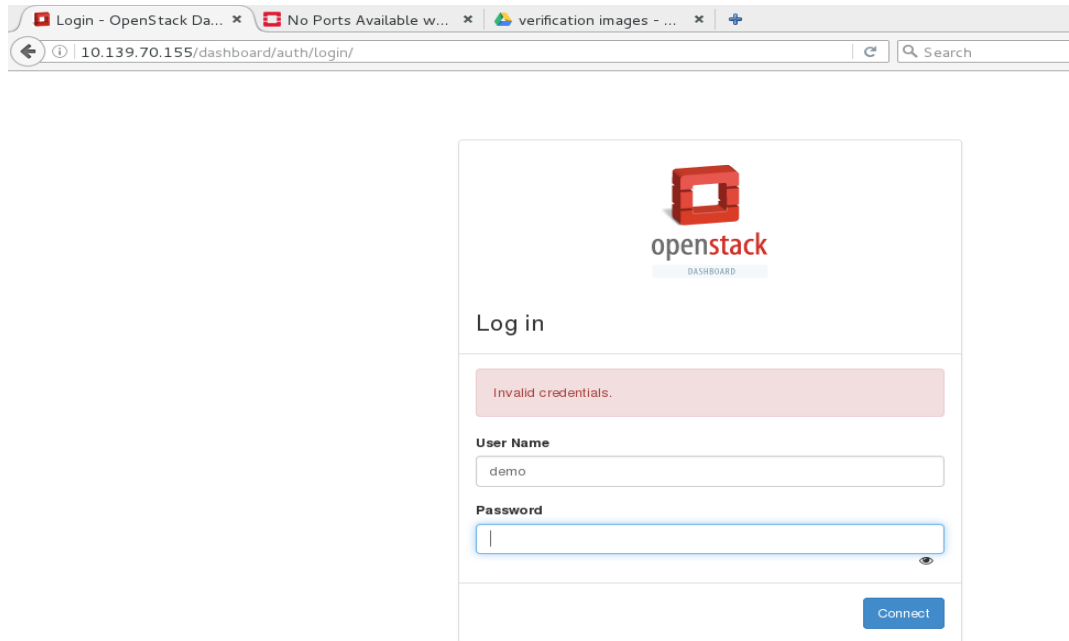


Figure 19: Verification result of Functional Requirement 2.1.5

6)	FR - 2.1.6 Security as a Service: The cloud environment should be secure so that only allowed traffic should be able to enter and exit the network.	Pass
----	--	------

Expected Result: Security groups should be enabled and only that traffic should be allowed.

Verification Steps:

1. Login to the dashboard
2. Go to Compute tab and click on Access and Security
3. Add/remove appropriate rules as per requirement

Actual Result:

User could provide security rules.

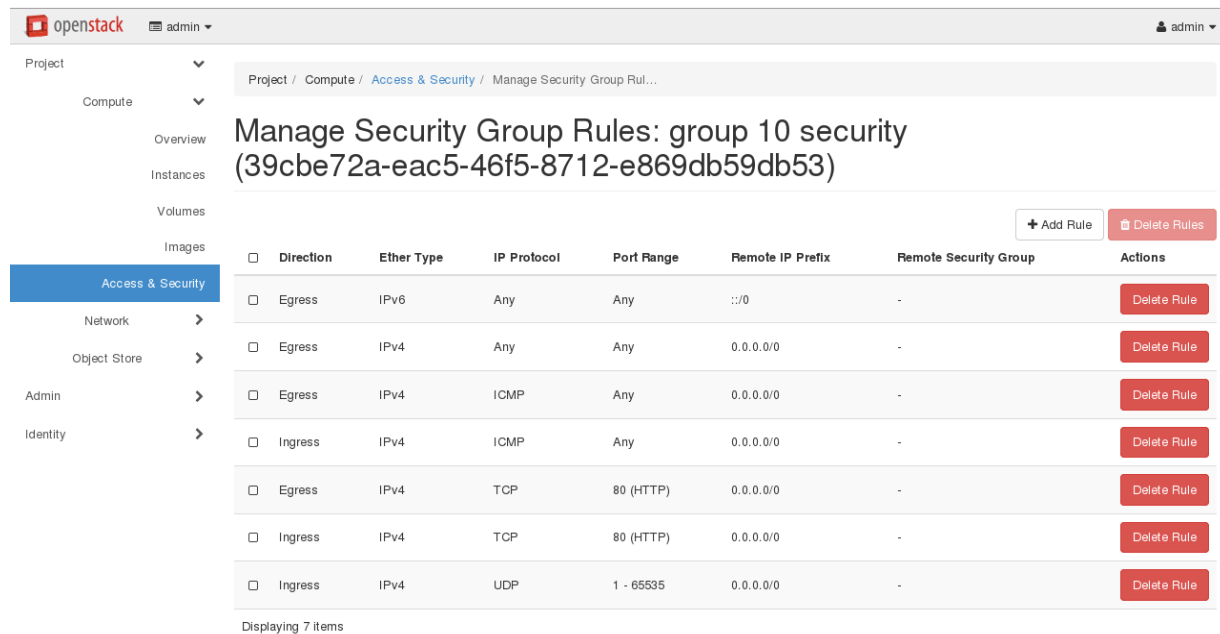


Figure 20: Verification result of Functional Requirement 2.1.6

7)	FR - 2.1.7 Image Management as a Service: After configuring the Glance Service load an image into the glance database and can receive the same back by API calls	Pass
----	---	-------------

Expected Result: Image load and retrieval should be successful

Verification Steps:

1. Login to the dashboard with user credentials
2. Click on the compute section and go to images tab
3. User should be able to add, delete, view any images.

Actual Result:

The user could add, delete and view images.

When comparing to the images section as seen in Figure-11 a new CentOS image is added and can be viewed from the dashboard.

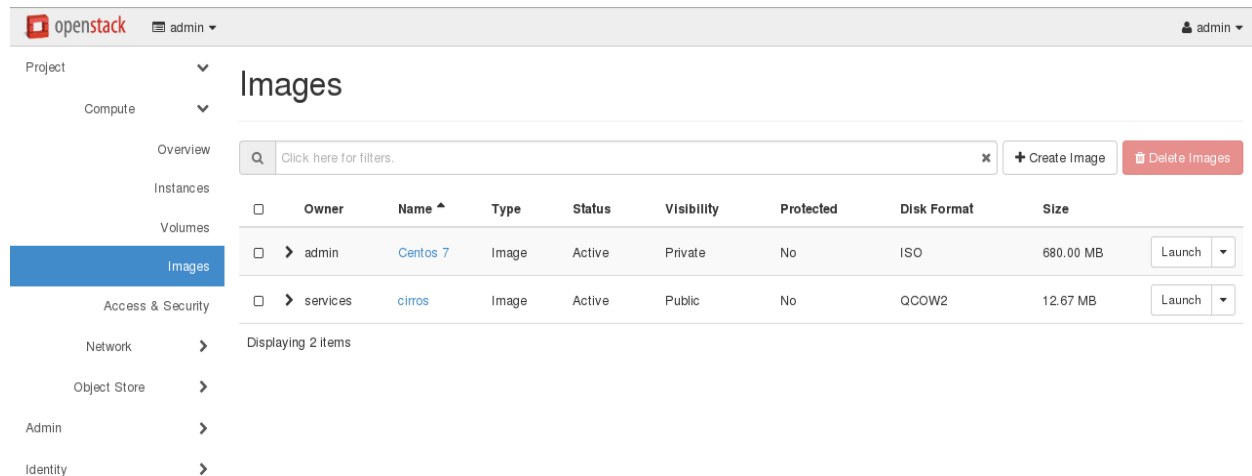


Figure 21: Verification result of Functional Requirement 2.1.7

8)	FR - 2.1.8 Network as a Service The user should be able to <ol style="list-style-type: none"> Create and delete network. Create and delete Routers. Create and delete network interface to the router. 	Pass
----	---	------

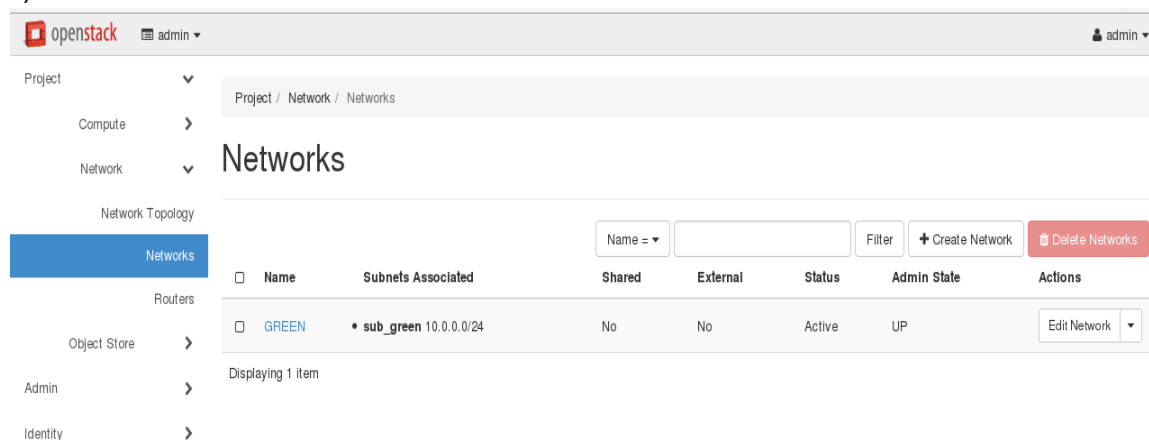
Expected Result: The user can create network via dashboard

Verification Steps:

1. Login to the dashboard
2. Go to Networks tab and click on create network
3. Give relevant details about the network and save
4. Go to router tab and create router and add interface of router to the network

Actual Result:

a) User could create a network



b) User could create routers and add interface to the network.

The screenshot shows the OpenStack dashboard interface for configuring a router. The breadcrumb trail is Project / Network / Routers / group 10 router. The page title is 'group 10 router'. There are tabs for Overview, Interfaces, and Static Routes. The Interfaces tab is active, showing a table of interfaces for the router. The table has columns: Name, Fixed IPs, Status, Type, Admin State, and Actions. There are three interfaces listed: an internal interface (2bcf50f-3b86) with IP 10.0.0.1, an external gateway (90fc8046-6247) with IP 192.168.20.5, and another internal interface (b593a26c-18cb) with IP 192.168.100.1. Each interface has a 'Delete Interface' button. There are also buttons for 'Add Interface' and 'Delete Interfaces' at the top right of the table.

Name	Fixed IPs	Status	Type	Admin State	Actions
(2bcf50f-3b86)	10.0.0.1	Active	Internal Interface	UP	Delete Interface
(90fc8046-6247)	192.168.20.5	Build	External Gateway	UP	Delete Interface
(b593a26c-18cb)	192.168.100.1	Active	Internal Interface	UP	Delete Interface

Figure 22: Verification result of Functional Requirement 2.1.8

9)	FR - 2.1.9 On-demand provisioning of Compute Resources The user should be able to launch, reserve, modify and terminate virtual instances.	Pass
----	--	------

Expected Result: The user should be able to create instances for computation.

Verification Steps:

1. Login to the dashboard
2. Go to Compute tab and click create instances.
3. Provide relevant details and save them.

Actual Result:

The user could create the instances and view/modify them.

The screenshot shows the OpenStack dashboard interface for managing instances. The breadcrumb trail is Project / Compute / Instances. The page title is 'Instances'. There are buttons for 'Launch Instance', 'Delete Instances', and 'More Actions'. Below these buttons is a table of instances. The table has columns: Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. There are three instances listed: 'red_instance' with IP 192.168.100.5, 'green_instance_t wo' with IP 10.0.0.7, and 'green_instance' with IP 10.0.0.5. Each instance has a 'Create Snapshot' button in the Actions column. There is also a search bar and a filter button at the top of the table.

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
red_instance	-	192.168.100.5	m1.tiny	-	Active	nova	None	Running	5 minutes	Create Snapshot
green_instance_t wo	-	10.0.0.7	m1.tiny	-	Active	nova	None	Running	31 minutes	Create Snapshot
green_instance	-	10.0.0.5	m1.tiny	-	Active	nova	None	Running	36 minutes	Create Snapshot

Figure 23: Verification result of Functional Requirement 2.1.9

10	FR - 2.1.10 Scalability: During installation, the scripts should prompt the user to manipulate the number of nodes required	Pass
----	--	------

Expected Result: New compute nodes should be added on executing the scripts.

Verification Steps:

1. Login to the node
2. Execute add_compute_node.sh script

Actual Result: As seen, a new compute node has been created.

```
[root@localhost ~(keystone_admin)]# date
Thu Apr 20 18:25:40 EDT 2017
[root@localhost ~(keystone_admin)]# nova hypervisor-list
+-----+-----+-----+-----+
| ID | Hypervisor | hostname | State | Status |
+-----+-----+-----+-----+
| 2 | compute-host2 | | up | enabled |
| 3 | compute-host1 | | up | enabled |
+-----+-----+-----+-----+
[root@localhost ~(keystone_admin)]# date
Thu Apr 20 18:25:50 EDT 2017
```

Figure 24: Verification result of Functional Requirement 2.1.10

To view the additional results for this test case, refer [Appendix C, Section 5.3.1 Test Case 10](#).

11	FR - 2.1.11 Isolation: Traffic should be isolated between two different tenants.	Pass
----	---	------

Expected Result: Vxlan tunnel should be present with appropriate configuration.

Verification Steps:

1. Login to the node
2. Execute ovs-vsctl command to display the vxlan interface configuration

Actual Result:

As seen, vxlan tunnel is configured between tenants to provide isolation.

```

type: vxlan
Bridge br-tun
  Controller "tcp:127.0.0.1:6633"
    is_connected: true
  fail_mode: secure
  Port br-tun
    Interface br-tun
      type: internal
    Port "vxlan-c0a80034"
      Interface "vxlan-c0a80034"
        type: vxlan
        options: {df_default="true", in_key=flow, local_ip="192.168.0.22", out_key=flow, remote_ip="192.168.0.52"}
    Port "vxlan-c0a80020"
      Interface "vxlan-c0a80020"
        type: vxlan
        options: {df_default="true", in_key=flow, local_ip="192.168.0.22", out_key=flow, remote_ip="192.168.0.32"}
  Port patch-int
    Interface patch-int
      type: patch
      options: {peer=patch-tun}

```

Figure 25: Verification result of Functional Requirement 2.1.11

12	Storage as a Service: The cloud environment should be able to provide storage for the instances created.	Pass
----	---	------

Expected Result: Storage is provided whenever instance is created.

Verification Steps:

1. Login to the dashboard and create instances.
2. Go to Compute tab and click on Volumes
3. Volumes attached to the created instances should be displayed.

Actual Result:

The screenshot shows the OpenStack dashboard interface. The 'Volumes' tab is selected under the 'Compute' section. The table lists two volumes:

Name	Description	Size	Status	Type	Attached To	Availability Zone	Bootable	Encrypted	Actions
e8d71416-1956-40b5-8c56-34d5d147d0eb	-	1GiB	In-use	iscsi	Attached to green_instance_tw on /dev/vda	nova	Yes	No	Edit Volume
2677e517-74e9-49ba-8be6-534f540b5445	-	1GiB	In-use	iscsi	Attached to green_instance on /dev/vda	nova	Yes	No	Edit Volume

At the bottom of the table, it says 'Displaying 2 items'.

Figure 26: Verification result of Storage as a Service

5.3.2 System and Integration Testing

System testing is performed on the entire system in the context of a Functional Requirement Specification. It is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. The system and integration testing verifies whether various OpenStack components can communicate amongst themselves and integrate to form a working system.

Tests & Results:

Table 3: System and Integration Testing Tests and Results

Test Case #	Test Case Description and Verification	Result
1)	Compute and controller nodes can connect to the internet via the network node	Pass

Expected Result: The compute and controller node should be able to ping the internet

Verification Steps:

1. Login to compute and controller nodes
2. Execute command ping 8.8.8.8
3. Ping should be successful

Actual Result: Ping was successful

```
[root@localhost ~]# date
Thu Apr 20 13:14:10 EDT 2017
[root@localhost ~]#
[root@localhost ~]# ping -c 4 192.168.0.22
PING 192.168.0.22 (192.168.0.22) 56(84) bytes of data.
64 bytes from 192.168.0.22: icmp_seq=1 ttl=64 time=0.206 ms
64 bytes from 192.168.0.22: icmp_seq=2 ttl=64 time=0.190 ms
64 bytes from 192.168.0.22: icmp_seq=3 ttl=64 time=0.238 ms
64 bytes from 192.168.0.22: icmp_seq=4 ttl=64 time=0.212 ms

--- 192.168.0.22 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.190/0.211/0.238/0.022 ms
[root@localhost ~]#
[root@localhost ~]# date
Thu Apr 20 13:14:53 EDT 2017
[root@localhost ~]#
[root@localhost ~]# ping -c 4 192.168.0.42
PING 192.168.0.42 (192.168.0.42) 56(84) bytes of data.
64 bytes from 192.168.0.42: icmp_seq=1 ttl=64 time=0.217 ms
64 bytes from 192.168.0.42: icmp_seq=2 ttl=64 time=0.244 ms
64 bytes from 192.168.0.42: icmp_seq=3 ttl=64 time=0.228 ms
64 bytes from 192.168.0.42: icmp_seq=4 ttl=64 time=0.247 ms

--- 192.168.0.42 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.217/0.234/0.247/0.012 ms
[root@localhost ~]#
[root@localhost ~]# date
Thu Apr 20 13:15:34 EDT 2017
[root@localhost ~]# █
```

Figure 27: Verification result of network connectivity between nodes

2)	The instances created in OpenStack setup can ping each other successfully	Pass
----	---	------

Expected Result: Ping should be successful between two instances

Verification Steps:

1. Login to any instance via the dashboard
2. Execute command ping <ip address of the other instance>
3. Ping should be successful

Actual Result: Ping was successful

a) Ping between instances on same subnet

```
Mon Apr 17 22:32:29 UTC 2017
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:84:53:A8
          inet addr:10.0.0.5  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe84:53a8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:96 errors:0 dropped:0 overruns:0 frame:0
          TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5179 (5.0 KiB)  TX bytes:4504 (4.3 KiB)

$
$ ping -c 4 10.0.0.7
PING 10.0.0.7 (10.0.0.7): 56 data bytes
64 bytes from 10.0.0.7: seq=0 ttl=64 time=1.955 ms
64 bytes from 10.0.0.7: seq=1 ttl=64 time=0.572 ms
64 bytes from 10.0.0.7: seq=2 ttl=64 time=0.474 ms
64 bytes from 10.0.0.7: seq=3 ttl=64 time=0.439 ms

--- 10.0.0.7 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.439/0.860/1.955 ms
$ date
Mon Apr 17 22:33:39 UTC 2017
$
```

b) Ping between instances on different subnet

```
$ date
Mon Apr 17 22:58:55 UTC 2017
$ ping -c 4 192.168.100.5
PING 192.168.100.5 (192.168.100.5): 56 data bytes
64 bytes from 192.168.100.5: seq=0 ttl=63 time=1.709 ms
64 bytes from 192.168.100.5: seq=1 ttl=63 time=0.641 ms
64 bytes from 192.168.100.5: seq=2 ttl=63 time=0.672 ms
64 bytes from 192.168.100.5: seq=3 ttl=63 time=0.651 ms

--- 192.168.100.5 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.641/0.918/1.709 ms
$ date
Mon Apr 17 22:59:11 UTC 2017
$
```

Figure 28: Verification result of network connectivity between instances

To view the network topology for this test case, refer [Appendix C, Section 5.3.2 Test Case 1 - 4](#).

3)	The tunnels can route the packets from the compute node to the network node successfully	Pass
----	--	------

Expected Result: Pings should go through. Also, check TCPDUMP on the nodes to verify the packets are sent via tunnel

Verification Steps:

1. Login to compute node
2. Execute command ping <ip address of network node>
3. Start tcpdump on network node
4. Ping should be successful and tcpdump should show packets via tunnel

```
[root@localhost ~]# date
Thu Apr 20 10:51:34 EDT 2017
[root@localhost ~]# ifconfig enp4s0
enp4s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.22 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::ec58:4fff:665d:de4 prefixlen 64 scopeid 0x20<link>
    ether 00:1a:a0:07:a5:0c txqueuelen 1000 (Ethernet)
    RX packets 801477 bytes 91285621 (87.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1017243 bytes 507290386 (483.7 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 17

[root@localhost ~]# tcpdump -i enp4s0 -n -e | grep -i VXLAN
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp4s0, link-type EN10MB (Ethernet), capture size 65535 bytes
10:51:54.972485 a4:ba:db:51:73:63 > 00:1a:a0:07:a5:0c, ethertype IPv4 (0x0800), length 148: 192.168.0.32.56401 > 192.168.0.22.4789: VXLAN, f
lags [I] (0x08), vni 48
10:51:54.972645 00:1a:a0:07:a5:0c > a4:ba:db:51:73:63, ethertype IPv4 (0x0800), length 148: 192.168.0.22.35856 > 192.168.0.32.4789: VXLAN, f
lags [I] (0x08), vni 78
10:51:54.972962 a4:ba:db:51:73:63 > 00:1a:a0:07:a5:0c, ethertype IPv4 (0x0800), length 148: 192.168.0.32.56401 > 192.168.0.22.4789: VXLAN, f
lags [I] (0x08), vni 78
10:51:54.973001 00:1a:a0:07:a5:0c > a4:ba:db:51:73:63, ethertype IPv4 (0x0800), length 148: 192.168.0.22.35856 > 192.168.0.32.4789: VXLAN, f
lags [I] (0x08), vni 48
10:51:55.972629 a4:ba:db:51:73:63 > 00:1a:a0:07:a5:0c, ethertype IPv4 (0x0800), length 148: 192.168.0.32.56401 > 192.168.0.22.4789: VXLAN, f
lags [I] (0x08), vni 48
10:51:55.972721 00:1a:a0:07:a5:0c > a4:ba:db:51:73:63, ethertype IPv4 (0x0800), length 148: 192.168.0.22.35856 > 192.168.0.32.4789: VXLAN, f
lags [I] (0x08), vni 78
10:51:55.973135 a4:ba:db:51:73:63 > 00:1a:a0:07:a5:0c, ethertype IPv4 (0x0800), length 148: 192.168.0.32.56401 > 192.168.0.22.4789: VXLAN, f
lags [I] (0x08), vni 78
10:51:55.973174 00:1a:a0:07:a5:0c > a4:ba:db:51:73:63, ethertype IPv4 (0x0800), length 148: 192.168.0.22.35856 > 192.168.0.32.4789: VXLAN, f
lags [I] (0x08), vni 48
^C41 packets captured
41 packets received by filter
0 packets dropped by kernel

[root@localhost ~]# date
Thu Apr 20 10:51:59 EDT 2017
[root@localhost ~]#
```

Figure 29: Verification result of network connectivity between nodes

4)	Virtual instance connects to the internet via ping and can download packages	Pass
----	--	------

Expected Result: Ping and download of packages should be successful

Verification Steps:

1. Login to any instance via dashboard
2. Execute command ping 8.8.8.8
3. It should be successful
4. Execute “wget” command to download any package and it should be successful

Actual Result:

The instance could ping the internet successfully.

```
$ date
Wed Apr 19 00:49:37 UTC 2017
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:84:53:A8
          inet addr:10.0.0.5  Bcast:10.0.0.255  Mask:255.255.0
          inet6 addr: fe80::f816:3eff:fe84:53a8/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:1380 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2106 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:159649 (155.9 KiB)  TX bytes:200828 (196.1 KiB)

$ ping -c 4 8.8.8.8
PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: seq=0 ttl=42 time=20.223 ms
64 bytes from 8.8.8.8: seq=1 ttl=42 time=22.416 ms
64 bytes from 8.8.8.8: seq=2 ttl=42 time=19.699 ms
64 bytes from 8.8.8.8: seq=3 ttl=42 time=19.739 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 19.699/20.519/22.416 ms
$ date
Wed Apr 19 00:50:01 UTC 2017
$ _
```

The instance could download package from the internet.

```
$ date
Thu Apr 20 20:43:39 UTC 2017
$ ifconfig eth0
eth0      Link encap:Ethernet  HWaddr FA:16:3E:2C:55:F2
          inet addr:10.0.0.5  Bcast:10.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe2c:55f2/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:783 errors:0 dropped:0 overruns:0 frame:0
          TX packets:617 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5286942 (5.0 MiB)  TX bytes:48572 (47.4 KiB)

$ sudo wget http://ftp.gnu.org/gnu/wget/wget-1.5.3.tar.gz
Connecting to ftp.gnu.org (208.118.235.20:80)
wget-1.5.3.tar.gz  100% |*****| 436k  0:00:00 ETA
$
$ ls -lrt /dev/wget-1.5.3.tar.gz
-rw-r--r--  1 root  root  446966 Apr 20 20:43 /dev/wget-1.5.3.tar.gz
$ date
Thu Apr 20 20:44:06 UTC 2017
$ sni
```

Figure 30: Verification result of internet connectivity from an instance

To view the network topology for this test case, refer [Appendix C, Section 5.3.2 Test Case 1-4](#).

5)	Login as a user and load an image that is privately available	Pass
----	---	------

Expected Result: Private image created by a user should not be accessible to other users

Verification Steps:

1. Login to dashboard
2. Create an image and make it private to that user
3. Logout and login via different user
4. This new user should not be able to view the previous image

Actual Result: The new user was not able to view the private image

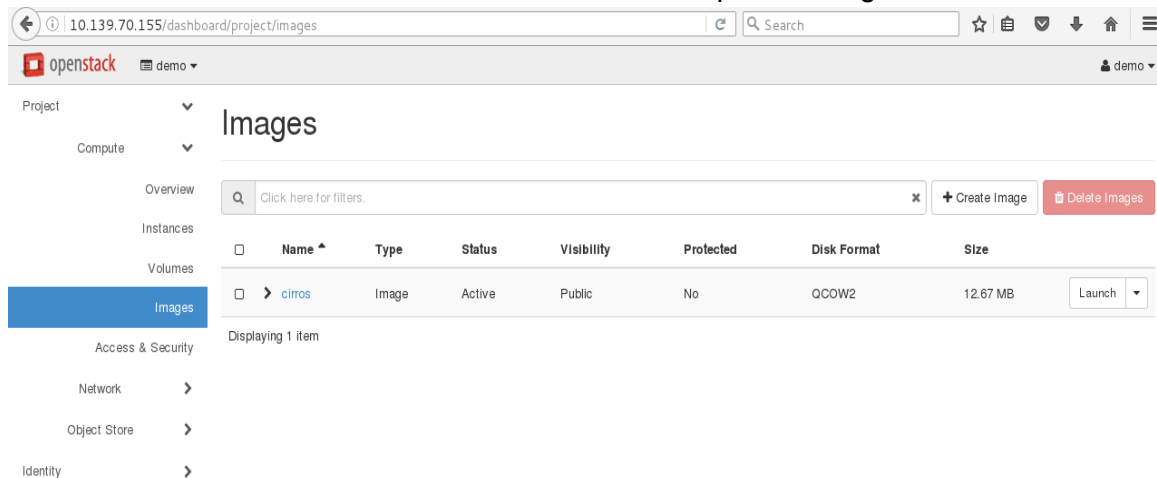


Figure 31: Verification result of user access to private images

For other screenshots related to this test case refer [Appendix C, Section 5.3.2 Test Case 5](#).

5.3.3 Stress and Performance Testing:

This type of testing determines the robustness of software by testing beyond the limits of normal operation. Stress tests commonly put a greater emphasis on robustness, availability, and error handling under a heavy load. Stress and performance testing verified that the script can install a horizontally scalable OpenStack environment and can handle varying load as per system configuration.

Tests and Results:

Table 4: Stress and Performance Testing Tests and Results

Stress Testing:

Test Case #	Test Case Description and Verification	Result
1)	Utilize all available resources on the compute node to launch as many VMs as theoretically possible	Pass

Expected Result: Successful reservations of VMs

Verification Steps:

1. Login to the dashboard
2. Launch 10 m1.tiny instances on T310 dell compute server nodes.
3. Limiting factors were 4 GB RAM, number of virtual CPU instances available to OpenStack scheduler and limited 100 GB storage space on compute nodes

Actual Result: Instances were successfully created and it shows that “Resource Quota is exceeded”.

The screenshot shows the OpenStack dashboard interface. The left sidebar contains navigation links for Project, Compute, Overview, Instances (selected), Volumes, Images, Access & Security, Network, Admin, and Identity. The main content area is titled 'Instances' and shows a table of 10 instances. The table columns are Instance Name, Image Name, IP Address, Size, Key Pair, Status, Availability Zone, Task, Power State, Time since created, and Actions. The instances listed are:

Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
green instance scale-2	-	192.168.123.14	m1.tiny	group 10 ssh key pair	Active	nova	None	Running	0 minutes	Create Snapshot
green instance scale-1	-	192.168.123.13	m1.tiny	group 10 ssh key pair	Active	nova	None	Running	0 minutes	Create Snapshot
green instance new-2	-	192.168.123.12	m1.tiny	-	Active	nova	None	Running	37 minutes	Create Snapshot
green instance new-1	-	192.168.123.10	m1.tiny	-	Active	nova	None	Running	37 minutes	Create Snapshot
red instance new-2	-	10.0.0.11	m1.tiny	-	Active	nova	None	Running	38 minutes	Create Snapshot

Figure 32: Verification result of launching 10 instances

2)	Launching virtual instance once compute capacity was saturated	Pass
----	--	------

Expected Result: Instance reservation should fail

Verification Steps:

- 1. Follow verification steps of above test case
- 2. Launch additional instance

Actual Result: The instance reservation fails throwing the error as shown in the figure below.

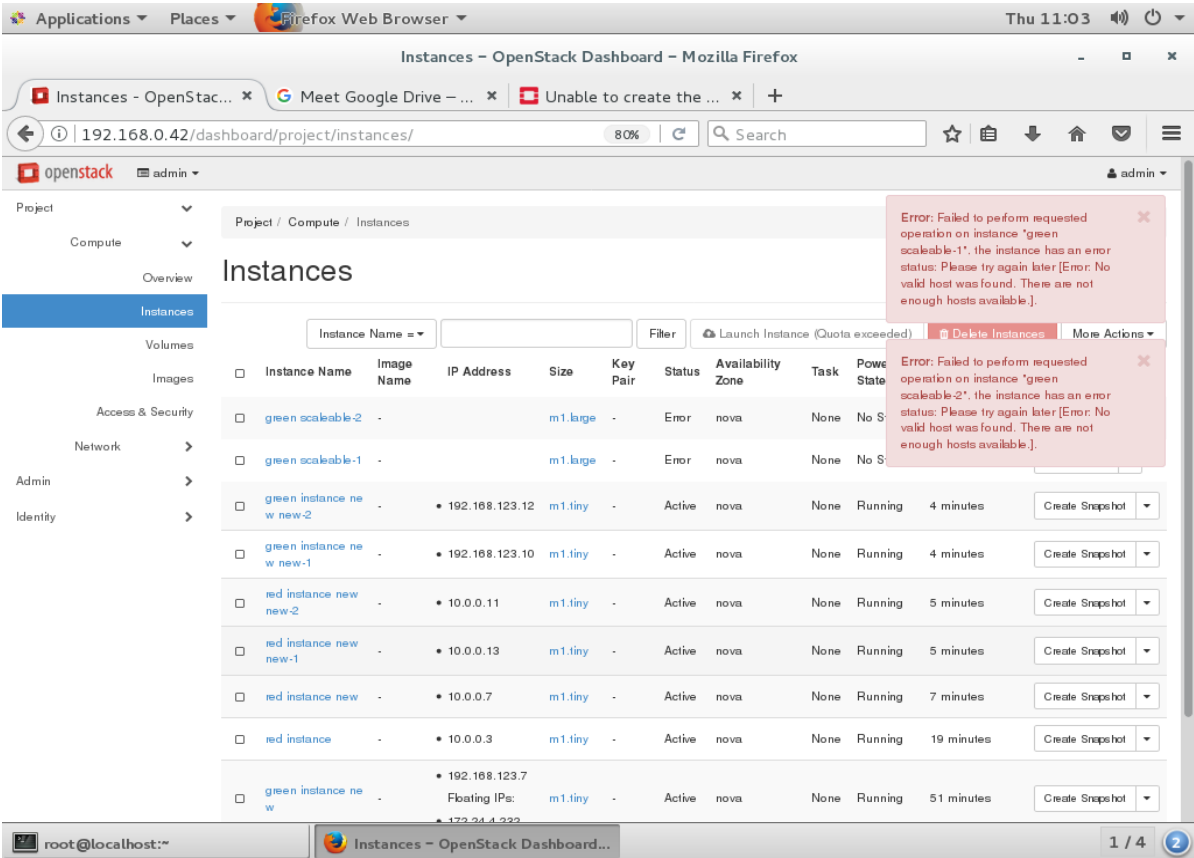


Figure 33: Verification result of launching additional instances after saturation

For more results on this test case, refer [Appendix C, Section 5.3.3 Test Case 2](#).

Performance Testing:

To execute the performance testing, the ideal scenario should use similar hardware specifications for both the set ups. However, due to resource constraints in lab, we have used hardware specifications as mentioned in Table 5.

Hardware Specification:

	Multi Server	Single Server
	Each node run as a separate physical machine.	Each node run as a virtual machine.
Controller Node	2 cores (@ 2.13GHz), 4GB memory, 300GB storage and 1 GBps NIC card	2 Cores (@ 2.40GHz), 4GB Memory, 120GB Storage
Compute Node	16 cores (@ 2.40GHz), 24GB memory 1TB storage and 1 GBps NIC Card	8 Cores (@ 2.40GHz), 10GB Memory, 500GB Storage
Network Node	2 cores (@ 2.13GHz), 4GB memory, and 300GB storage and 1 GBps NIC Card	2 Cores (@ 2.40GHz), 4Gb Memory, 120GB Storage

Table 5: Hardware specifications for Performance Testing

Note: In single server, all the VM instances shares the same NIC card for the external network connectivity but internal communication among VMs will happen via OpenvSwitch.

Benchmarking tool: We have used “Rally” benchmarking tool [17] for OpenStack to carry the performance testing.

1)	Performance comparison between single and multi-server implementation for creation and listing of network information from controller node.	Pass
----	---	-------------

Expected Result: The single server implementation should perform better than multi server in terms of time.

Verification Steps:

1. Login to dashboard as admin
2. Install Rally benchmarking tool on the controller node of Multi Server setup.
3. Create a task configuration file to create and list down the information of 70 networks.
4. Run the rally command to execute the task by running command “rally task start config.json”.

- On execution of the above command, rally will share the required performance statistics as a html file.
- Similarly run steps 1 through 5 for single server setup.
- Compare the obtained results for both setups

Actual Result: The single server implementation performs better than multi server in terms of time. This is because the communication between network and controller service in a single server setup happens internally on the same physical server via openvSwitch.

Multi Server Performance Statistics:

Total durations

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
neutron.create_network	0.608	1.067	1.321	1.407	1.511	1.09	100.0%	70
neutron.list_networks	0.082	0.231	0.373	0.442	0.471	0.239	100.0%	70
total	0.69	1.298	1.694	1.849	1.982	1.329	100.0%	70

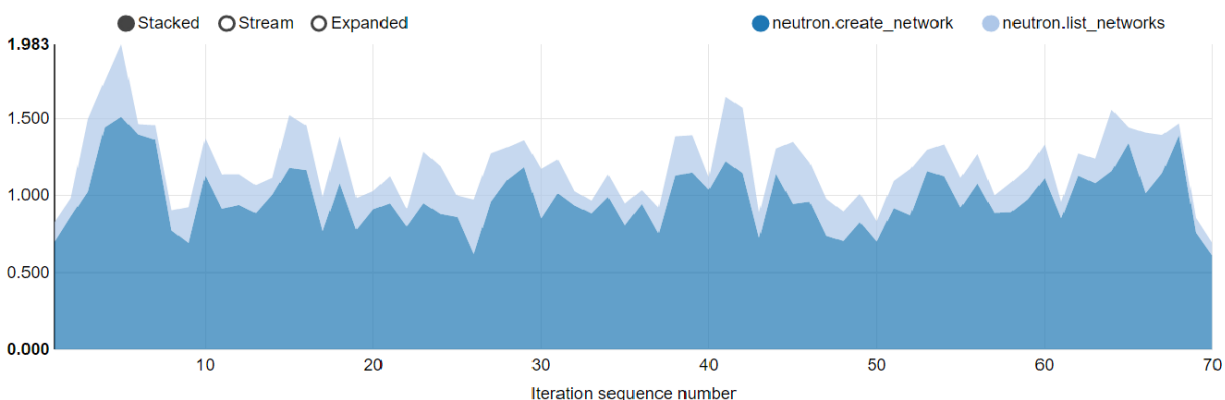


Figure 34: Performance statistics of network task for Multi Server setup

Single Server Performance Statistics:

Total durations

Action	Min (sec)	Median (sec)	90%ile (sec)	95%ile (sec)	Max (sec)	Avg (sec)	Success	Count
neutron.create_network	0.513	0.949	1.188	1.378	1.478	0.97	100.0%	70
neutron.list_networks	0.066	0.192	0.335	0.396	0.424	0.205	100.0%	70
total	0.595	1.172	1.463	1.563	1.746	1.175	100.0%	70

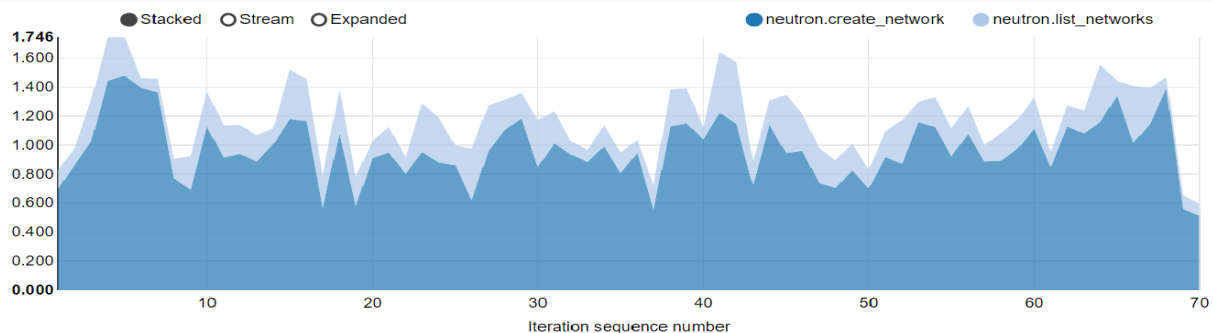


Figure 35: Performance statistics of network task for Single Server setup

2)	Performance comparison between single and multi-server implementation for creation and deletion of 10 virtual instances.	Pass
----	--	------

Expected Result: The single server implementation should perform better than multi server in terms of time.

Verification Steps:

1. Login to dashboard as admin
2. Install Rally benchmarking tool [17] on the controller node of Multi Server setup.
3. Create a task configuration file to create and delete 10 virtual instances
4. Run the rally command to execute the task by running command “rally task start config.json”.
5. On execution of the above command, rally will share the required performance statistics as a html file.
6. Similarly run steps 1 through 5 for single server setup.
7. Compare the obtained results for both setups

Actual Result: The single server implementation performs better than multi server in terms of time. This is because the communication between network, compute and controller services in a single server setup happens internally on the same physical server via openvSwitch.

Multi Server Performance Statistics:

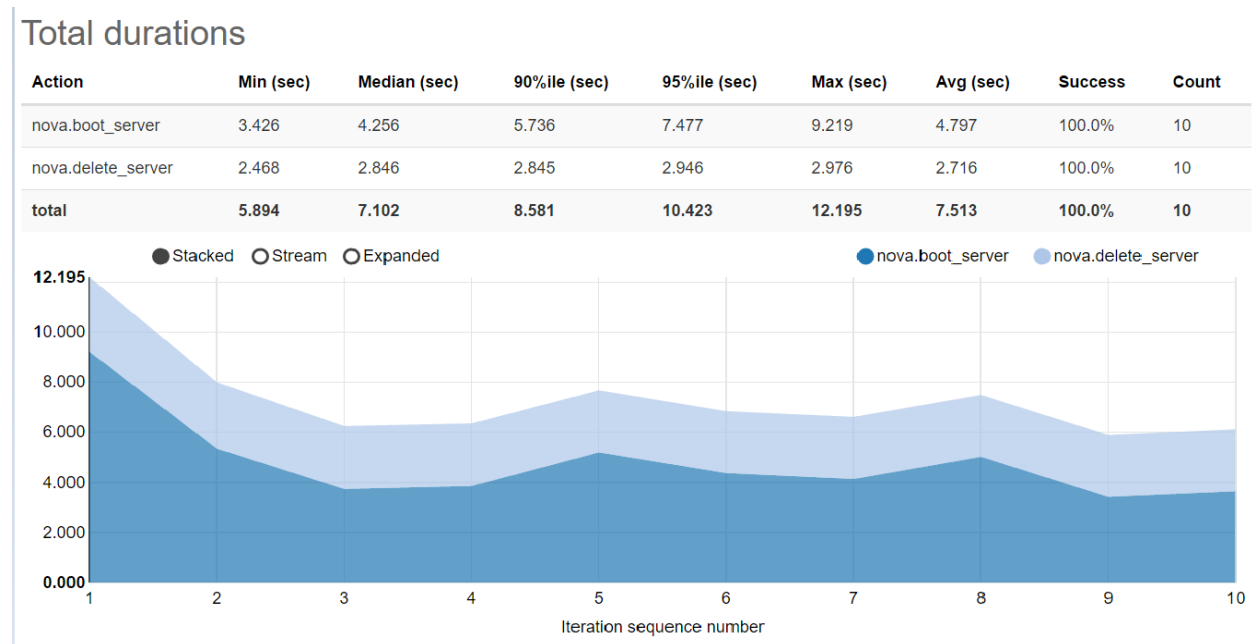


Figure 36: Performance statistics of VM task for Single Server setup

Single Server Performance Statistics:

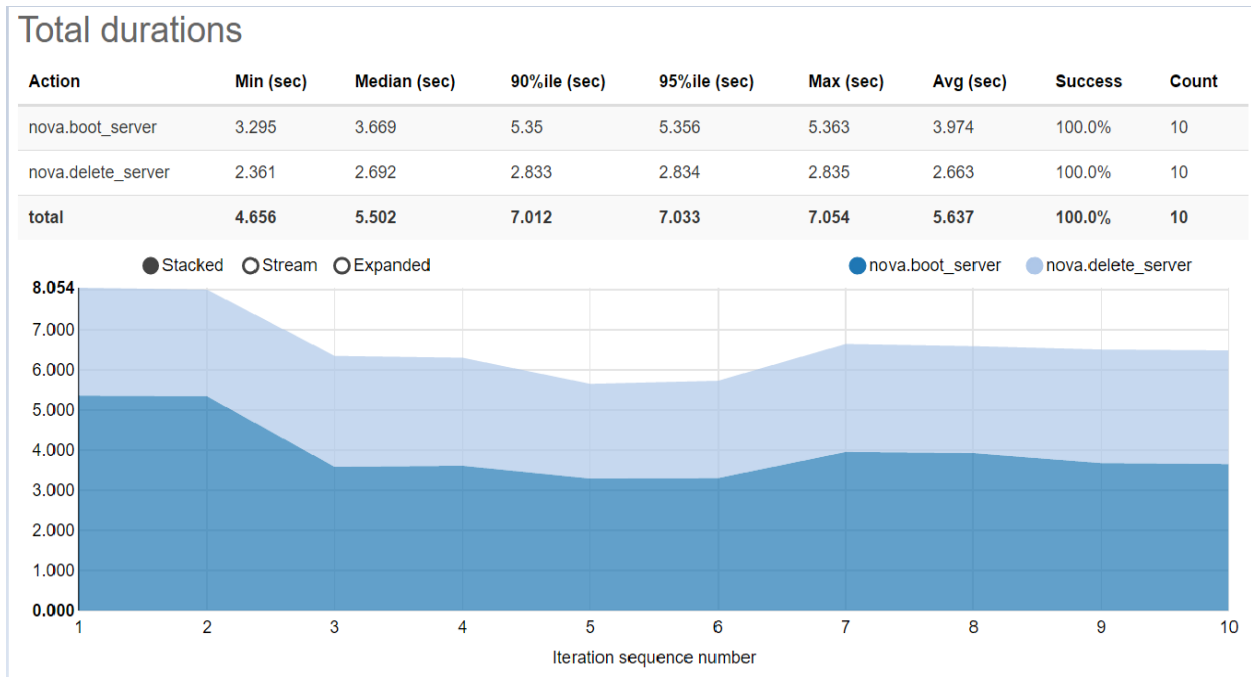


Figure 37: Performance statistics of VM task for Single Server setup

5.3.4 Security Testing:

Security testing is a process intended to reveal flaws in the security mechanisms of an information system. Security requirements may include specific elements of confidentiality, integrity, authentication, availability, authorization. Security testing validates that the given script securely installs OpenStack environment

Tests and Results:

Table 6: Security Testing Tests and Results

Test Case #	Test Case Description and Verification	Result
1)	Connection to the VMs should only be made using key pairs based authentication	Pass

Expected Result: Key-pair based authentications should be enabled

Verification Steps:

1. Login to dashboard
2. Generate key pair and use it while launching a VM. For information on how to generate the keypair, check [Appendix C, Section 5.3.4 Test Case 1](#).

Actual Result:

As seen, the key-pair was generated and was used to login into instance via SSH.

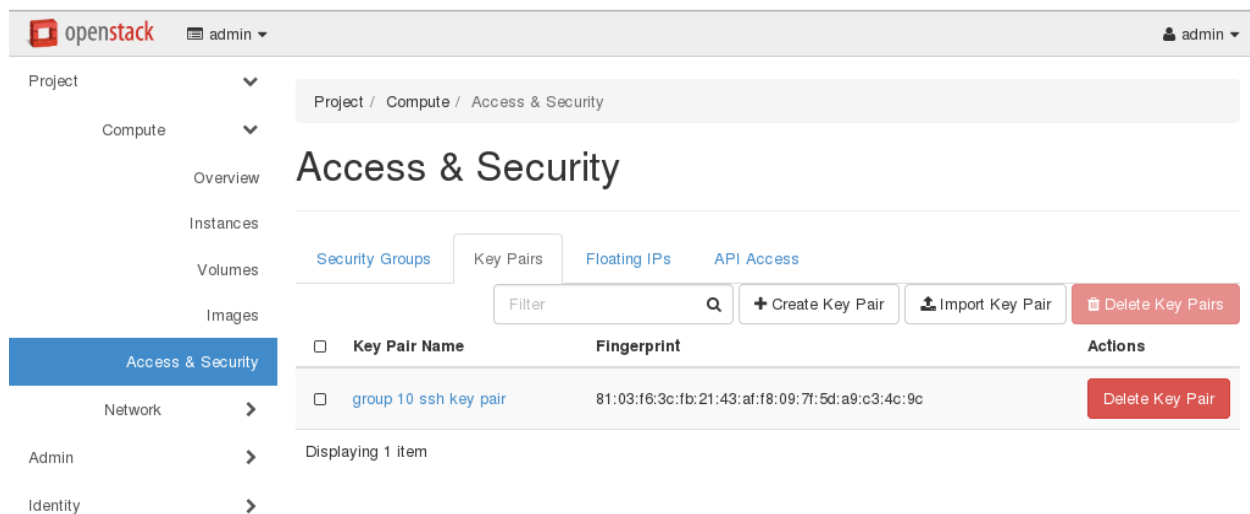


Figure 38: Verification result of generating key pair and using it while launching instance

2)	Key pairs should be made available to the user and he/she should be allowed to change them	Pass
----	--	------

Expected Result: The user should be able to import key pairs

Verification Steps:

1. Login to dashboard as admin
2. Give the key-pair access to a demo user
3. Login as this user to launch instance using the assigned key-pair

Actual Result: The user could launch the VM using the key-pair.

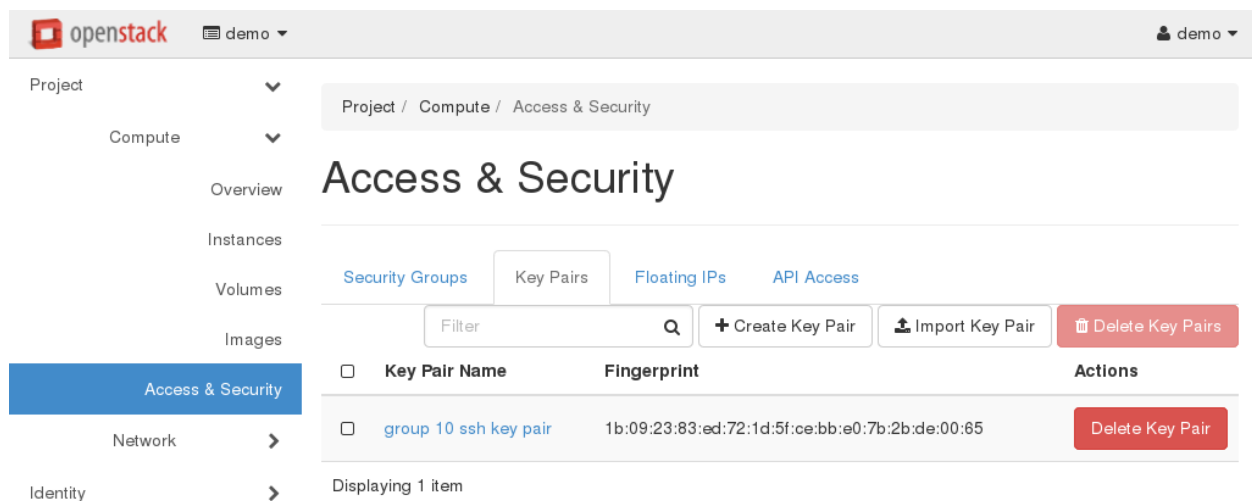


Figure 39: Verification result of enabling demo user to import generated key pair

Chapter 6: Schedule and Personnel

In this section, we list the individual responsibility with the timeline of their completion and progress.

6.1 Milestones:

Following are the milestones of the project

Table 7: Project Milestones

Milestone #	Milestone Description
1	Understanding the OpenStack architecture and installation procedure
2	Installation of OpenStack on a single physical machine with VMs acting as nodes
3	Installation of OpenStack on a cluster of physical machines
4	Automation of OpenStack Newton installation on a single physical machine
5	Automation of OpenStack Newton installation across a cluster of physical machines
6	Consolidation of the automation scripts
7	Testing the automation scripts to check whether the baseline OpenStack Newton environment has been installed properly
8	Verification of the installation by launching VMs and running few applications over them
9	Documentation and Presentation

Some of these milestones were subdivided into tasks. Following tables highlight the important tasks accomplished in those milestones.

Milestone #2 - Installation of OpenStack on a single physical machine with VMs acting as nodes was subdivided into the following tasks

Table 8: Milestone 2 task breakdown

Task #	Task Description
1	Installation of VMs using KVM
2	Installation of VMs using CentOS 7 (prerequisite for OpenStack Newton)
3	Scripts to automate creation of VMs based on user's input
4	Configuration of basic OpenStack Newton network across the VM

5	Wrote scripts for following tasks a) User Configuration b) Network health checkup c) Clean up d) Check Pre-requisites
6	Installation and configuration of OpenStack services on the VMs

Milestone #6 - Consolidation of the automation scripts - was subdivided into the following tasks

Table 9: Milestone 6 task breakdown

Task #	Task Description
1	Deciding on the installation flow to be followed
2	Consolidate all the scripts to facilitate the installation process

Milestone #7 - Testing the automation scripts to check whether the baseline OpenStack Newton environment has been installed properly was subdivided into the following tasks

Table 10: Milestone 7 task breakdown

Task #	Task Description
1	Unit Testing
2	System and Integration Testing
3	Stress and Performance Testing
4	Security Testing

6.2 Project Timeline:

Below table shows high level timeline of tasks performed.

Table 11: Project Timeline

Task	Owner	Status	Target Date
Understand basics of Openstack	Everyone	Completed	03/01/2017
Network configuration in Lab	Jalandhar, Xingyao	Completed	03/06/2017
Openstack Service distribution	Everyone	Completed	03/14/2017
Requirement and Design	Jalandhar, Xingyao & Ronak	Completed	03/22//2017
Multi server Installation	Jalandhar, Xingyao & Hengjin	Completed	04/07/2017
Multi server Verification	Ronak & Manushri	Completed	04/18/2017
Single server Installation	Ronak & Manushri	Completed	04/22/2017
Single server Verification	Jalandhar, Xingyao & Hengjin	Completed	04/27/2017
Merge all the scripts	Everyone	Completed	04/29/2017
Documentation	Everyone	Completed	05/04/2017

6.3 Schedule

The schedule shown below describes the major tasks which were distributed amongst the team members. The following table gives a detailed description of task details in each milestone and their primary owners.

Table 12: Project Schedule and Contributions

Activity	Start Date	End Date	Progress	Owner	Contributor
Understanding basic services of Openstack	02/15/2017	03/01/2017	Completed	Team	
Choose the Operating System	03/02/2017	03/03/2017	Completed	Xingyao	Hengjin
Set up internet access on the machines	03/04/2017	03/06/2017	Completed	Jalandhar	
Different ways of Openstack installation and their feasibilities w.r.t operating system.	03/04/2017	03/10/2017	Completed	Manushri	Jalandhar, Ronak
Installation using DevStack	03/05/2017	03/10/2017	Completed	Xingyao	Hengjin
How to customize the configuration to meet the purpose?	03/05/2017	03/06/2017	Completed	Xingyao	Hengjin
Installation using PackStack	03/06/2017	03/07/2017	Completed	Manushri	Ronak
How to customize the configuration to meet the purpose?	03/07/2017	03/08/2017	Completed	Manushri	Jalandhar
Setup basic networking in a multiple server cluster	03/08/2017	03/10/2017	Completed	Jalandhar	Ronak
Configure Database for the users in multi server cluster	03/11/2017	03/15/2017	Completed	Jalandhar	Xingyao
Configure Identity Service in multi server cluster	03/11/2017	03/15/2017	Completed	Ronak	Jalandhar
Configure Image Service in multi server cluster	03/11/2017	03/15/2017	Completed	Hengjin	Manushri
Configure horizon on multi server cluster	03/11/2017	03/15/2017	Completed	Ronak	Manushri

Set up dedicated GRE tunnel between compute and network node in single server	03/11/2017	03/15/2017	Completed	Xingyao	Jalandhar
Debugging & resolving intermittent networking issues while configuring services on multiple server cluster	03/18/2017	03/20/2017	Completed	Jalandhar	Ronak
Understanding external network requirements to allow VM to communicate to internet in multiple server setting	03/18/2017	03/20/2017	Completed	Team	
Write the script for User input configuration	03/21/2017	03/25/2017	Completed	Hengjin	Xingyao
Write the script for Network health checkup	03/21/2017	03/25/2017	Completed	Manushri	Jalandhar
Write the script for cleanup	03/21/2017	03/25/2017	Completed	Ronak	Hengjin
Finding prerequisites for each service to automate the installation	03/21/2017	03/25/2017	Completed	Hengjin	Ronak
Automate Installation of VM on KVM	03/26/2017	04/05/2017	Completed	Manushri	
Automation of User input configuration	03/26/2017	04/05/2017	Completed	Xingyao	
Automate the basic network setup for single and multi server	03/26/2017	04/05/2017	Completed	Jalandhar	Ronak
Automate installation of OpenStack services for single and multi server	04/10/2017	04/10/2017	Completed	Manushri	Jalandhar
Merging various scripts and consolidating them to function as a single appliance	04/12/2017	04/13/2017	Completed	Hengjin	Manushri
Unit Testing of single server and multiple server setup	04/12/2017	04/27/2017	Completed	Ronak	Xingyao
System Testing of single server and multiple server setup	04/12/2017	04/27/2017	Completed	Manushri	Jalandhar

Stress and performance testing of single server and multiple server setup	04/12/2017	04/27/2017	Completed	Hengjin	Xingyao
Security testing of single server and multiple server setup	04/12/2017	04/27/2017	Completed	Jalandhar	Manushri
Verification of executing the scripts via USB	04/27/2017	04/29/2017	Completed	Team	
Documentation for Homework 3 and final report	03/01/2017	05/02/2017	Completed	Team	
In-class and final project presentation and Oral exam	04/27/2017	05/04/2017	Completed	Team	

References

- [1] Brief outline on OpenStack:
<https://www.openstack.org/>
- [2] Introduction to OpenStack:
<https://docs.openstack.org/security-guide/introduction/introduction-to-openstack.html>
- [3] Overview of services provided by the OpenStack Newton Release:
<https://docs.openstack.org/newton/install-guide-rdo/overview.html>
- [4] Information regarding Nova:
<http://www.webopedia.com/TERM/O/openstack-nova.html>
- [5] Information regarding Neutron:
www.sdxcentral.com/cloud/open-source/definitions/what-is-openstack-quantum-neutron/
- [6] Information regarding Keystone:
<http://blog.flux7.com/blogs/openstack/tutorial-what-is-keystone-and-how-to-install-keystone-in-openstack>
- [7] Information regarding Horizon:
<https://docs.openstack.org/developer/horizon/intro.html>
- [8] Information regarding Glance:
<https://docs.openstack.org/developer/glance/>
- [9] Information regarding Cinder:
<http://searchstorage.techtarget.com/definition/Cinder-OpenStack-Block-Storage>
- [10] Information regarding Swift:
<https://wiki.openstack.org/wiki/Swift>
- [11] Openstack training guides
<https://docs.openstack.org/draft/training-guides/>
- [12] Openstack operations guide
https://docs.openstack.org/openstack-ops/content/example_architecture.html
- [13] Operations of virtual router
<http://www.rackspace.com/cloud/servers/vrouter>
- [14] OpenStack: The Open Source Cloud Operating System.
<http://www.openstack.org/software/openstack-dashboard/>
- [15] OpenStack hardware requirements:
<https://docs.openstack.org/mitaka/install-guide-ubuntu/overview.html#figure-hwregs>
- [16] RDO Frequently asked questions:
<https://www.rdoproject.org/rdo/faq/>
- [17] Rally Benchmarking Tool:
https://rally.readthedocs.io/en/0.0.4/tutorial/step_1_setting_up_env_and_running_benchmark_from_samples.html

Appendices

Appendix A: OpenStack services installed

Following Basic OpenStack services were installed:

- **Nova(Compute):**

OpenStack Compute (Nova) is a cloud computing fabric controller, which is the main part of an IaaS system. It is designed to manage and automate pools of computer resources. It can work with widely available virtualization technologies, bare metal and high-performance computing (HPC) configurations. KVM, VMware, and Xen are available choices for hypervisor technology. It is written in Python and uses many external libraries. Compute's architecture is designed to scale horizontally on standard hardware with no proprietary hardware or software requirements and provide the ability to integrate with legacy systems and third-party technologies.

The Nova component is supposed to be running the compute jobs for the cluster. So the jobs that the users submit to the cluster will be delegated to the compute nodes by the controller. For delegating the work off to nova, the controller should have the knowledge of all the compute nodes in the cluster and the details of the nodes regarding their current status and the capacity of compute and memory of each node to delegate proper amount of work to corresponding nodes. This is where the nova component at the controller nodes comes in. This component keeps the information of the occupancy and other details of all the compute nodes in the cluster to delegate work effectively.

- **Neutron(Network):**

OpenStack Networking (Neutron) is a system for managing networks and IP addresses. OpenStack Networking ensures the network is not a bottleneck or limiting factor in a cloud deployment, and gives users self-service ability, even over network configurations.

OpenStack Networking provides networking models for different applications or user groups. Standard models include flat networks or VLANs that separate servers and traffic. OpenStack Networking manages IP addresses, allowing for dedicated static IP addresses or DHCP. Floating IP addresses let traffic be dynamically rerouted to any resources in the IT infrastructure, so users can redirect traffic during maintenance or in case of a failure.

Users can create their own networks, control traffic, and connect servers and devices to one or more networks. Administrators can use software-defined networking (SDN) technologies like OpenFlow to support high levels of multi-tenancy and massive scale. OpenStack networking provides an extension framework that can deploy and manage additional network services—such as intrusion detection systems (IDS), load balancing, firewalls, and virtual private networks (VPN).

- **Keystone(Identity):**

OpenStack Identity (Keystone) provides a central directory of users mapped to the OpenStack services they can access. This component of Openstack is responsible to enforce security in the deployment of services. It component takes care of Authentication, Authorization and Accounting related to Openstack deployment usage by any user. Keystone is the Identity Store which is the collection of all the data related to the various users that are known to the controller and their rights. The Identity Store is accessed every time a user requests access to any functionality of the Openstack instance to check whether the user actually has the rights to access the functionality he is requesting to use. Additionally, the catalog provides a queryable list of all of the services deployed in an OpenStack cloud in a single registry. Users and third-party tools can programmatically determine which resources they can access.

- **Glance (Image):**

OpenStack Image (Glance) provides discovery, registration, and delivery services for disk and server images. Stored images can be used as a template. It can also be used to store and catalog an unlimited number of backups. The Image Service can store disk and server images in a variety of back-ends, including Swift. The Image Service API provides a standard REST interface for querying information about disk images and lets clients stream the images to new servers.

Glance adds many enhancements to existing legacy infrastructures. For example, if integrated with VMware, Glance introduces advanced features to the vSphere family such as vMotion, high availability and dynamic resource scheduling (DRS). vMotion is the live migration of a running VM, from one physical server to another, without service interruption. Thus, it enables a dynamic and automated self-optimizing datacenter, allowing hardware maintenance for the underperforming servers without downtimes. Other OpenStack modules that need to interact with Images, for example Heat, must communicate with the images metadata through Glance. Also, Nova can present information about the images, and configure a variation on an image to produce an instance. However, Glance is the only module that can add, delete, share, or duplicate images.

- **Horizon (Dashboard):**

OpenStack Dashboard (Horizon) provides administrators and users with a graphical interface to access, provision, and automate deployment of cloud-based resources. This service is deployed on the controller because it is the initial point of contact for the clients to access the services provided by the cluster. The design accommodates third party products and services, such as billing, monitoring, and additional management tools. The dashboard is also brand-able for service providers and other commercial vendors who want to make use of it. The dashboard is one of several ways users can interact with OpenStack resources. Developers can automate access or build tools to manage resources using the native OpenStack API or the EC2 compatibility API. It is only logical to allow the users to connect to the node where all the decisions regarding the required resources are taken so that there is no redirection from a node to another even before the requirement is worked upon to increase the latency of the service experienced

by the user.

Appendix B: Screenshots of additional Results

Section 4.4: Results for list of installed services

2. List of compute services

The screenshot shows the OpenStack Admin dashboard at the URL `192.168.0.42/dashboard/admin/info/`. The user is logged in as 'admin'. The navigation menu on the left includes Project, Admin, and System. The main content area is titled 'System Information' and has tabs for Services, Compute Services, Block Storage Services, and Network Agents. The 'Compute Services' tab is active, displaying a table of installed services.

Name	Host	Zone	Status	State	Last Updated
nova-cert	localhost.localdomain	internal	Enabled	Up	0 minutes
nova-consoleauth	localhost.localdomain	internal	Enabled	Up	0 minutes
nova-scheduler	localhost.localdomain	internal	Enabled	Up	0 minutes
nova-conductor	localhost.localdomain	internal	Enabled	Up	0 minutes
nova-compute	localhost.localdomain	nova	Enabled	Up	0 minutes

Displaying 5 items

Version: 10.0.2

3. List of Block Storage services

The screenshot shows the OpenStack Admin dashboard at the URL `192.168.0.42/dashboard/admin/info/`. The user is logged in as 'admin'. The navigation menu on the left includes Project, Admin, and System. The main content area is titled 'System Information' and has tabs for Services, Compute Services, Block Storage Services, and Network Agents. The 'Block Storage Services' tab is active, displaying a table of installed services.

Name	Host	Zone	Status	State	Last Updated
cinder-volume	localhost.localdomain@lvm	nova	Enabled	Up	0 minutes
cinder-scheduler	localhost.localdomain	nova	Enabled	Up	0 minutes

Displaying 2 items

Version: 10.0.2

4. List of network services

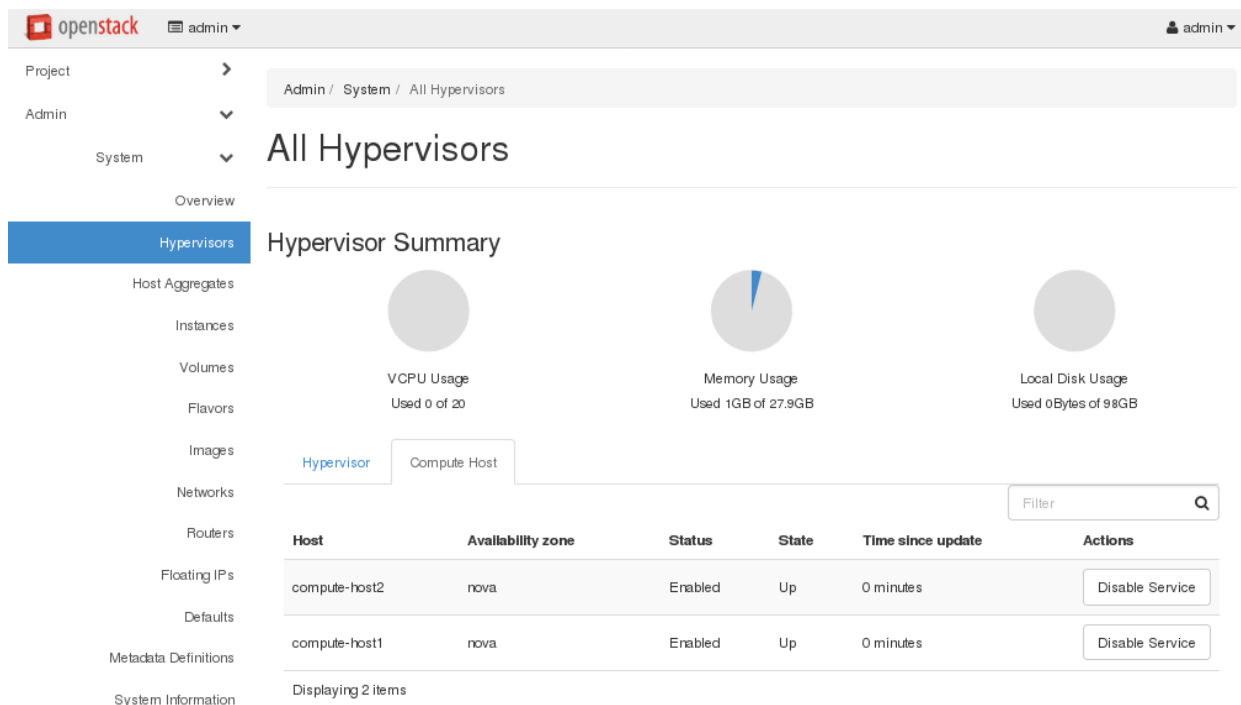
The screenshot shows the OpenStack dashboard interface. The browser address bar displays `192.168.0.42/dashboard/admin/info/`. The dashboard header includes the OpenStack logo, the user `admin`, and a search bar. The left sidebar contains a navigation menu with categories: Project, Admin, System, Overview, Hypervisors, Host Aggregates, Instances, Volumes, Flavors, Images, Networks, Routers, Floating IPs, Defaults, Metadata Definitions, and Identity. The `System` category is expanded, showing `System Information` as the selected item. The main content area is titled `System Information` and includes a breadcrumb trail: `Admin / System / System Information`. Below the title, there are tabs for `Services`, `Compute Services`, `Block Storage Services`, and `Network Agents`. The `Network Agents` tab is active. A table lists the network agents, with columns for `Type`, `Name`, `Host`, `Status`, `State`, `Last Updated`, and `Actions`. The table contains five rows of data, all showing agents running on `localhost.localdomain` with a status of `Enabled` and state of `Up`. A `View Routers` button is located in the `Actions` column of the first row. The bottom right corner of the dashboard displays `Version: 10.0.2`.

Type	Name	Host	Status	State	Last Updated	Actions
L3 agent	neutron-l3-agent	localhost.localdomain	Enabled	Up	0 minutes	View Routers
DHCP agent	neutron-dhcp-agent	localhost.localdomain	Enabled	Up	0 minutes	
Open vSwitch agent	neutron-openvswitch-agent	localhost.localdomain	Enabled	Up	0 minutes	
Metadata agent	neutron-metadata-agent	localhost.localdomain	Enabled	Up	0 minutes	
Metering agent	neutron-metering-agent	localhost.localdomain	Enabled	Up	0 minutes	

Appendix C: Additional screenshots of Verification Results

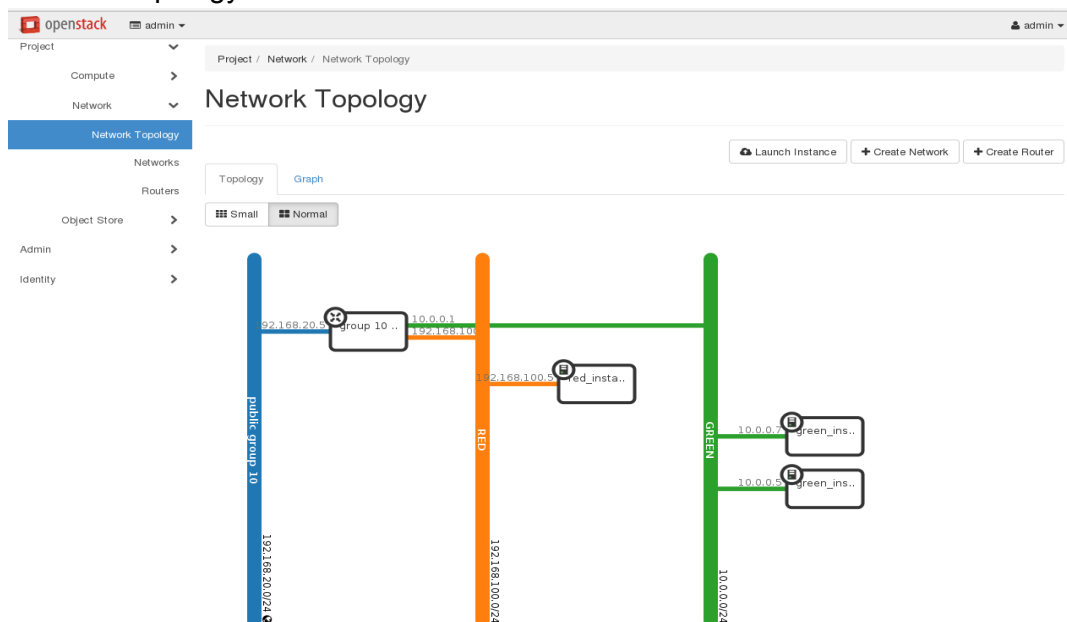
Section 5.3.1 Test Case 10

Dashboard view of addition of new compute node



Section 5.3.2 Test Case 1 to 4

Network Topology



Section 5.3.2 Test Case 5

Image was privately created by admin

The screenshot shows the OpenStack dashboard interface. The left sidebar contains a navigation menu with options: Project, Compute, Overview, Instances, Volumes, Images (selected), Access & Security, Network, Object Store, Admin, and Identity. The main content area is titled 'Centos 7' and includes a 'Launch' button. Below the title, there are two tabs: 'Image' and 'Security'. The 'Image' tab displays the following details:

ID	7431159a-ee61-45de-90da-daf29faf5d79
Type	Image
Status	Active
Size	680.00 MB
Min. Disk	0
Min. RAM	0
Disk Format	ISO
Container Format	BARE
Created At	2017-04-18T23:57:43Z
Updated At	2017-04-18T23:57:47Z

The 'Security' tab displays the following details:

Owner	c6e8ab8951c749f4a339bb5583d89dbe
Filename	-
Visibility	Private
Protected	No
Checksum	d2ec6cfa7cf6d89e484aa2d9f830517c

Below the tabs, there is a 'Custom Properties' section with the following details:

schema	/v2/schemas/image
Virtual Size	
Description	Centos 7 Image Group 10
file	/v2/images/7431159a-ee61-45de-90da-daf29faf5d79/file
Tags	

Admin was able to see both private and public images.

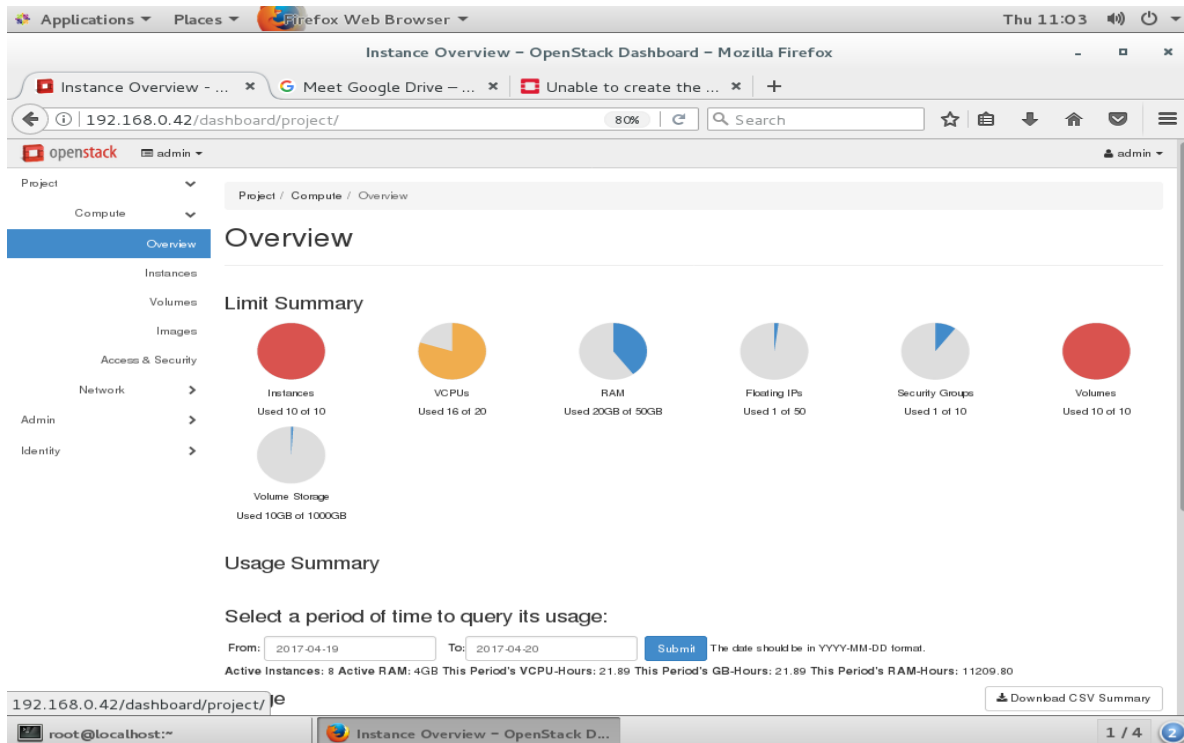
The screenshot shows the OpenStack dashboard interface. The left sidebar contains a navigation menu with options: Project, Compute, Overview, Instances, Volumes, Images (selected), Access & Security, Network, Object Store, Admin, and Identity. The main content area is titled 'Images' and includes a search bar, a 'Create Image' button, and a 'Delete Images' button. Below the buttons, there is a table listing the images:

	Owner	Name ^	Type	Status	Visibility	Protected	Disk Format	Size	
<input type="checkbox"/>	admin	Centos 7	Image	Active	Private	No	ISO	680.00 MB	Launch
<input type="checkbox"/>	services	cirros	Image	Active	Public	No	QCOW2	12.67 MB	Launch

Below the table, it says 'Displaying 2 items'.

Section 5.3.3 Test Case 2

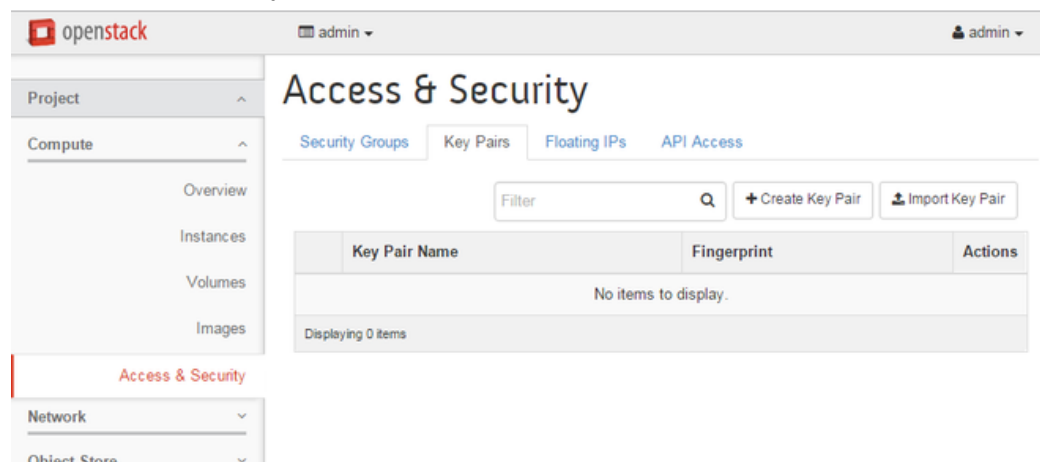
Dashboard view of saturation of resources



Section 5.3.4 Test Case 1

Steps to generate keypair in Horizon

1. Click "Compute" under the "Project" option in the Horizon left-hand menu.
2. Select "Access & Security".
3. Click the "Key Pairs" tab.



4. Click “+Create Key Pair”.
5. Name your new key pair and click “Create Key Pair”.

Create Key Pair

Key Pair Name *

openstacktip

Description:

Key pairs are ssh credentials which are injected into images when they are launched. Creating a new key pair registers the public key and downloads the private key (a .pem file).

Protect and use the key as you would any normal ssh private key.

Cancel Create Key Pair

6. Click Access & Security again to see your new key pair.

openstack admin admin

Access & Security

Security Groups Key Pairs Floating IPs API Access

Filter Q + Create Key Pair Import Key Pair Delete Key Pairs

Key Pair Name	Fingerprint	Actions
openstacktip	ba:b6:4d:9d:0c:6b:3f:3e:fa:3e:81:fc:b9:4a:30:10	Delete Key Pair

Displaying 1 item

Appendix D: Steps to follow to install Openstack using scripts

Following are the steps to be followed to install the cloud Infrastructure:

1. Insert the USB stick to one of the node.
2. Open the terminal
3. Run the command “bash -c install.sh”.

To view all the scripts, click on the following NCSU Github link.

<https://github.ncsu.edu/xhuang17/OpenStackonaStick>