UNIVERSITY OF OSLO
Department of Informatics

A Study Evaluating Open Source Cloud Computing Platforms

Behzad Pashaie
behzadp@ifi.uio.no
s181785@stud.hioa.no

Network and System Administration
Oslo University College

31.05.2013

Table of Contents

List of Figures

List of Tables

Abstract

In the last few years several open source private cloud platforms have appeared on the market side by side of many proprietary peers. Evaluating the existing open source private cloud software from different aspects makes it possible to determine which open source cloud software fits best to our hardware resources and budget. This thesis evaluates two of the major open source cloud platforms, OpenStack and OpenNebula, by investigating the deployment of the cloud platforms, and analyzing the user community and developer community of the software. Different criteria are defined in the form of questions, which are answered through systematic experiments.

Acknowledgements

It is an honor for me to express my deepest appreciation to the following people and recognize their support in the process of writing this thesis:

- My beloved family for their emotional support.

- Hårek Haugerud my supervisor, for initiating this master program, and for his great support, structural feedback, encouragement and patience with me whilst writing this thesis.

- Kyrre Begnum for his support, guidance and feedback on this thesis.

- Oslo University College and the University of Oslo for making this master program possible. I especially wish to thank the skillfull and kind teachers Tore Audun Høie, AEleen Frisch, Amir Maqbool Ahmed, Siri Fagernes, Tore Møller Jonassen, Ismail Hassan and Alfred Sewitsky Bratterud.

- Anette Haugen for thorough feedback on language and structure.

- All my classmates and friends for their continuous support and understanding during this master.

Chapter 1

1.Introduction

The next section explains the motivation behind the importance of evaluating open source cloud platforms. This chapter continues with the problem statement and it ends with a section briefly describing the structure of this thesis.

1.1 Motivation

In recent years cloud computing has transformed the computing landscape. In fact, cloud computing is a combination of several technologies among which virtualization and networking stand at core. Considering all the possibilities that cloud computing provides, many businesses and institutes have eagerly moved their attention toward adopting this new technology into their IT infrastructure. Generally speaking, cloud services are accessible as public cloud services or in the form of private clouds. The next chapter provides detailed background knowledge to cloud computing, various cloud models and different existing cloud provider software.

There are several advantages that give significant importance to cloud computing. Some of these advantages are explained in the following list:

- Cost Reduction:
  Public cloud services provide hardware and software computing resources at an affordable price, thus companies reduce a lot of expenses by eliminating the need to pay for physical hardware, software licensing and maintenance in addition to employee expenses. Depending on the amount of computing resources required by a company, deploying a private cloud may reduce the expenses even more.

- Flexibility and scalability:
  Using cloud services provides high flexibility since the services can be used whenever and as much as they are needed to quickly meet business demands.

- Availability:
  Cloud computing services can be accessed from any where over the network.

- Environment-friendly:
  Deploying cloud computing reduces the energy used by companies to empower the hardware resources, which decreases their carbon footprint. Some studies show that deploying cloud computing can result in at least 30% less energy consumption and carbon emissions than using on-site servers.[1]

- Increased collaboration:
  Since cloud resources can be shared over the network, it allows researchers or employees to collaborate with each other from far distances.

Nowadays several proprietary and open source cloud platforms exist. There are many reasons for organizations to prefer the use of open source cloud platforms to manage their IT infrastructures. Generally this could be due to organizational cultures, security, regulatory concerns or, most importantly, for economical reasons. On the subject of cloud platforms, the following points (argued among cloud experts) may show the advantages of deploying open source cloud platforms over proprietary cloud platform[2]:

- Depending on an API provided by a proprietary vendor results in incompatibility with any other private or public cloud platform, which results in vendor lock-in.

- Proprietary software does not necessarily provide better functionality. This is because open source software may have hundreds of contributors worldwide developing new functionality, fixing bugs, and enabling speed and quality of development.

- Proprietary cloud platforms do not scale without restrictions. Most of proprietary software follow the key principle “use more, pay more”, while the best open source solutions can freely be downloaded and used.

- Open source clouds provide good support. There are now companies, such as Canonical, which provide comprehensive support for both private open source clouds and virtual machine instances in the public cloud.

When a number of virtual machines in the cloud grow, deploying and managing updates and upgrades becomes harder. The major open source cloud platforms solve this issue via dynamic service deployment, which is based on a minimal number of virtual machine images.

Even though all open source cloud platforms aim to provide a private cloud, all of them will vary from different aspects like projects’ start date, programming languages, philosophy of the community, activeness of the software communities, etc. Depending on these differences each open source cloud platform will have its own characteristics concerned with complexity of the software, learning curve, performance, security, etc.

The main challenge rises when organizations have to determine which open source cloud platform will fit their requirements best. As discussed earlier in this chapter, cloud computing is an encapsulation of several technologies, making cloud platforms quite complex systems. A solution to this problem is to evaluate the existing open source cloud software. However, evaluating all open source cloud platforms remains a very challenging task. The challenge will mainly depend on the evaluation level - the deeper the evaluation the heavier the task.

It should be noted that this thesis was also written partly on request from the Center for Information Technology at the University of Oslo (USIT), which has recently decided to deploy a private cloud into the university's IT infrastructure.

Thus this thesis aims to take part in the challenge of evaluating OpenStack and OpenNebula that are two of the leading market open source IaaS providers.

1.2 Problem Statement

Between OpenStack and OpenNebula open source cloud platforms, which one suits better to deploy a private cloud? Evaluation of these two open source cloud platforms will be based on the following questions:

- What are the evaluation criteria needed in order to make a thorough comparison of cloud computing platforms.
- Which open source cloud platform is better in the deployment process?
- Which open source cloud platform has the more active and supporting user community?
- Which open source cloud platform is more complex?

1.3 Structure of the Thesis

This thesis is structured in the following way:

Chapter 2 (Background) provides related work and literature. Chapter 3 (Methodology) explains the methods and techniques used in this thesis. Chapter 4 (Results) contains process and observations of deploying OpenStack and OpenNebula open source cloud platforms. Chapter 5 (Discussion and Future Work) provides an overall evaluation of the results, problems encountered and a view towards further research required. Chapter 6 (Conclusions) is the last chapter and answers the problem statement of this thesis. The Bibliography section provides a list of references used in this thesis. The Appendix section lists all scripts written to automate the process of cloud deployments.

Chapter 2

2. Background and Literature

This chapter provides a background of the technologies used in this thesis and a review of related work.

2.1 Cloud Computing

Today computational tasks are required in all aspects of modern human life everywhere and all the time. Cloud computing as a next stage in evolution of the Internet, has risen as a solution to fulfill the requirements for heavy computational tasks without acquiring expensive hardware and software to do the service. Cloud computing provides the means through which everything from computing power to computing infrastructure, applications, business processes to personal collaboration can be delivered as a service wherever and whenever needed. National Institute of Standards and Technology (NIST) defines cloud computing as follows:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."[3]

Virtualization and autonomic computing are two of the main technologies that lie at the center of cloud computing. Virtualization makes the hardware available as software. Virtualization enables faster IT operations by providing the required agility, and increases infrastructure utilization to reduce expenses. Autonomic computing automates the process so that the user can provision resources on-demand. Automation minimizes user involvement to speed up the process and reduces the possibility of human errors.[4]

As pointed out earlier, one of the major benefits of deploying cloud technology is costs reduction. For example, and according to Forbes, in 2009 the revenue of cloud services was over $58.6 billion.[5]With cloud computing accounting for just 2.3 percent of the global market, there is big a capacity for growth.[5] The Gartner project expects the revenue for cloud services to approach $152.1 billion in 2014.[5] One reason for the increased demand for cloud computing is the explosive growth of data.

According to projections by Century Link [6], by 2015 the amount of data being generated and replicated in the world will have increased by four times.[5]

Today cloud computing is delivered in several different service models. These service models will be covered in more detail in the Background chapter, but for now we note that the three fundamental service models are listed and briefly described as[7]:

- Software-as-a-Service (SaaS):
  SaaS provides complete application software to the end users via a web portal and web service technologies. The fee to use these services is paid by credit card or bank accounts. Applications provided by SaaS are run, maintained, and supported by a service vendor[8].

- Platform-as-a-Service (PaaS):
  PaaS provides a computing platform and a solution stack as a service. In this model, the end user creates and develops software using tools and/or libraries from the service vendor. The end user has to control the software deployment and configuration settings. The service vendor provides the networks, servers, storage and other services.[9]

- Infrastructure-as-a-Service (IaaS):
  IaaS delivers computing resources in the form of hardware, networking, and storage services in virtual form. It can include the delivery of operating systems and virtualization technology to manage the resources.[10]

Among the various cloud computing models, the Infrastructure-as-a-Service (IaaS) model is the focus of this thesis. The cloud computing services and platforms are available in both proprietary and open source free forms. The proprietary cloud vendors deliver public cloud services as a pay-per-use model. The following are some of the main cloud computing platforms [11, 12]:

- Amazon's Elastic Compute Cloud (EC2) is probably the most generalized and best known of the cloud computing service offerings.
- IBM Computing on Demand or Blue Cloud is a highly enterprise-focused cloud computing offering that can cross over between public and private cloud applications.
- Microsoft's Azure cloud computing, based on Microsoft Vista and .NET technology, includes both cloud computing and cloud-hosted extension services. It also supports public and private cloud computing plans.
- Sun Cloud, like IBM's offering, is available both in public and private cloud forms. Since Oracle is acquiring Sun, this offering may change over time.
- Salesforce.com's Force.com cloud is easily integrated with Salesforce.com's application tools.
- Google's AppEngine cloud targets particularly web developers and web hosting applications.

2.2 OpenStack

Rackspace Hosting and NASA released OpenStack as a new cloud management platform in July 2010. The goal of OpenStack is to enable organizations to create and offer cloud-computing services. The first community version of OpenStack is called Austin and the code of it was a combination of NASA's Nebula platform and Rackspace's cloud files platform. OpenStack updates the software regularly every six months. Currently more than 150 companies have joined the OpenStack project among which are AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP, IBM, NEC, VMware and Yahoo. [13]

OpenStack collaborates with developers and cloud computing technologists to produce a global open source cloud platform to deliver solutions for all types of public and private clouds. OpenStack is intended to be simple to implement, massively scalable, and feature-rich. This open source software is released under the terms of the Apache License. By the time of writing this thesis, the OpenStack community consists over 9429 technologists, developers, researchers, corporations and cloud-computing experts within 87 countries.[14]

OpenStack is mostly developed to be used on the Linux operating system. This Software enables administrators to control large pools of compute-, storage, and networking resources throughout datacenters through OpenStack Dashboard, and provides-end node users with a web interface to provision resources.

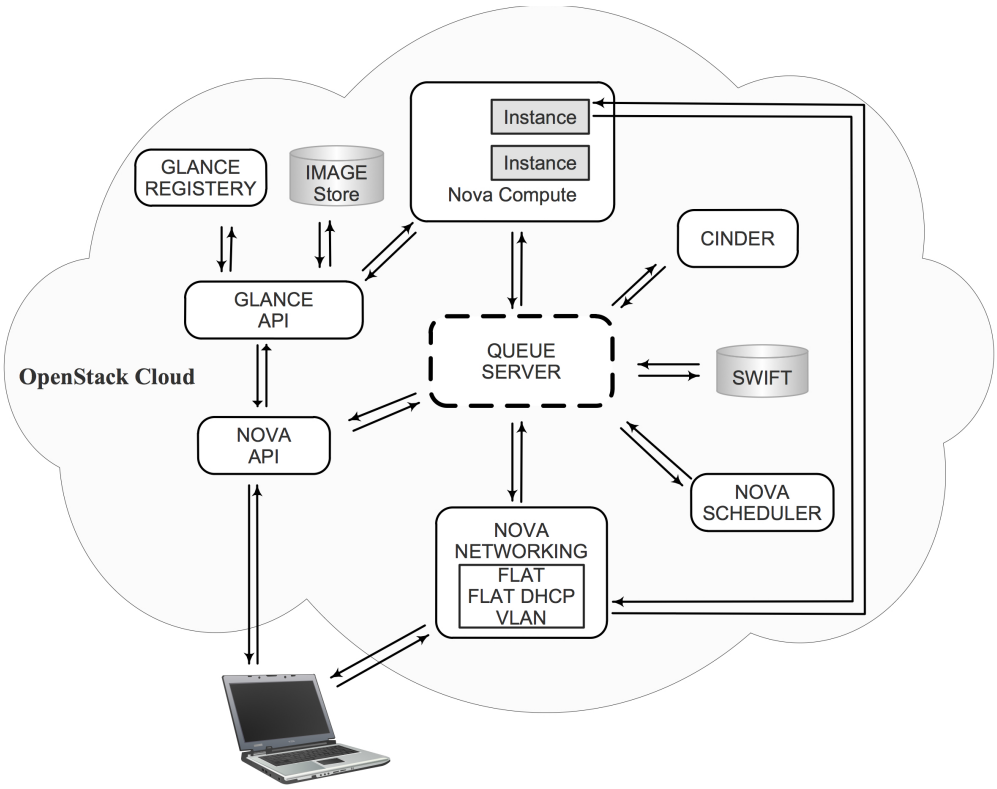

Figure 2.1Overview of OpenStack component correlation

Figure 1.1 provides an overview of various OpenStack components and how they communicate. The modular architecture of OpenStack consists of the following inter-related components[13]:

- OpenStack Compute (code-name Nova):
  This is the component OpenStack is written in Python. Nova provides virtual servers upon demand. It manages and automates pools of compute resources. It supports many available virtualization technologies such as KVM, Xen and VMWare.

- OpenStack Object Storage (code-name Swift):
  Swift provides a large scalable redundant storage system, where objects and files are stored to multiple disk drives spread throughout multiple servers in the data center. This component is responsible of ensuring data replication and integrity across the cluster.

- OpenStack Image Service (code-name Glance):
  Glance provides a repository for virtual disk images. These disk images are used in OpenStack Compute. Glance is responsible for discovery, registration, and delivery services of virtual disk images. Glance enables clients to register new virtual disk images, queries for information on publicly available disk images, and the use of Glance's client library for streaming virtual disk images. While this service is technically optional, a big cloud infrastructure will need it. Glance allows uploading private and public disk images of the following formats[15]:

    - Raw
    - Machine (kernel/ramdisk outside of image, a.k.a. AMI)
    - VHD (Hyper-V)
    - VDI (VirtualBox)
    - qcow2 (Qemu/KVM)
    - VMDK (VMWare)
    - OVF (VMWare, others)

- OpenStack Networking (code-name Quantum):
  Quantum provides network-connectivity-as-a-service between interface devices (e.g. vNICs). The service allows users to create networks and attach interfaces to them. OpenStack Networking ensures the network will not be the bottleneck or limiting factor in a cloud deployment. Quantum has a pluggable architecture, which supports many popular networking vendors and technologies.

- OpenStack Identity (code-name Keystone):
  Keystone is the central authentication and authorization component in an OpenStack cloud. It also provides the following services[16]:

    - **Identity** - provides authentication credential validation and data about Users, Tenants and Roles and other associated metadata.
    - **Token** - validates and manages Tokens used for authenticating requests once a user/tenant’s credentials have already been verified.

- **Catalog** - provides an endpoint registry to be used for endpoint discovery.
- **Policy** - provides a rule-based authorization engine.

- OpenStack Dashboard (code-name Horizon):
  Horizon provides a modular web-based user interface for all the OpenStack services. It allows users to perform most operations such as launching an instance, assigning IP addresses and setting access controls. Third party products and services, such as billing, monitoring and additional management tools, can be plugged into Dashboard. Service providers and other commercial vendors can have their own brand of Dashboard if they require.

- OpenStack Block Storage (code-name Cinder):
  Cinder provides persistent block storage to guest VMs. Cinder as a new service for OpenStack separates storage management logic. Before the Essex version of OpenStack Cinder was included in Nova and was called nova-volume. Since Nova became a large piece of software and logical interdependencies between components within Nova became more and more complex, the project developers decided to make storage management logic separate from Nova, and thus Cinder was developed.[17]

2.3 OpenNebula

In 2005 OpenNebula started as a cloud research project by Distributed Systems Architecture (DSA) Research Group[18] and it was released publicly in March 2008. OpenNebula is an open source industry standard for data center virtualization. It provides an effective open source toolkit to build private, public, and hybrid IaaS either in scientific or in business environments.

OpenNebula is one of the very few open source cloud platforms that provide support for many existing virtualization technologies like KVM, Xen, VMWare, Hyper-V, OpenVZ and VirtualBox. It does not have any specific infrastructure requirements, fitting well into any pre-existing environment, storage, network, or user-management policies. OpenNebula software is tested to estimate scalability and robustness in large-scale VM deployments, and under stress conditions. An example of OpenNebula robustness is the CERN (European Organization for Nuclear Research) infrastructure prototype built by OpenNebula, which allowed for managing 480 servers and instantiated 16,000 VMs.[19, 20]

Figure 2.2 displays the various components of OpenNebula and the level different components operate. The lower components interact directly with the host's resources and the higher components interact with OpenNebula user interfaces. The drivers presented in the lowest level directly communicate with the underlying operating system.

These components are briefly described in the following list:

- Information drivers:
  These hypervisor-specific drivers retrieve the current status of virtual machines. Via SSH these drivers are copied and executed on all physical hosts.

- Virtual Machine drivers:
  These hypervisor-specific drivers manage the virtual machine on the current hosts.

- Transfer drivers:
  Transfer drivers manage virtual machine images through NFS, iSCSI, SSH.

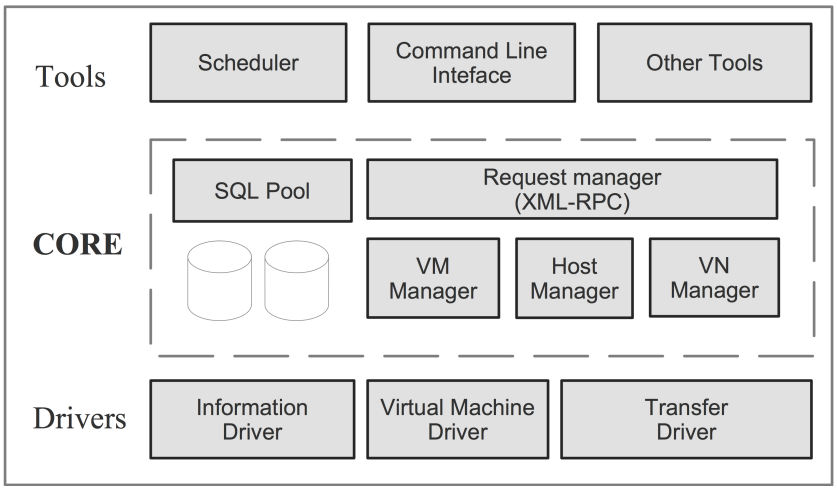

Figure 2.2 Different components in OpenNebula cloud platform

OpenNebula uses a simple SQLite database or a replicated MySQL database to store information retrieved from physical hosts and virtual machines, disk images and virtual networks.

The core of OpenNebula is written in highly optimized C++ code. OpenNebula native cloud API is available as Java, Ruby and XML-RCP API. OpenNebula secures communication between hosts through SSH RSA keypairs and Secure Socket Layer (SSL). Virtual networks are isolated with firewalls and “ebtables”. Some features of OpenNebula with respect to private cloud are presented in the following list[20]:

- User Management:
  Multiple users can be created who can only access their own instances. Users can be limited by quotas.

- VM Image Management:
  A centralized image catalog registers and manages every disk image.

- Virtual Network Management
  Multiple networks can be defined and bonded to different physical interfaces, with either static or dynamic IP addresses.

- Virtual Machine Management:
  Every virtual machine can be launched under every available hypervisor in the cloud.

- Service Management:
  A group of virtual machines can be grouped for being deployed together at boot time.

- Infrastructure Management:
  Physical hosts can be managed alone or grouped into independent clusters.

- Storage Management:
  Most common storage solutions like FibreChannel, iSCSI and **Network Attached Storage** (**NAS**) found in data centers are supported.

- Information Management:
  All hosts and VMs are monitored every few seconds. Integration with standard monitoring tools such as Ganglia is possible.

- User Interface:
  Every aspect of the OpenNebula can be managed vie OpenNebula command-line tools.

- Operation Center:
  The web interface provides most functionalities of the command line.

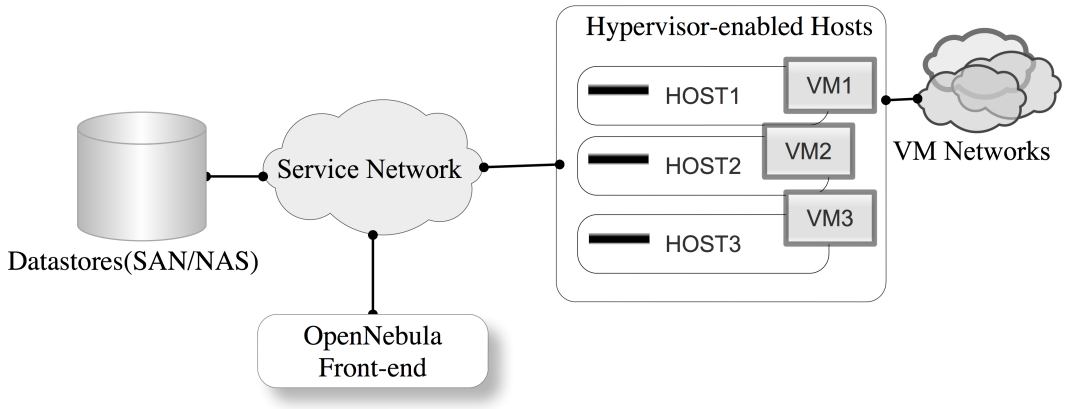

Figure 2.3 Different components of an OpenNebula cloud deployment

Figure 2.3 shows the main components of an OpenNebula cloud system. These components are briefly described as following:

- Front-end:
  OpenNebula services are run from the Front-end server.

- Hypervisor-enabled Hosts:
  These hosts require virtualization capabilities and a hypervisor to provide the resources needed by VMs.

- Datastores:
  The base VMs images are saved in these data repositories.

- Service Network:
  This physical network interconnects OpenNebula components.

- VM Networks:
  This physical network supports VLAN for the VMs.

2.4 CloudStack

CloudStack is a cloud computing software that enables creating, managing, and deploying Infrastructure-as-a-Service cloud services. This software is distributed freely under terms of the Apache License 2. CloudStack, apart from owning its own API, implements the S3 APIs, vCloud API, and provides support for CloudBridge Amazon EC2, which enables converting an Amazon API into a CloudStack API.

Originally Cloud.com developed CloudStack and released most of CloudStack as free software under the GNUv3 license in May 2010. One year later Citrix purchased Cloud.com and released the remaining code under GPLv3 license in August 2011. Citrix released CloudStack 3.0 in February 2012 and donated CloudStack to the Apache Software Foundation In April 2012. CloudStack online community provides timely technical support for free. Developers, users and contributors have access to weekly builds as well as the native source of the CloudStack project. [21, 22]

CloudStack is a console for managing data center computing resources. A number of well-known information-driven companies, such as Zynga, Nokia Research Center and Cloud Central, have deployed CloudStack to offer public cloud services, private cloud, or as part of a hybrid cloud.

CloudStack provides support for many popular hypervisors such as KVM, Xen, Xen Cloud Platform (XCP) and VMware. CloudStack API is compatible with AWS EC2 and S3.

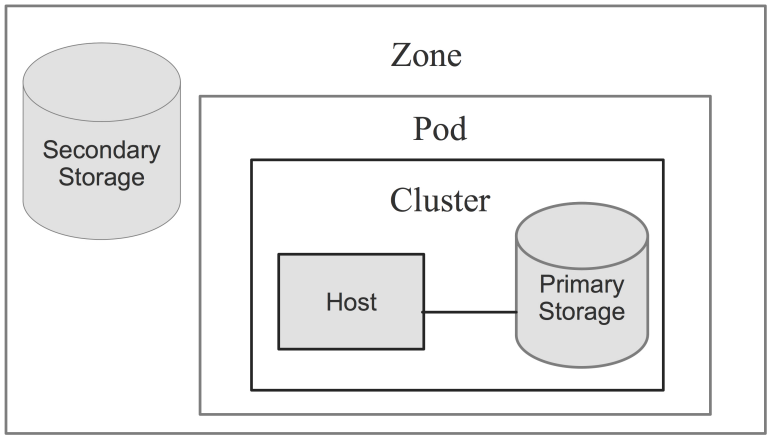

Figure 2.4 Overview of CloudStack infrastructure layers

To build a minimal CloudStack cloud the following components are required:

- Cloud Management Server:
  The Management Server is the CloudStack software. Cloud infrastructure is configured and managed through Management Server UI or API. The CloudStack Management Server runs in a Tomcat container and needs a MySQL database for persistence.

- Compute Nodes:
  The compute node is where hypervisor is installed. Virtual machines will be created on this node.

- Cloud Network:
  This network has to provide communication between CloudStack components through a Layer3 switch.

Figure 2.4 illustrates an overview of how Management Server organizes the cloud infrastructure. Each of these components is explained briefly in the following list [23-25]:

- Zone (Datacenter):
  A zone is an endpoint for accessing hosts. A zone consists of one or more pods and secondary storage.

- Pod:
  A pod is typically a dedicated rack of hardware that includes a layer-2 switch and one or more clusters.

- Cluster:
  A cluster consists of one or more hosts and primary storage.

- Host (Node):
  A host is a single compute node within a cluster. A host consists of one or more virtual machines (Instances).

- Instances (VMs):
  Each instance has an ID, IP address, etc.

- Primary Storage:
  This storage is associated with a cluster, and it stores the disk volumes for all VMs running on hosts in that cluster.

- Secondary Storage:
  This storage is associated with a zone. It stores templates, ISO images, and disk volume snapsshots.

2.5 Eucalyptus

Eucalyptus as an open source cloud platform was released for the first time in May 2008 (Eucalyptus 1.0). Eucalyptus stands for “Elastic Utility Computing Architecture,

Linking Your Programs To Useful Systems". This open source cloud platform is used to build private and hybrid clouds compatible with Amazon Web Services API including EC2, S3, EBS, and IAM. This compatibility enables virtual machines to be moved between a Eucalyptus private cloud and the Amazon public cloud, creating a hybrid cloud. Users are also able to manage Amazon or Eucalyptus instances with Eucalyptus commands.

The Eucalyptus cloud platform is mostly written in Java and few components in C and is published under GNU GPLv3 (only) with Proprietary relicensing. Eucalyptus provides support for KVM and VMware ESXi hypervisors. Eucalyptus is available via binary packages from several Linux distribution repositories or it can be built from source. At the time of writing this thesis Eucalyptus 3.3 had been released as the latest version of the software in April 2013. Eucalyptus cloud platform has a modular architecture with six distinct components [26-30]:

1. The cloud Controller (CLC):
   CLC is a web interface as well as EC2-compatible SOAP and Query interfaces, which is responsible for exposing and managing the underlying virtualized resources (servers, network, and storage). CLC is also used to manage high-level resource scheduling and system accounting as well as handling authentication, reporting, and quote management.

2. Walrus:
   Walrus provides persistent data storage to the virtual machines in a Eucalyptus cloud. It is interface compatible with Amazon's Simple Storage Service (S3).

3. The Cluster Controller (CC):
   CC gathers information about a set of Node Controllers and it schedules VM execution on specific Node Controller. The CC is also responsible to manage the VM networks.

4. The Storage Controller (SC):
   SC functions like AWS Elastic Block Store (EBS). SC communicates with the Cluster Controller and Node Controller. It is used to manage Eucalyptus block volumes and snapshots within a cluster. It can interface with various storage systems like NFS, iSCSI, and SAN.

5. The VMware Broker (Broker or VB):
   VB is an optional component, which enables Eucalyptus to deploy virtual machines on VMware infrastructure elements.

6. The Node Controller (NC):
   NC runs on all machines that host VM instances. The NC is used to control the execution, inspection, and termination of virtual machines. The NC also manages the virtual network endpoint. NC downloads and caches images from Walrus, and it also creates and caches instances as well.

2.6 Amazon EC2 and S3

Amazon Elastic Compute Cloud (Amazon EC2) is a main part of Amazon.com's cloud computing platform, also known as Amazon Web Services (AWS). Amazon EC2 was mostly developed in Cape Town, South Africa. A primary version of EC2 cloud service was released publicly on August 25, 2006 and the beta label was dropped on October 23, 2008. Amazon played a key role in the development of cloud computing by improving their data centers.[31-33]

Amazon EC2 allows users to rent virtual computers on which they can run their own applications. Through EC2 web service a virtual machine (an instance) can be booted from an Amazon Machine Image, which can contain any desired application software. Users pay for the time that servers are active. For latency optimization and high levels of redundancy EC2 services are available from different geographical locations. EC2 uses Xen as underlying virtualization technology. As explained before, in early 2008 Eucalyptus was the first available open source AWS API-compatible cloud platform.[33]

Amazon S3 (Simple Storage Service) is another web service provided by Amazon Web Services. Amazon S3 provides storage through web services interfaces such as REST, SOAP and BitTorrent. It was publicly available, in the United States in March 2006 and then in Europe in November 2007. S3 users can store and retrieve any amount any time from anywhere through a simple web services interface.[34, 35]

S3-based storage costs per gigabyte per month. Applications can access S3 through an API. For example, the Apache Hadoop supports a special filesystem to support reading from and writing to S3 storage during a MapReduce job. There is a S3 filesystem for Linux, which can mount a remote S3 filestore on an EC2 image, as if it were local storage.[34]

2.8 Related work

This section provides an overview of previous studies related to evaluating open source cloud platforms. It should be noted that many of these works are more than one year old and thus the information they provide may not be of that much of use.

A webpage published on Wikipedia has collected basic general information of some open source and proprietary cloud platforms for purpose of comparison. This information includes the programming language, initial release date, supported operating systems and supported hypervisors. The information provided in the given comparison is too basic to evaluate open source cloud platforms. [36]

Another private cloud evaluation published by Lance Albertson [37], compares four open source private cloud platforms OpenStack, CloudStack, Eucalyptus and Ganeti. The results in this evaluation are in straight form of yes no and it does contain a detailed discussion about the results. This work compares several aspects of the open source cloud platforms listed as bellow:

- Storage

- Virtual disk image
- Interface and network
- Ease of installation
- Amount of upfront configuration needed for a base install
- Ease of initialization of a cluster
- Strengths / Weaknesses

As it can be seen, this cloud evaluation does not cover other open source cloud platforms such as OpenNebula, Cloud Foundry, Nimbus, OpenShift, OVirt and OpenQRM. [37-42].

A comparative study conducted by Mahjoub and coworkers [43] compares OpenNebula, Nimbus, Eucalyptus, Xen Cloud Platform, AbiCloud and OpenStack.
The survey aims to help users to choose the better cloud platform with the most suitable open source virtualization technologies. In this work the following aspects of the cloud are compared:

- Main purpose
- Architecture
- Virtual Machine Placement
- Storage
- Network
- Access interface
- Security
- Fault-tolerance
- Load balancing

Table 2.1 [43] displays part of the results presented in this survey related to OpenNebula and OpenStack. As can be seen, Table 2.1 compares only some general information and some basic features of the software.

	OpenStack	**OpenNebula**
Produced by	Rackspace, NASA, Dell, Citrix, Cisco, Canonical etc.	European Union
Main purpose	Offers Cloud Computing services	Build private Cloud
Users	Enterprises, service providers and researchers	Researchers on Cloud Computing and Virtualization
Supported OS	- Linux - Windows - Requires x86 Server	Linux (Ubuntu, RedHat Enterprise Linux, Fedora et SUSE Linux Enterprise Server)
Architecture	Integration of OpenStack object and OpenStack compute	- Centralized - Three components - Minimum two servers
Language	Python	Java, Ruby and C++
Storage	OpenStack Store	- SCP - SQLite3
Network	OpenStack Compute	Manual configuration
Access interface	Web interface Web	- EC2 WS API - OCCI API
User	- Certification	- Authentication
Administrator	- Certification	Root (Only if necessary)

Load balancing	The cloud controller	Nginx
Fault tolerance	Replication	Database backend (registers virtual machine information)
Live migration		Shared FS
VMs location	OpenStack Compute	Cluster node
Compatibility with EC2	No	Yes
Used by		Reservoir Project, NUBA

Table 2.1Comparison of OpenNebula and OpenStack cloud platforms

Sempolinski, P. and Thain, D. [44] carried out a study that compares Eucalyptus, OpenNebula and Nimbus. This study describes some of the features of cloud platforms and analyzes the primary differences in the overarching structure and guiding philosophy of these open source cloud platforms. It aims to examine some aspects of these cloud platforms that do not change as rapidly as the software features such as the structure, centralization and customizability level of the software.

Another open source cloud platforms comparison conducted by Von Laszewski, G. and his coworkers [45] in Indiana University, compares more qualitative features of Nimbus, Eucalyptus, OpenStack, OpenNebula.

Table 2.2 [45] presents parts of the feature comparison of OpenNebula and OpenStack from the study discussed in the previous section. The checkmarks (✔) indicate a positive evaluation.

	OpenStack	**OpenNebula**
Interfaces	EC2 and S3, Rest Interface. Working on OCCI ✔✔	Native XML/RPC, EC2 and S3, OCCI, Rest Interface ✔✔✔
Hypervisor	KVM, XEN, VMware Vsphere, LXC, UML and MS HyperV ✔✔✔	KVM, XEN and VMWare ✔✔
Networking	- Two modes: (a) Flat networking (b) VLAN networking -Creates Bridges automatically -Uses IP forwarding for public IP -VMs only have private IPs ✔✔✔	- Networks can be defined to support Ebtable, Open vSwitch and 802.1Q tagging -Bridges must exists in the compute nodes -IP are setup inside VM ✔✔✔
Software deployment	- Software is composed by component that can be placed in different machines. - Compute nodes need to install OpenStack software ✔	Software is installed in frontend ✔✔✔
DevOps deployment	Chef, Crowbar, Puppet ✔✔✔	Chef, Puppet ✔✔
Storage (Image Transference)	- Swift (http/s) - Unix filesystem (ssh) ✔	Unix Filesystem (ssh, shared filesystem or LVM with CoW) ✔
Authentication	X509 credentials, LDAP ✔✔✔	X509 credential, ssh rsa keypair, password, LDAP ✔✔✔
License	OpenSource – Apache	OpenSource Apache

	✔	✔

Table 2.2 Comparison of OpenNebula and OpenStack cloud platforms

Deploying all of these cloud platforms, they suggested that OpenNebula and Nimbus were easier to install. Nimbus was found very reliable. OpenStack and Eucalyptus Clouds were very resource consuming.

Stefan Wind [46] has carried out a study evaluating Eucalyptus, OpenNebula, Abi-Cloud and Nimbus. His study explained the differences of cloud platforms and provided specific recommendations for using the platforms.

Cordeiro, T. and his coworkers [47] presented a comparative description about three open source cloud platforms; Xen Cloud Platform, Eucalyptus and OpenNebula. The work discussed differences of cloud platforms and described illustrative examples of use.

One of the most recent related studies to this thesis was carried out by Xiaolong and his coworkers [48], and compared OpenStack and OpenNebula. This work discusses differences of OpenStack and OpenNebula from provenance, architecture, hypervisors, security and community activity. It should, however, be noted that this study is only based on theoretical knowledge and implementing the cloud platforms is considered future work. This is the main distinction between the evaluation of cloud platforms presented in Xiaolong's study and the evaluation of cloud platforms presented in this thesis.

One disadvantage with this study is the logic behind some of the arguments concerning evaluation of the cloud platform. For example, in the last part of section Xiaolong's study it is written "OpenStack is straightforward because the guide documentations can be seen on the website, so it is possible to build a cloud even you have no any specialist personnel"[*sic*]. [48] The problem with this argument is that the guide documentation can be seen on OpenNebula websites as well as other cloud platforms.

In the field of already existing work relevant to this thesis, the excellent research carried out by Qingye Jiang (John)[49], a developer at Eucalyptus Systems Inc, is of importance. Since the beginning of 2009, Jiang has run an experiment to compare the user- and developer communities of OpenStack, OpenNebula, Eucalyptus and CloudStack cloud platforms.[49-51] His research is based on the communication between community members in the form of mailing lists or public forum discussions. He has developed a Java program that retrieves all the forum posts and mailing list messages into a MySQL database for further processing and analysis. This experiment provides scientific and up-to-date data to evaluate cloud platform communities. The work of Qingye Jiang has had a paramount importance in this thesis. Any use of his experimental figures and analytical information has been made with the permission of Mr Jiang.

Chapter 3

3. Methodology

This chapter provides the guidelines that will describe the approach toward solving the problem-statement of this thesis. Following this are descriptions of the hardware, software, evaluation criteria and alternatives to the approach presented in this thesis.

3.1 Hardware and Software

Table 3.1 presents the hardware specifications of three Dell PowerEdge server machines that are used to build the private cloud infrastructures presented in this thesis.

Server 1	
Product:	Dell PowerEdge 2950- 64 bits
CPU	Intel(R) Xeon(R) CPU 2.33GHz- x86-64
Cores	4
System Memory	4 GiB
Hard-disks	6 × DELL 749GB (as two hard disks first one 1 × 749GB and second as 5× DELL 749GB in RAID 0)
Hardware RAID	LSI Logic/Symbios Logic SAS1078
Ethernet interface	2 × NetXtreme II BCM5708 Gigabit Ethernet 1Gbit/s w: 64 bits
Server 2	
Product:	Dell PowerEdge 2850- 64 bits
CPU	4 × Intel(R) Xeon(TM) CPU 3.00GHz - x86-64
Cores	4 ×1 = 4
System Memory	2 GiB
Hardware RAID	PowerEdge Expandable RAID controller 4
Hard-disks	2 × 73 GB SICI RAID 0
Ethernet interface	2 × Intel 82541GI Gigabit Ethernet Controller 1Gbit/s w: 32 bits
Server 3	
Product:	Dell PowerEdge 2850- 64 bits

CPU	4 × Intel(R) Xeon(TM) CPU 3.00GHz - x86-64 1core
Cores	4 ×1 = 4
System Memory	1 GiB
Hardware RAID	PowerEdge Expandable RAID controller 4
Hard-disks	2 × SICI 36 GB in RAID 0
Ethernet interface	2 × Intel 82571GI Gigabit Ethernet Controller 1Gbit/s w: 32 bits 2 × Intel 82541GI Gigabit Ethernet Controller 1Gbit/s w: 32 bits

Table 3.1 Hardware specifications

The parallel evolution of the Ubuntu Operating system with cloud computing makes the Ubuntu operating system a good choice onto which build cloud infrastructures. Both the latest version of Ubuntu 12.10 server 64x and Long Term Support version Ubuntu 12.04 server 64x (LTS) are used on servers to host the cloud infrastructures designed for the experimental purposes of this thesis. The reason for using both of these versions is that different cloud deployment instruction manuals are based on these different Ubuntu versions.[52]

All the figures in this thesis are produced by ConceptDraw PRO drawing tools.

3.2 Evaluation criteria

Evaluating a cloud platform is a multifaceted challenge. It is important to distinguish irrelevant factors from truly crucial ones. Generally, open source software has generic characteristics and software-specific characteristics. The generic characteristics are common among all software, e.g. functionality, cost, support, maintenance, usability, flexibility/customizability, interoperability, license, performance and security issues.[53] Software-specific characteristics present features and services delivered by the software. Some software specific characteristics related to a cloud platform are Storage, Virtualization, Network and Management[54-56].

The more characteristics investigated and considered in software evaluation, the more reliable the evaluation will be. Some of the factors that challenge software evaluation are listed as following:

- Investigating the rights criteria.
- Right methods and measurement techniques to investigate the criteria.
- Amount of time that investigating criteria may require.
- Amount of time that can be dedicated to the evaluation.
- Right hardware resources.
- Amount of hardware resource.

Considering all factors mentioned above and based on some software evaluation reviews and suggestions, the following software attributes are investigated in this thesis. [53-60]

3.2.1 Deployment

In the process of installing the cloud platform, several aspects of the deployment process can be checked.

1. How many packages are installed through the cloud deployment process?
2. How many of the total packages are prerequisite packages?
3. What are the prerequisite packages?
4. How many files are configured?
5. How many lines are configured in total?
6. How long does the total deployment process take?
7. How long does the partial automatic cloud deployment process take?
8. How many lines of code does the automation script contain?
9. How automatable is the process of deployment when using scripts?

    - Between 100% and 75% of the process can be provisioned via script
    - Between 75% and 50% of the process can be provisioned via script.
    - Between 50% and 25%of the process can be provisioned via script.
    - Less than 25% of the process can be provisioned via script.

10. Did the cloud deployment behave as expected?
11. What is the debugging difficulty level?

    - **Easy**: Installation guide provided the guidelines to fix the issue.
    - **Medium**: Documentation, log files, and searching the web fixed the issue.
    - **Hard**: Documentation, log files, searching the web and communicating with software community fixed the issue.
    - **Very Hard**: Documentation, log files, searching the web, communicating with software community and consulting with some cloud expert fixed the issue.

12. What level of expertise the platform deployment manuals are written for?

    - **Medium**: requires basic knowledge of command line, installing packages and networking concepts.
    - **High**: Requires high knowledge of managing operating system and networking, databases and debugging abilities.
    - **Very High**: Requires expert understanding of operating systems and networking and other underlying technologies such as Virtualization.

Regarding some of the questions presented in the list above the following points should be considered:

- The time taken by the deployment process will be measured approximately excluding the time used to prepare the operating system and network connectivity. Naturally this time will depend on the skills of the system administrator as well as repetition of the deployment.
- The time taken by the partial deployment process is an approximate time amount including the time to run the scripts and the manual deployment parts.

- It is difficult to estimate an exact level of automation of the deployment process. Therefore the automatability level is defined as an intervall. Additionally, an approximate actual percentage will be given.

3.2.2 Support

1. Which operating systems does the cloud platform provide support for?
2. How many deployment instructions are available?
3. How many books are published for the software?
4. Is commercial support available?

3.2.3 Usability

5. How many user interfaces does the cloud platform provide?

3.2.4 Compatibility

6. Is the cloud platform compatible with similar technologies?

3.2.5 License

7. Under what license is the cloud platform published?

3.2.6 Platform Complexity

1. How many lines of code are written for the cloud platform software?
2. How many components does the cloud platform have?
3. How many different programing languages are used in writing the cloud platform software?

3.2.7 User and Developer Community

1. What is the monthly number of topics (threads) that are being discussed through the user community?
2. What is the monthly number of posts (messages) that are being discussed through the user community?
3. What is the monthly participation rate of the online community?
4. What is the monthly numbers of people who have participated in forum or mailing list discussions (number of monthly active participants)?
5. What is the accumulated community population (total number of users and developers who have participated in forum or mailing list discussions)?
6. How many percent of monthly participants are new members?
7. What is the monthly number of commit operations committed to the source code of the cloud platform?
8. What is the monthly number of unique contributors (identified by unique github.com accounts) for the project?
9. What is the monthly number of unique institutes that contribute to the development of the cloud platform?
10. Which institutions contribute to the source project?

11. How often is there a new release of the cloud platform?
12. When was the last stable version of the software released (History of stable releases)?
13. If a message is posted to the user community of the cloud platform, how often was the message replied to?
14. Was the answer helpful in solving the problem?
15. How long did it take to get a reply?

3.3 An Alternative Methodology

Previous section in this chapter described the approach used in this thesis. One alternative to the methodology used to evaluate open source cloud software could be an evaluation based on a survey, which would ask about the user's experience or the issues that users have faced using the software. This survey could be posted on the user community of the cloud software, in order for the users of the cloud platform to complete. Additionally, the survey could be given to already known users of the software outside the user community.

Chapter 4

4. Results

This chapter provides details of how a private cloud was designed and set up using OpenStack and OpenNebula cloud software. Some tables represent the result at the end of each experiment. The last section in this chapter provides the results from investigating the user- and developer-community for OpenStack, OpenNebula, CloudStack and Eucalyptus.

4.1 OpenStack

There are several OpenStack deployment guides referenced on the OpenStack web site. [61] Among these OpenStack deployment guides the following three deployment guides are used in this thesis [61]:

- OpenStack Folsom Tow Node Setup on Ubuntu 12.10
- OpenStack Folsom Three Node Setup on Ubuntu 12.10
- OpenStack Folsom Three Node Setup on Ubuntu 12.04 (LTS)

The OpenStack installation and deployment guides are mainly developed for Ubuntu, Red Hat Enterprise, CentOS, and Fedora Linux operating systems. There are two versions of documentation for Ubuntu, a basic installation guide for Ubuntu 12.04, which is a document of approximately 20 pages, and the main installation guide for Ubuntu/Red Hat, consisting of 141 pages. Both manuals focus on setting up OpenStack using Ubuntu server 12.04(TLS). The basic version helps to set up OpenStack Folsom for development purposes (using the Ubuntu cloud Archive). It follows a three-node setup architecture with one Controller, one Network and one Compute node. More Compute nodes can be added later. This document aims to be a good start for beginners in OpenStack who want to install a testing infrastructure.

The following open-stack deployment guides are provided for Ubuntu 12.10:

- OpenStack Folsom Deployment Guide for Ubuntu 12.10 [62]
- Basic OpenStack Folsom Deployment Guide (Nimbula) [63]

4.1.1 Basic OpenStack Folsom Two Node Setup on Ubuntu 12.10

This experiment aims to setup a private cloud by deploying OpenStack on two nodes running Ubuntu 12.10. Figure 4.1 shows the topology of the cloud infrastructure designed and deployed for this OpenStack deployment experiment. One node will be the Controller node for managing the cloud infrastructure and the other node will be the Compute node with the KVM hypervisor.

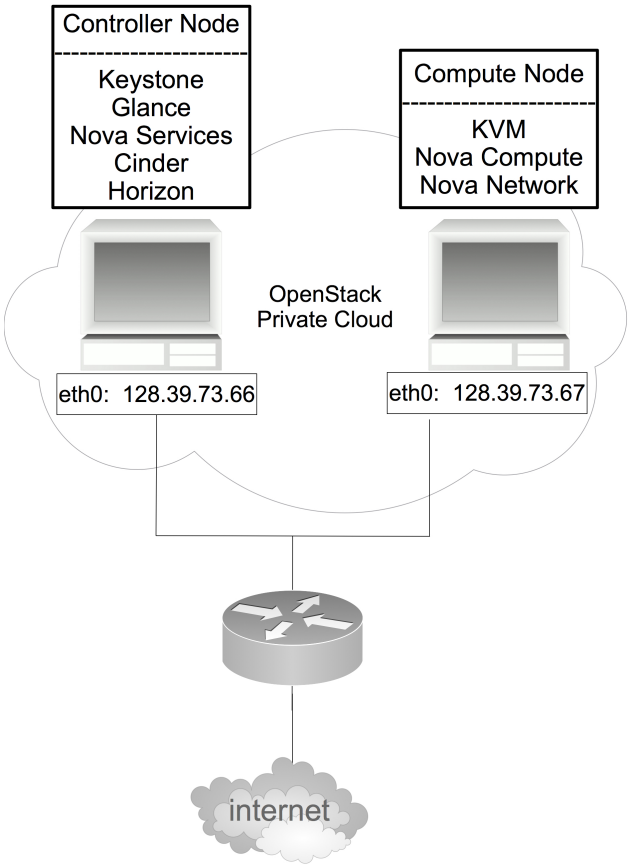

Figure 4.1 Topology of OpenStack Folsom two-node deployment

4.1.1.1 Setting up the Controller Node

The first step in setting up the Controller node is updating and upgrading the fresh installation of Ubuntu 12.10. Note that all the configurations require super-user mode.

All nodes of an OpenStack infrastructure require synchronization with each other. To achieve this synchronicity an NTP Server is installed on all nodes, where the entirety of nodes will synchronize with the NTP server on the Controller node.

Next the MySQL server is installed and configured to accept all incoming requests. This is followed by the creation of databases and database users for various OpenStack components.

In the Background chapter Figure 2.1 demonstrated the OpenStack architecture, where a queue server provided the communication between all the OpenStack components. OpenStack uses AMQP (Advanced Message Queuing Protocol), which is an open standard that passes business messages between applications.

The AMQP broker can be either RabbitMQ or Apache Qpid. The RabbitMQ server is open-source message broker software that implements the AMQP standard. Apache Qpid is another open source message broker, which implements the AMQP standard and allows programs to communicate by exchanging messages. OpenStack manuals use the RabbitMQ server, thus RabbitMQ server is installed on the Controller node.

To setup up networks for a hosted virtual machine and enable VLANs on NICs, installing vlan and bridge-utils packages are followed as the next steps. IP_Forwarding needs to be enabled in a sysctl.conf file permanently.

The first OpenStack component installed on the Controller node is Keystone the OpenStack identity service. Note that a Keystone database and a Keystone database user are created in earlier steps.
After Keystone is installed it has to be configured to be able to access the Keystone database. After configuring the Keystone we restart the Keystone service in order for the identity service to synchronize with Keystone database.
The keystone database needs to be filled with data required to identify the OpenStack components. This is done through executing two scripts created by developers of OpenStack team. These scripts named *keystone_basic.sh* and *keystone_endpoints_basic.sh* are downloaded and set to be executable. Before executing these scripts some small modifications need to be done in these files. In the *keystone basic.sh* script, the $HOST_IP variable needs to be set to Controller node's public IP address. In the *keystone_endpoints_basic.sh* script, the $HOST_IP, $EXT_HOST_IP, & $MYSQL_HOST variables need to be modified accordingly. Last but not least we run *keystone_basic.sh* followed by *keystone_endpoints_basic.sh*. Notice running *keystone basic.sh script produces* no output while executing *keystone endpoints basic.sh* produces output with some tables presenting the data inserted into the Keystone database.

The next step is creating and assigning OpenStack identification variables. For simplicity reasons these variables are placed in a file and sourced to load into the system.

After installing Keystone, setting up OpenStack continues with the installation of the image storage service called Glance. Again notice that a Glance database with a corresponding data base user with proper access-right are created. Glance is configured by updating *glance-api-paste.ini* and *glance-registry-paste.ini* files with authentication data. The *glance-api.conf* is updated with the right database information and flavor type. Then Glance-api and Glance-registry services require a restart. Finally, glance service need to synchronize with the glance database followed by restarting the services again to take into account the new modifications.

To verify glance's successful installation, a new image is added to the store. If all has gone well so far, an output with the newly created image ID is printed out. Running the following command will display the newly added image in the list of images existing in the image store.

```
> glance image-list
```

Networking on the Controller node needs to be configured to contain a bridge called *br100*. The network configuration for the Controller node is presented in appendix A.5.

The scripts that were run earlier assume that Quantum will be used instead of nova-networks. Additionally, having both endpoints on the same installation can cause some serious conflicts. To prevent this from happening, Quantum endpoint and service need to be removed.

Nova is subsequently installed and related files are configured. A Nova database and database user are created. Configuring Nova requires modifying the authtoken *api-paste.ini*. The main Nova configuration file is *nova.conf*. The content of this file is presented in appendix A.4. When Nova is set up and configured, it requires synchronizing with the Nova database. Finally restart all of Nova services to take new changes in to account.

Cinder the OpenStack volume manager is the next service to be installed and configured. Cinder is used to manage volumes for virtual machines. After installing Cinder packages, authentication data is provided the Cinder in */etc/cinder/api-paste.ini /etc/cinder/cinder.conf* files. Notice Cinder also need to have a database, therefore a Cinder database and Cinder database user are created earlier. Cinder has to synchronize with its database.
Synchronize the database. Cinder is also has to be provided with some volume to use it in its services. So a physical volume under /dev/loop2 is created followed by a volume group named cinder-volumes is created and made persist after a server reboot.

The last OpenStack component to be installed on Controller node is Horizon, which provides the web interface to control the cloud.

Some bugs have been reported using the default Ubuntu theme by Horizon. This theme is be disabled in */etc/openstack-dashboard/local_settings.py* file.

Finally a serve reboot may be required in order to make Horizon to authenticate properly. If rebooted Nova services need to be run again.

4.1.1.2 Setup Compute Node

This section will cover how to add a Compute node to the OpenStack cloud Infrastructure. In the case of having several nodes as Compute node, the same configuration applies.
As super user the freshly installed Ubuntu server 12.10 is updated and upgraded.

As discussed earlier in the Controller node setup section, NTP server is required to keep all nodes synchronized. Thus NTP server is installed and configured to keep the Compute node synchronized with the Controller node.

To provide virtual VLANs and bridges vlan and bridge-utils packages are installed. Also IP forwarding is enabled permanently.

Notice is essential that Compute node's hardware provides support for virtualization. Making sure about virtualization support KVM hypervisor and it's dependency packages are installed and */etc/libvirt/qemu.conf* file is configured.

System default virtual bridges need to be deleted. Also Live migration is to be enabled by updating value of some variables in */etc/libvirt/libvirtd.conf*, */etc/init/libvirt-bin.conf* and */etc/default/libvirt-bin* files. Changes are applied by restarting the libvirt service.

Then nova-network package is installed to provide networking features for Cloud.

Network interface on Compute node requires to play as a bridge. The Network configuration on Compute node is presented in appendix A.6. Notice this configuration is identical to configuration on the Controller node except the IP address. Then bridge *br100* is added followed by restarting the networking services.

Now Nova-api and Nova-compute services are installed and configured. Configuration applies to updating authtoken section in */etc/nova/api-paste.ini* file. The main Nova configuration file *nova.conf* is configured as presented in appendix A.4.

After configuring Nova it has to synchronize with the Nova database on the Controller node. Services are restarted to apply changes.

At this point listing Nova services on the Compute node via command line should list all running services with smiley icon ':)' pointing out that setting up the Compute node is successfully completed.

Now OpenStack Dashboard (web-interface) Horizon can be accessed form URL pointing to IP of the Controller node http://128.39.73.66/horizon. The following credentials are the login credentials:

username: admin
password: admin_pass

After logging in, a new project needs to be created. Note that admin has to be added as project member. The ID of a newly created project needs to be copied. This ID is used to bind a network with this project via command line on the Controller node.

Running the following command on the Controller node will create a network (10.33.14.0/24) and bind it with new project. Notice the value of fixed_range option is same as the value mentioned nova.conf in both Controller node and Compute node.

```
> nova-manage network create --label=NimbulaNetwork --
fixed_range_v4=10.33.14.0/24 --bridge=br100 --project_id=<NEWLY CREATED
PROJECT ID> --num_networks=1 --multi_host=T
```

At this point the setting up an OpenStack private cloud is completed.

4.1.1.3 Observations and Outcomes

Table 4.1 provides the outcomes of the experiment described above related to the questions asked about the deployment process in the Methodology chapter.

1. Total number of packages installed:	24 Packages installed on Controller node 10 Packages installed on Compute node **34 Packages installed totally**
2. Total number of prerequisite packages:	**19 Packages**
3. Name of prerequisite packages:	1. mysql-server 2. python-mysqldb 3. rabbitmq-server 4. 3 × ntp server 5. vlan 6. bridge-utils 7. curl 8. openssl 9. iscsitarget 10. iscsitarget-dkms 11. memcached 12. vlan 13. bridge-utils 14. cpu-checker 15. kvm 16. libvirt-bin 17. pm-utils
4. Total number of files configured:	17 Files on Controller node 10 Files on Compute node **27 Files configured totally**
5. Total number of lines configured:	**228 Lines**
6. Total time taken by deployment:	**7 Hours**
7. Total time taken by partial automatic deployment:	**30 Minutes**
8.Total number of lines included in automation script:	**311 Lines**
9. Automatability level of deployment:	Between 75% and 100% **Actually 90%**
10. Deployment ended with expected behavior:	**No**
11. Debugging level:	**Very Hard**

12. Level of expertise required by deployment	**Very High**

Table 4.1 Results of OpenStack two-node deployment

Regarding the process of the OpenStack deployment described in this subsection the following issues were encountered:

- Despite of the fact that most of deployment process went well as expected by the deployment guide, no VM instances could be created, and in addition the system could be debugged. This was due to the fact that the log files were to complicated to track the errors. This experiment was repeated several times from scratch to be sure that all went according to the description in the deployment guide. Nevertheless, the system ended in the same error status each time.
- Most of the deployment process could be automated by scripts. Only network configurations were not included in the deployment process.

- Three unclear issues were identified in the deployment guide:
  1. The section related to "Install and configure Keystone", provides two links for downloading the scripts that were used to fill the Keystone database; one of the links provides the scripts with Quantum and the other link provides the scripts the name scripts without Quantum.
  2. The networking configurations are very unclear.
  3. The main error rises in the last part of the deployment where creating a network fails. Several days trying to solve this issue did not help.

4.1.2 OpenStack Folsom Three Node Setup on Ubuntu 12.10

The previous section covered a two-node OpenStack basic setup process on Ubuntu 12.10 without Quantum component. This section will present more advanced multi node OpenStack setup with Quantum as network manager for cloud.

Figure 4.2 provides an overview of the topology of this OpenStack deployment. As can be seen from Figure 4.2, there are three nodes that provide the cloud infrastructure. The Controller node and the Compute node are more or less similar to the Controller node and the Compute node in the previous setup model, except that in the previous model a Nova network was used as cloud network manager. So the new node in this model is a Network node, which will be the network manager for the cloud infrastructure to provide networking features for Cloud.

An OpenStack standard setup with Quantum requires four distinct physical networks. These networks are listed below:

- Management network

Provides internal communication between OpenStack components. Since it is used for internal communication within the data center it is wise to use private IP addresses.

- External network

Provides Internet access for VMs, hence the IP addresses need to be public so they can be reachable on the Internet.

- Data network

Provides data communication within the cloud deployment for VMs. The IP addresses used on this network depend on the Quantum plugin in use.

- API network

Exposes all OpenStack APIs to tenants. IP addresses used on this network should be public addresses to be reachable on the Internet. The same network used as external network can be used here as well.

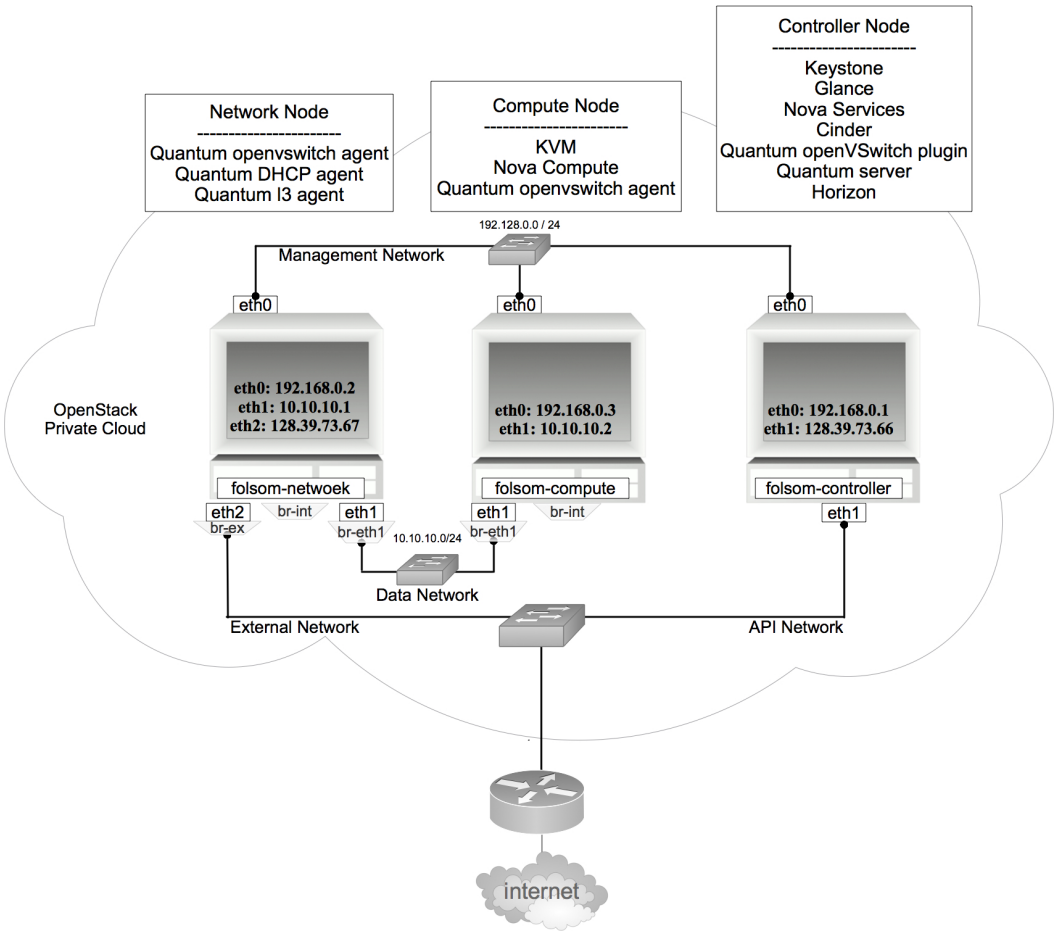

Figure 4.2 Topology of OpenStack Folsom three-node deployment

4.1.2.1 Requirement

To setup this model requires several network interfaces. As displayed in Figure 4.2, both the Controller node and the Compute node require two network interfaces, while the Network node requires three network interfaces.

4.1.2.2 Setting up the Controller Node

Setting up the Controller node begins with updating and upgrading the freshly installed Ubuntu 12.10 64bits server. Network configuration for the Controller node is presented in appendix B.6.

As mentioned in the previous OpenStack deployment, all nodes in the cloud need to be synchronized. This is achieved by installing NTP server. After installing NTP server, it is configured to synchronize with Ubuntu's NTP server. Of course any other NTP server can replace it. After configuring the NTP server we restart the NTP service to update it with new configurations.

Next the MySQL server is installed and configured to accept all incoming requests. Then databases for various OpenStack components are created as keystone, glance, quantum, nova and cinder databases. In addition, for each database a corresponding database user with proper access- rights is created.

Installing RabbitMQ server provides the queue server for the cloud infrastructure. The corresponding package is installed.

Vlan and bridge-utils packages are installed to provide for the setup up networks for a hosted virtual machine and to enable VLANs on NICs. IP_Forwarding is permanently enabled in sysctl.conf file.

Now the basis for building the OpenStack cloud is ready, and subsequently Keystone the OpenStack identity manager is installed via keystone package. Keystone configuration provides the Keystone with database URL containing the database address database, database credentials in */etc/keystone/keystone.conf* file. After the configuration the Keystone service is restarted and synchronized with its database.

Similar to previous OpenStack deployment the Keystone database is filled with the necessary data by two running the scripts *keystone_basic.sh* and *keystone endpoints basic.sh* scripts, which are download from a link provided on the deployment guide. Before executing the scripts $HOST_IP and $HOST_IP_EXT variables should be updated to match with corresponding IP addresses used in this deployment. Then the *keystone_basic.sh* is run, followed by *keystone_endpoints_basic.sh*.

A file containing some variables corresponding to an OpenStack tenant is created and loaded into the system. These system variables define the access credentials to enable an administration account for administrating OpenStack Cloud. These variables are as following:
OS_TENANT_NAME=admin
OS_USERNAME=admin
OS_PASSWORD=admin_pass
OS_AUTH_URL="http://128.39.73.66:5000/v2.0/"

After this, Glance the OpenStack image store is installed and configured. Configuring Glance is to update the authtoken in */etc/glance/glance-api-paste.ini* and */etc/glance/glance-registry-paste.ini* files with necessary data for Glance to be able to

authenticate against Keystone service. Also the sql connection variable and the flavor variable need to be updated in */etc/glance/glance-api.conf* and */etc/glance/glance-registry.conf* files. To make these changes take effect, Glance-api and Glance-registry services are restarted and finally the Glance service is synchronized with the Glance database on the Controller node. At this point, the Glance service is ready and for test purpose an operating system image is downloaded and added to the Glance image store. Listing the Glance images, the newly added image should appear in output in the list of Glance images.

Preparing the Controller node continues by installing and configuring Quantum server and Quantum openVSwitch plugin. Notice that the quantum database and database user were created earlier. Configuring Quantum is to update sql_connection variable as well as the OVS plugin in */etc/quantum/plugins/openvswitch/ovs quantum plugin.ini* file. The authtoken in */etc/quantum/api-paste.ini* file is updated aswell. Finally, restart the Quantum server.

Then Nova, the main OpenStack component, is installed. This component contains six separate packages. To configure Nova the authtoken section in the */etc/nova/api-paste.ini* file is updated with the required data. The main configuration file */etc/nova/nova.conf* file is presented in appendix B.4. Nova is synchronized with nova database and finally Nova services are restarted. A successful Nova installation can be confirmed by following command:

```
> nova service list
```

Cinder, the OpenStack volume manager, is the next component to be installed. The corresponding Cinder packages are installed, which contain three packages related to Cinder and three packages related ISCSI (Internet Small Computer System Interface) services used for linking data storage facilities. After installing these packages, update all false values to true in */etc/default/iscsitarget* file and then iscsitarget and open-iscsi services are started. The authtoken in */etc/cinder/api-paste.ini* file is updated. The configuration for */etc/cinder/cinder.conf* is presented in appendix B.7. Finally when all configurations are done cinder needs to synchronize with cinder database.

In the same way as the previous setup, cinder requires a physical volume and a volume group under. So the physical /dev/loop2 is formatted and used instead of a second hard disk. Then a volume group named cinder-volumes is created on top of /dev/loop2.
How to create these volumes is presented in appendix B.8. Make sure the volume is not lost on a server reboot.
Finally we restart cinder-api and cinder-volume services to get updated with new configurations.

The last OpenStack component on the Controller node is Horizon. The corresponding package is installed in addition to memcached package. Optionally OpenStack Ubuntu theme could be removed. Finally apache web service and memcached service are restarted.

At this point setting up Controller node is completed and OpenStack Dashboard can be accessed from http://128.39.73.66/horizon. Log-in credentials are as following. Notice that these credentials are the same as the variables loaded into the system ear-

lier.

```
User: admin
Password: admin_pass
```

4.1.2.3 Setup Network Node

As super user we update and upgrade the freshly installed Ubuntu server 12.10 64bits server.
As Figure 4.2 displays, the Network node has three network interfaces. Make sure our network configuration is configured as expected. The network configuration for Network node is presented in appendix B.9.

NTP server is installed and configured to keep Network node synchronized with Controller node.

Then Vlan and bridge-utils packages are installed. IP_Forwarding is enabled in sysctl.conf file permanently.

OpenVSwitch is installed on the Network node to forward traffic between different VM within the cloud and traffic between a VM and a physical network. So the corresponding packages for openVSwitch are installed. No configuration is needed for this step.

As Figure 4.2 displays, the Network node requires three network bridges; br-int for VM integration, br-eth1 for VM configuration and br-ex is used to make VM accessible from the internet. These bridges are created and br-eth1 is attached to eth1 NIC and br-ex is attached to eth2 NIC.

Now Quantum openVSwitch agent, l3 agent and dhcp agent are installed. These are the main packages required by the Network node. Configuring Quantum occurs by update the authtoken in */etc/quantum/api-paste.ini* file to be able to authenticate against Keystone on Controller node.

Then in */etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini* file following variables are updated:

```
sql_connection = mysql://quantumUser:quantumPass@128.39.73.66/quantum
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

The following variables need updating in */etc/quantum/l3_agent.ini* file:

```
auth_url = http://128.39.73.66:35357/v2.0
auth_region = RegionOne
admin_tenant_name = service
admin_user = quantum
admin_password = service_pass
metadata_ip = 128.39.73.66
metadata_port = 8775
```

In */etc/quantum/quantum.conf* file rabbit_host variable is set to the Controller node IP.

Finally restart all Quantum services for configuration to take place.
At this point setting up the Network node is completed. Building up cloud infrastructure will continue with setting up a Compute node

4.1.2.4 Setting up a Compute Node

Setting up a Compute node starts with updating and upgrading a fresh Ubuntu 12.10 Server x64. The network configuration for the Network node is presented in appendix B.9.

To keep the Compute node synchronized with the Controller node, NTP server is installed and configured.

Similarly to previous nodes, Vlan and bridge-utils are installed. In addition, IP_Forwarding is also permanently enabled in sysctl.conf file.

To check if hardware on this node supports virtualization, cpu-checker package can be used. after checking virtualization support KVM hypervisor and its dependency, packages are installed.

Configuring hypervisor the *cgroup_device_acl* array in the */etc/libvirt/qemu.conf* file is updated as presented in appendix B.11.

Some variables are updated in the following files */etc/libvirt/libvirtd.conf*, */etc/init/libvirt-bin.conf*, */etc/default/libvirt-bin* as presented in appendix B.12. To load the new values, libvirt-bin service is restarted.

In the same manner openVSwitch was installed on the Network node, it needs to be installed on the Compute node as well.

Default virtual bridge needs to be deleted and as displayed in Figure 4.2 two virtual bridges are created on the Compute node; br-int for VM integration and br-eth1 for VM configuration. Bridge br-eth1 is attached to Eth1 NIC. These commands are presented in appendix B.13.

Quantum openVSwitch agent package is installed and configured. Variables presented in appendix B.14 are updated in */etc/quantum/plugins/openvswitch/ovs_ quantum_plugin.ini* file.

In */etc/quantum/quantum.conf* file check rabbit_host variable is set to the Controller node IP address. Finally quantum-plugin-openvswitch-agent is restarted.

Next nova-compute-kvm package is installed. Configuring Nova requires updating authtoken section in /etc/nova/api-paste.ini file as presented in appendix B.15. Also update /etc/nova/nova-compute.conf file as presented in appendix B.16. Modify the /etc/nova/nova.conf according to appendix B.4. Finally all Nova services are restart-

ed.
Now listing Nova service by running the following command will give an overview of all Nova service, whether they are running successfully or not.

At this point the setup of up a Compute node is completed.

4.1.2.5 Creating a VM

To start a VM, create a new tenant, user, and internal/external networks.
Note: The following operations are done on the Controller node. The commands to run these operations are presented in appendix B.17.

A new subnet is created inside the new tenant network. A virtual router is created and added to the subnet.
Then an external network is created with the tenant ID belonging to the service tenant. A subnet containing floating IPs is created. The router is set for the external network as well. VMs gain access to the metadata server locally present in the controller node via the external network. On the Controller node newly created routes are added. OpenStack Dashboard currently does not allow assigning floating IPs to VMs. This is done via command line. A floating IP is allocated to the project one tenant. Then associate the floating IP to the VM. Commands to perform these operations are presented in appendix B.17.

According to the manual, the VM is accessible and we can ping it [62].

At this point the OpenStack setup is completed. Table 4.2 provides the results observed in this experiment.

4.1.2.6 Observations and Outcomes

Table 4.2 provides the outcomes of the experiment described above related to the questions asked about deployment in the Methodology chapter.

Regarding the process of the OpenStack deployment described in this subsection, the following issues were encountered:

- The network addresses used in the manual guide were unnecessarily confusing and new relevant IP and network addresses had to be adopted.
- The deployment guide lacked proper networking configuration.
- After the deployment was finished, surprisingly 15 commands were required to create the first VM through terminal.
- The system failed to launch a VM instance.
- None of the several attempts made to debug the system were helpful.
- Understanding the deployment guide in details would require high knowledge of all involving technology in the cloud deployment.

1. Total number of packages installed:	27 Packages installed on Controller node

	8 Packages installed on Network node 10 Packages installed on Compute node **45 Packages installed totally**
2. Number of prerequisite packages:	**26 Packages**
3. Name of prerequisite packages:	3 × ntp mysql-server python-mysqldb rabbitmq-server 3 × vlan 3 × bridge-utils curl openssl iscsitarget iscsitarget-dkms open-iscsi memcached 2 × openvswitch-switch openvswitch-datapath-dkms cpu-checker kvm libvirt-bin pm-utils openvswitch-datapath-dkms
4. Total number of files configured:	23 Files on Controller node 8 Files on Network node 21 Files on Compute node **52 Files configured totally**
5. Total umber of lines configured:	**232 Lines**
6. Total time taken by deployment:	**10 Hours**
7. Total time taken by partial automatic deployment:	**40 Minutes**
8.Total number of lines included in automation script:	**413 Lines**
9. Automatability level of deployment:	Between 75% and 100% **Actually 90%**
10. Deployment ended with expected behavior:	**No**
11. Debugging level:	**Very Hard**
12. Level of expertise required by deployment	**Very High**

Table 4.2 Results of OpenStack three-node deployment

4.1.3 OpenStack Folsom Three Node Setup on Ubuntu 12.04 (LTS)

This section covers a basic OpenStack Folsom deployment for development purposes

on top of Ubuntu 12.04 (LTS) operating system. This OpenStack deployment guide is provided by the OpenStack website and is part of the OpenStack documentation. It aims at deploying an OpenStack infrastructure for testing purpose. Figure 4.3 shows the topology of the infrastructure designed for deploying an OpenStack cloud.

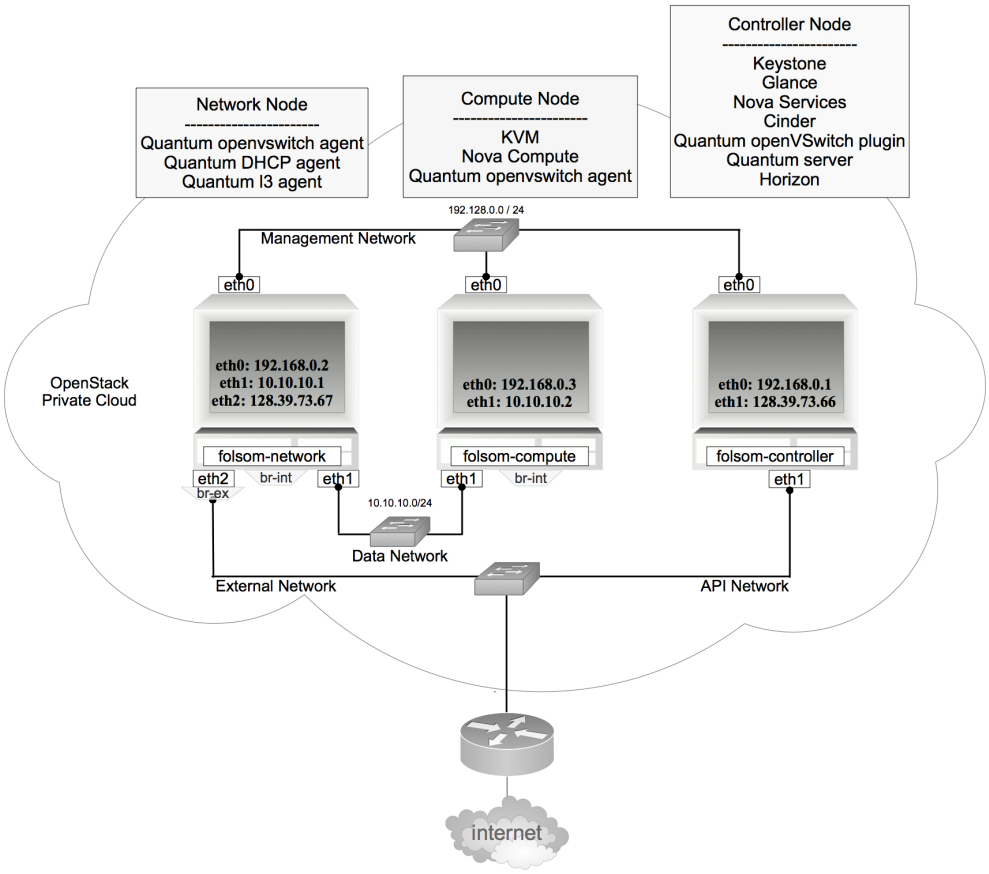

Figure 4.3 Topology of OpenStack Folsom three-node deployment

4.1.3.1 Requirement

This setup requires at least three machines servers with Ubuntu 12.04 (LTS). In addition to having enough network interfaces, three NICs on the Network node and two NICs for each of Controller node and Compute node are required. Also, on the Controller node two hard drives are required.

4.1.3.2 Setting up Controller Node

The Controller node in this OpenStack infrastructure provides the following services:

- Databases (with MySQL)
- Queues (with RabbitMQ)
- Keystone, Glance
- Nova
- Cinder
- Quantum Server (with Open-vSwitch plugin)
- Dashboard (with Horizon)

Note that Ubuntu 12.04 LTS by default has OpenStack Essex version. So before updating and upgrading the system we tweak the system to use the Ubuntu cloud Archive for Folsom and run the commands presented in appendix C.6. Then we update and upgrade the system.

Network interfaces on the Controller node are configured as presented in appendix C.7.

Update the following variables in */etc/sysctl.conf* file and restart networking.

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Notice the hostname for the Controller node is "folsom-controller". All three nodes building the cloud must be able to resolve each others names. So names of all nodes together with their corresponding IP addresses are placed in */etc/hosts*.

NTP server is installed and configured on all nodes to keep them Synchronized. Controller node with Ubuntu's NTP server and other nodes synchronize with Controller node.

Same as previous OpenStack setups, MySQL server is installed and configured to accept all incoming requests. Then all required databases with database users are created as presented in appendix C.8.
Next rabbitmq-server package is installed to provide the queue server. Default password for RabbitMQ server is changed to prevent unauthorized access to queue server as following:

```
>rabbitmqctl change_password guest password
```

Next we install Keystone, but notice that in previous setups there was just one keystone package to install Keystone, while this setup requires two more packages; python-keystone and python-keystoneclient packages. We install all three packages and configure */etc/keystone/keystone.conf* file as presented in appendix C.9. Then Keystone service is restarted and synchronized with its corresponding database.

A file is created with environment variables and loaded into the system as presented in appendix C.10.

There are two scripts containing the initial data for the Keystone database, developed by the OpenStack team to be suitable for this setup. The links for these scripts are given in appendix C.11. These scripts are downloaded and executed.

Subsequently, Glance and its dependency packages are installed and configured. Note that Glance was installed just with one glance package in previous setups while this setup requires glance-api, glance-registry, python-glanceclient, and glance-common packages as well. To configure Glance must update some variables in both */etc/glance/glance-api.conf* and */etc/glance/glance-registry.conf* files as presented in appendix C.12. Finally glance-api restart and service glance-registry services are restarted and Glance is synchronized with Glance database to create tables as presented in appendix C.13. To test the Glance service an image is downloaded and added to the Glance image store as shown in appendix C.13. Now listing existing images stored in

Glance image store will have to display the newly added image (appendix C.13), which confirms the image successfully being added to image store.

Next, corresponding Nova packages are installed. In this setup python-nova, python-novaclient, and nova-common packages are installed. (appendix C.14).

To configure Nova some variables in */etc/nova/api-paste.ini* file are updated as presented in appendix C.14.

Since Cinder is going to be used for volumes, each part concerning "nova-volume" should be deleted in the same file */etc/nova/api-paste.ini*. Some sed command presented in appendix C.14 help to delete these parts from */etc/nova/api-paste.ini*. The content of */etc/nova/nova.conf* file is modified as presented in appendix C.4. Last step to complete the Nova configuration is to synchronize Nova with nova database to create Nova tables and restart Nova services (appendix C.14).

Next the Cinder volume manager is installed and configured as presented in appendix C.15. */etc/default/iscsitarget* is configured and iSCSI services are started. Then in */etc/cinder/cinder.conf* and */etc/cinder/api-paste.ini* files some variables are updated.

In the previous setup Cinder used /dev/loop2 to provide volumes to VMs but in this setup a second hard disk is formatted and used to provide volumes for VMs. Hence a second hard disk is formatted with a Linux partition. Then a physical volume is created on /dev/sdb1 followed by a volume group called Cinder-volumes on /dev/sdb1. Finally Cinder is synchronized with the cinder database. To create tables and Cinder services are restarted (appendix C.15).

At this point Quantum is installed. To configure Quantum some variables are updated in */etc/quantum/quantum.conf, /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini* and */etc/quantum/api-paste.ini* files. Finally quantum-server service is restarted to complete the Quantum server configuration (appendix C.16).

The last OpenStack Component to be installed on Controller node is Dashboard (Horizon). Accordingly, the Horizon required packages are installed (appendix C.17).

At this point setting up Controller node is completed. OpenStack Dashboard can be accessed at http://128.39.73.66/horizon by the following credentials.
Username : admin
Password: password
or
Username: demo
Password: password

4.1.3.3 Setting up Network Node

The Network node provides Virtual Bridging, DHCP Server and Virtual Routing.

Since Ubuntu 12.04 LTS by default has OpenStack Essex, prior to updating the system use Ubuntu cloud Archives for Folsom as presented in appendix C.6. Then the system is updated and upgraded.

Networking is configured as shown in appendix C.18.
In */etc/sysctl.conf* file the following variables are updated.

```
net.ipv4.ip_forward=1
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

Networking is restarted to apply the new changes. The system must be able to resolve its hostname "folsom-network" and others nodes as well. Therefore /etc/hosts is configured with this data in order to resolve all.

Next the NTP server is installed and configured to synchronize with the Controller node (appendix C.19).

Network services are provided by installing the quantum-plugin-openvswitch-agent and quantum-dhcp-agent quantum-l3-agent packages. Then the openvswitch-switch service is started.

As figure 4.3 displays, there are two bridges in the Network node, namely the br-int for internal communication and the br-ex for external communication. These bridges are created and br-ex is attached to Eth2 NIC.
In */etc/quantum/l3 agent.ini*, */etc/quantum/api-paste.ini*, */etc/quantum/quantum.conf* */etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini* file and */etc/quantum/dhcp agent.ini* files some variables are updated as presented in appendix C.20. Finally all quantum services are restarted for new configuration to take place (appendix C.20).

To create virtual networking, firstly define the environment variables in a file and then load them into the system (appendix C.21). In the previous OpenStack setup for three-node architecture in the section for "Creating a VM" section, several operations were needed to create routers and networks. In this setup, however, operations all are collected in a script developed by the OpenStack development team. The link pointing to this script is presented in appendix C.22. This script is downloaded and some network numbers are modified to suit the numbers prepared for in this project.
For the configuration of L3 service, external network ID is assigned to gateway_external_network_id variable in */etc/quantum/l3_agent.ini* file. The router ID is assigned router_id variable in */etc/quantum/l3_agent.ini*. Finally, the quantum-l3-agent service is restarted.

At this point the setup of the Network node is completed. Building up the OpenStack cloud continues with setting up a Compute node.

4.1.3.4 Setting up Compute Node

The Compute node provides hypervisor (KVM), Nova-compute, and Quantum OVS Agent.

Since Ubuntu 12.04 LTS by default has OpenStack Essex, prior to updating the system use Ubuntu cloud Archives for Folsom as presented in appendix C.6. Then system is updated and upgraded.

Network interfaces on the Compute node are configured as presented in appendix C.23.
In */etc/sysctl.conf* file the following variables are updated.

```
net.ipv4.conf.all.rp_filter = 0
net.ipv4.conf.default.rp_filter = 0
```

/etc/hosts is configured to resolve the Compute node's hostname "folsom-compute" as well as other nodes.

Further, the NTP server is installed and configured to synchronize with the Controller node (appendix C.19).

The KVM hypervisor and its dependency packages are installed. Some variables are updated in */etc/libvirt/qemu.conf* , */etc/libvirt/libvirtd.conf, /etc/init/libvirt-bin.conf,* in */etc/default/libvirt-bin* files. Finally, libvirt service is restarted to load the new configurations (appendix C.24).

The next step Nova is to install. In */etc/nova/api-paste.ini and /etc/nova/nova-compute.conf* files some variables are updated as presented in appendix C.25. Content of */etc/nova/nova.conf* is provided in appendix C.4. Nova services are restarted to make configurations take place.

Then the OpenvSwitch service is installed and service is started. Internal bridge "br-int" is added. quantum-plugin-openvswitch-agent is installed. Some variables in */etc/quantum/quantum.conf* and /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini files are updated as presented in appendix C.26. Finally the Quantum agent service is restarted.

4.1.3.5 Creating a VM

Now the OpenStack can be managed via OpenStack API or Dashboard.
After logging into OpenStack Horizon the default security policies are added to allow ICMP and SSH. Then a personal keypair is created and an instance can be launched. Since Horizon does not manage L3 in Folsom release of OpenStack, Quantum CLI is used to configure the floating IP. Then a floating-IP is created and attached to the virtual port of the VM that was launched.

At this point the OpenStack setup is completed. Table 4.3 provides the results observed for this experiment.

As noticed, the OpenStack deployment explained above had a lot of similarities with the second OpenStack cloud deployment presented in subsection 4.1.2. Nevertheless, these two had some differences described as following:

- By default Ubuntu 12.04 LTS provides the OpenStack Essex release, therefore to install Folsom release on Ubuntu 12.04 ubuntu-cloud-keyring package had to be installed. Ubuntu 120.10 did not require this.
- The *keystone-data.sh* and *keystone-endpoints.sh* scripts that were used to fill keystone database with data (users, tenants, services) and create the endpoints,

were downloaded from different sources on the second and third OpenStack deployments.

- Creating the network for the OpenStack cloud is done via a script on the OpenStack deployment for Ubuntu12.04, while on the second deployment creating a network was done manually.
- There are some configuration differences in configurations presented in deployment guides such as networking configurations.
- The deployment guide for Ubuntu 12.04 is much more readable and organized.
- The main difference is the results each deployment guide produces.

4.1.3.6 Observations and Outcomes

Table 4.3 provides the outcomes of the experiment described above related to the questions asked about deployment in the Methodology chapter.

1. Total number of packages installed:	37 Packages installed on Controller node 5 Packages installed on Network node 8 Packages installed on Compute node **50 Packages installed totally**
2. Number of prerequisite packages:	**21 Packages**
3. Name of prerequisite packages:	1. 3 × ntp 2. 3 × ubuntu-cloud-keyring 3. mysql-server 4. python-mysqldb 5. rabbitmq-server 6. iscsitarget 7. open-iscsi 8. iscsitarget-dkms 9. linux-headers-`uname -r` 10. apache2 11. libapache2-mod-wsgi 12. memcached 13. python-memcache 14. kvm 15. libvirt-bin 16. pm-utils 17. openvswitch-switch
4. Total number of files configured:	17 Files on Controller node 11 Files on Network node 11 Files on Compute node **39 Files configured totally**
5. Total umber of lines configured:	**212 Lines**
6. Total time taken by deployment:	**10 Hours**
7. Total time taken by partial automatic deployment:	**40 Minutes**

8.Total number of lines included in automation script:	**451 Lines**
9. Automatability level of deployment:	Between 75% and 100% **Actually 95%**
10. Deployment ended with expected behavior:	**Partially**
11. Debugging level:	**Very Hard**
12. Level of expertise required by deployment	**Very High**

Table 4.3 Results of OpenStack three-node deployment

Among the previous three OpenStack deployments, the last experiment was the only successful OpenStack deployment, where a VM could be created and accessed via VNC client on OpenStack web interface. The deployment guide provided a well-described cloud infrastructure design and deployment. Still, there were some issues regarding the OpenStack deployment as are explained as following:

- No credentials for logging into the VM instance launched from a VM image provided by OpenStack web site (ubuntu-12.04-server-cloudimg-amd64-disk1.img) were provided. All default logging credentials were tried but none of them could log in to the VM instance. However, another VM instance launched from cirros-0.3.0-x86_64-disk.img could be logged in. This is because the VM itself provided the right default credentials.
- The VM could not be accessed via SSH.
- Even though the VM was assigned with a public IP it cold not be accessible from internal and external networks.
- It should be noted that all the commands that were run to create the first VM in the second OpenStack deployment experiment, were handled through a script provided by a manual guide.

4.1.4 Evaluation Results Related to Usability, Support, Compatibility and License

Table 4.4 provides the results of investigations related to usability, support, compatibility and license criteria of OpenStack.

1. Supported operating systems:	Only **64-bit Linux** (CentOS, Debian, Fedora, RHEL, openSUSE, SLES, and Ubuntu)
2. Number of available different deployment instructions:	Four deployment guides [62-65]
3. Number of published books about the platform:	OpenStack cloud Computing Cookbook[66] Deploying OpenStack[67] OpenStack Operations Guide[68]
4. Commercial support exist:	**Yes,** by third parties such as Canonical, Red Hat, Rackspace, SUSE etc

5. User interfaces:	Command line Web based GUI (Horizon)
6. Compatibility with similar technologies	Compatible with EC2 API and OCCI (supports Open Cloud Computing Interface)
7. Cloud platforms license:	Apache 2.0 license

Table 4.4 General information about OpenStack

Regarding the information provided in Table 4.4 the following points should be considered:

- The OpenStack platform is only available for 64-bit Linux Operation systems.
- Despite of the fact that several OpenStack deployment guides are found, not all are useful. Among three of the OpenStack deployment guides used in this thesis, only one was useful.
- Among three books existing about OpenStack platform only two of them were reviewed in this thesis. OpenStack cloud Computing Cookbook[66] covers how to build and manage an OpenStack prototype within a VirtualBox environment. Therefore this book was not used much in this thesis. The other book OpenStack Operations Guide[68] was published recently (15.05.2013) at the Rackspace Austin office. When the latter book was published, the experiments with the OpenStack deployments had been carried out and thus the book was not used in this thesis.

4.2 OpenNebula

In order to setup an OpenNebula cloud in this thesis several sources were used, however it is mainly based on the official documentation of OpenNebula.[20, 69-71]

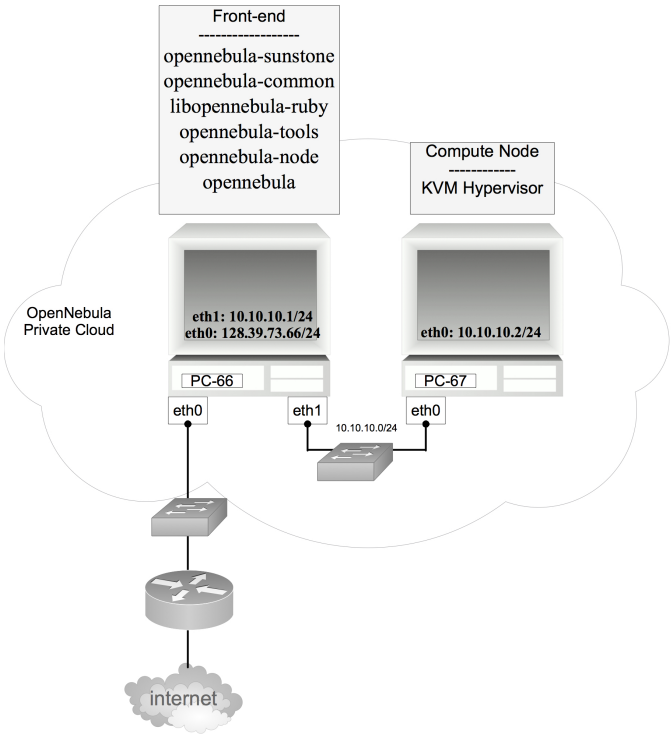

Figure 4.4 Topology of OpenNebula deployment

4.2.1 Setting up Front-end

Figure 4.4 displays the topology of the OpenNebula cloud setup for the purpose of this thesis with one Front-end and one Compute node. As mentioned in the Background chapter, Front-end is the server that runs the OpenNebula software. Furthermore, an OpenNebula cloud can have multiple Compute nodes (hypervisor-enabled hosts), where each Compute node can have a different hypervisor. The hypervisor used on the Compute node in this setup uses KVM.

Ubuntu 12.04 is installed on both Front-end and Compute node. Then both machines are connected and configured according to the network topology given by Figure 4.4.

At the time of writing this thesis, the OpenNebula software has the opennebula-4.0.0-1 version and it is available in form of both source-code and binary packages from the OpenNebula website.[72] Unfortunately only the old version of OpenNebula software can be installed through the apt-get method. The binary packages of the OpenNebula software are downloaded as a compressed tar file. The tar file contains the following packages:

- **opennebula-common**: provides the user and common files
- **libopennebula-ruby**: all ruby libraries
- **opennebula-node**: prepares a node as an opennebula-node
- **opennebula-sunstone**: OpenNebula Sunstone Web Interface
- **opennebula-tools**: Command Line interface
- **opennebula**: OpenNebula Daemon

The software packages are decompressed in */tmp/opennebula-3.8.3/* directory and installed with following command:

```
> sudo dpkg -i /tmp//tmp/opennebula-4.0.0-1/*.deb
```

Then in order to fix package dependency, running the following command is required:

```
>sudo apt-get install -f -y
```

In order to run correctly, some OpenNebula components require Ruby libraries. To ease the installation of these dependencies, OpenNebula has included a script that detects the Linux distribution and installs the required libraries. These libraries are as following:

- sqlite3 development library
- mysql client development library
- curl development library
- libxml2 and libxslt development libraries
- ruby development library
- gcc and g++
- make
- ruby >= 1.8.7

This script is located in the */usr/share/opennebula/* folder. The script is run as shown in the following command:

```
>/usr/share/opennebula/install_gems
```

Installing the OpenNebula package will create "oneadmin" user and group in the front-end. This account is used to run the OpenNebula services, administer and maintain OpenNebula tasks. Thus it is practical to create a password for the "oneadmin" user account and log into the system as "oneadmin" user. An alternative to this is to run the commands by using the "sudo -u oneadmin" method.

```
>sudo  passwd oneadmin
```

The user ID and group ID for "oneadmin" is required because later a "oneadmin" user account with the same user ID and group ID will be created on the Compute node. This account is used by the OpenNebula Front-end to connect and execute commands. We get the uid and gid for "oneadmin" user by the following command:

```
>id oneadmin
uid=106(oneadmin) gid=113(cloud) groups=113(cloud),114(libvirtd),115
(kvm)
```

Next on the Compute node "cloud" group is added and the "oneadmin" account with the OpenNebula /var/lib/one folder as the home folder is created.

```
>groupadd --gid 113 cloud
>useradd --uid 106 -g oneadmin -d /var/lib/one oneadmin
```

Now on the Front-end "oneadmin" user needs to be able to log into the "oneadmin" account on the Compute node using ssh without requiring a password. This can be achieved by running the following command on the Front-end:

```
>ssh-copy-id oneadmin@host1
```

Notice that host1 is the Compute node name for the Compute node. After running the previous command password-less ssh logging into "oneadmin" user on the Front-end must be possible.

Now logged into the Front-end as "oneadmin" user we are ready to start the OpenNebula daemons by the following command:

```
>one start
```

To verify if the OpenNebula daemon is running, the following command can be run on the Front-end:

```
>onevm list
```

In the case of an empty list of VMs OpenNebula is working, otherwise there has been some errors which can be tracked by checking the log files located at /var/log/one/ directory.

OpenNebula needs a database backend, where usually MySQL database is recommended. Thus MySQL server is installed and a database called “one” is created and granted the necessary permisions:

```
>mysql -u root -p
mysql> create database one;
mysql> CREATE USER 'one'@'localhost' IDENTIFIED BY 'onepassword';
mysql> GRANT ALL ON one.* TO 'one'@'localhost';
mysql> quit
```

After OpenNebula daemon is started it runs with its default configuration. Thus it is configured to fit our infrastructure requirements. The main configuration file is *oned.conf*, and is located at */etc/one* directory. The content of the *oned.conf* file is presented in appendix D.3. After applying the changes in the configuration file the OpenNebula daemon needs to restart for the changes to take place.

4.2.2 Setting up Compute Node

No OpenNebula component in required to be installed on Compute nodes. The following list shows the only requirements on any Compute node:

- ssh server running
- hypervisor working properly configured
- **ruby** >= 1.8.7

After installing a fresh Ubuntu 12.04 OS, the first step is to create **“oneadmin“** OpenNebula dedicated user account that will be used by the OpenNebula Front-end to connect and execute commands.

```
>sudo adduser oneadmin
```

After creating oneadmin user we setup a password-less login from the oneadmin user account on the Front-end by the following command:

```
oneadmin@front-end > ssh-copy-id oneadmin@host1
```

Now oneadmin account on the Front-end can connect to the oneadmin account of the Compute node without entering any password.

To provide oneadmin account on the Compute node with administrative privileges, it is added to the sudo group. The following change is applied in */etc/sudoers* file, so every user in the sudo group can execute any command with root privileges, without entering any password.

```
%sudo ALL=NOPASSWD: ALL
```

The oneadmin user is added to the sudo group with the following command:

```
>sudo adduser oneadmin sudo
```

Next, network bridges are configured. Notice every Compute node on the OpenNebula infrastructure need a bridge configured with the same name. Appendix 4.D displays the network configurations applied in */etc/network/interfaces*.

The networking service is restarted for new network configurations to take place.

Now it is time to install the hypervisor, which in this case is KVM. In order to install KVM on the Compute node, a kernel with kvm-intel or kvm-amd module is needed. After making sure that the Compute node hardware has support for virtualization, KVM and its perquisite packages are installed by the following command:

```
> sudo apt-get –y install libvirt-bin qemu-kvm ruby
```

It is required to add the "oneadmin" user to libvirt daemon groups by the following command:

```
>sudo adduser oneadmin libvirtd
```

Since live migrations are directly managed by libvirt, it is needed to enable the libvirt TCP port in */etc/default/libvirt-bin.*

```
libvirtd_opts="-d -l"
```

And in */etc/libvirt/libvirtd.conf* file uncomment the following variable.

```
listen_tcp = 1
```

For every Compute node to be used on the OpenNebula cluster, it needs to be registered on the Front-end. The Compute node used in this OpenNebula is registered on the Front-end by:

```
>onehost create host1 --im im_kvm --vm vmm_kvm  --net vnet_dummy
```

The parameters passed to this command specify the names of the scripts, which retrieve information, manage virtual machines, and specifies a network driver to enforce traffic management on this particular Compute node (host1). After registering the new Compute node on the Front-end it needs to be enabled as well by:

```
>onehost enable 0
```

Notice 0 in the previous command is the ID of the Compute node returned when registering host1 on the Front-end node.

All the registered Compute nodes can be listed by the following command.
```
>onehost list
```

The first option passed to register the Compute node is the Information Manager driver, which is used to gather information from the Compute node, and it depends on the hypervisor used on the Compute node. The Front-end transfers a few scripts to the Compute node by ssh/rsync and remotely executes them on the Compute node. The

default Information Manager scripts executed for a KVM based Compute node are found in */var/lib/one/remotes/im/kvm.d* folder.

The second option passed to register the Compute node is the virtualization driver, which is used to create, manage, and monitor VMs on the Compute nodes. The default virtualization driver for a Compute node using KVM are found in */var/lib/one/remotes/vmm/kvm* folder.

The last parameter passed to create the new Compute node on the Front-end mentions the networking driver, which is used to configure a particular network driver that is used when every new VM is launched.

Among several available network drivers (dummy, fw, ebtables, 802.1Q, ovswitch, vmware), dummy is used in this setup. Dummy is the default driver that does not enforce any particular network policy. Any VM connected on the same physical bridge can freely talk to the other VMs.

Another key component of OpenNebula cloud infrastructure is the Storage, which is a separate storage needed to run virtual machines and to maintain an Image Repository that is used to bootstrap the new VMs.

The default folder used for storage is */var/lib/one/datastores/*. A separate folder is created for each virtual machine with an incremental ID as name. This folder contains the automatically generated configuration files for the hypervisor and VM image. The images of the Image Repository are saved in */var/lib/one/images/* folder.

There are two main types of storage:

- Non-shared storage
- Shared storage

Non-shared storage does not require any additional hardware or software requirements and has the simplest form. On this storage type images are copied entirely from one Front-end to the Compute node through a file transfer protocol before launch. Thus deploying the VM, migration and shutdown procedures will be a slower. Because of this, password-less SSH login for oneadmin user from the Front-end to Compute node is needed.

In this setup we have to make sure that oneadmin user has access-right in to */var/lib/one/images/* folder.

By shared storage every Compute node is given direct access to the Image Repository. It provides the ability for near-instant deployment, migration, and shutdown procedures for every VM. Using NFS (through NAS or SAN) is a common choice when using shared storage.

Now let us add an image to the Front-end and create a VM.
A test image is downloaded from the following link in to /tmp/ folder.
`http://dev.opennebula.org/attachments/download/355/ttylinux.tar.gz`

Then the content of the file is extracted in */tmp/ttylinux/* folder.

We specify the network IP in */tmp/ttylinux/small_network.net* and configure the Compute node network bridge as follows:

```
NAME = "Small network"
TYPE = FIXED
#Now we'll use the cluster private network (physical)
BRIDGE = lan0
LEASES = [ IP="10.10.10.5"]
```

Also we provide the right path to the image-file in */tmp/ttylinux/image.one* file.

```
NAME = ttylinux
PATH = "/tmp/ttylinux/ttylinux.img"
TYPE = OS
```

The templates are then submitted to OpenNebula on the Front-end by the following commands:

```
>onevnet create small_network.net
> oneimage create image.one -d default
```

Each of these commands returns an ID. In */tmp/ttylinux/ttylinux.one* file we adjust the right IDs IMAGE_ID and NETWORK_ID to match the ID returned by previous commands.

```
DISK  = [ IMAGE_ID = 0 ]
NIC   = [ NETWORK_ID = 0 ]
```

Finally a VM is created by the following command:

```
>onevm create /tmp/ttylinux/ttylinux.one
```

Status of the VM can be seen by listing the VMs.

```
>onevm list
```

OpenNebula has a web interface called Sunstone. Sunstone is already installed since it was included in the binary package. The sunstone service can be started by the following command where –H option overwrites the default (localhost) to the public address of the Front-end.

```
> sunstone-server start -H 128.39.73.66
```

Checking */var/log/one/sunstone.log* file shows the status of the sunstone. By default sunstone uses port number 9869. This port-number can be changed by starting sunstone with –p option and the new port number.

Finally Sunstone can be accessed from 128.39.73.66:9869. An OpenNebula user credentials are required to log into Sunstone.

A new user can be created by following command:

```
> oneuser create oneadmin password
```

After logging in to the sunstone the main menu on the left-hand side of the window has the options for managing and monitoring the OpenNebula cloud service. The Vir-

tual Resources menu allows managing the Virtual Machine Templates, Instances and storage (Images). Using Virtual Machine templates allow instantiation of VMs, as many as required. A VM can be created from "Create a new Virtual machine" tab.
The Dashboard menu provides an overview of nodes, VMs and resources used in the Cloud.

At this point the OpenStack setup is completed. Table 4.5 provides the results observed in this experiment.

4.2.3 Observations and Outcomes

Table 4.5 provides the outcomes of the experiment described above related to the questions asked about deployment in the Methodology chapter.

1. Total number of packages installed:	17 packages installed on Front-end 4 packages installed on Compute node **22 Packages installed totally**
2. Number of prerequisite packages:	**13 Packages**
3. Name of prerequisite packages:	1. sqlite3 2. mysql 3. curl 4. libxml2 5. libxslt 6. gcc 7. g++ 8. make 9. ruby >= 1.8.7 10. libvirt-bin 11. qemu-kvm 12. bridge-utils 13. pm-utils
4. Total number of files configured:	1 files on Front-end 3 files on Compute node **4 Files configured totally**
5. Total number of lines configured:	**49 Lines**
6. Total time taken by deployment:	**3 Hours**
7. Total time taken by partial automatic deployment:	**30 Minutes**
8.Total number of lines included in automation script:	**70 Lines**
9. Automatability level of deployment:	Between 75% and 100% **Actually 80%**
10. Deployment ended with expected behavior:	**No**
11. Debugging level:	**Hard**
12. Level of expertise required by deployment	**High**

Table 4.5 Results of the OpenNebula deployment process

Regarding the OpenNebula deployment explained above, the following issues were encountered:

- There did not exist a precise deployment instruction that would guide the deployment step by step, thus the main documentation provided on the OpenNebula website had to be reviewed and learned. The "OpenNebula 3 cloud Computing"[20] book provided good instructions on how to deploy the OpenNebula platform on Ubuntu operating system. Unfortunately, however, configurations mentioned in the book did not match the configurations of the latest release of the OpenNebula platform. Thus to deploy an OpenNebula cloud both official documentation and "OpenNebula 3 cloud Computing"[20] were used.
- After finishing the deployment, OpenNebula had some errors that could be tracked by log files. Some errors could be fixed and but due to time limits some errors remained unfixed. Nevertheless, it should be noted that the unsolved errors were related to the KVM hypervisor.
- Related to the level of automatability of OpenNebula, it can be said that most of OpenNebula installation and configuration can be automated. However, there are still some things, such as managing the Compute nodes on the Front-end node, that cannot be done by automation before having the Compute nodes ready.

4.2.4 Evaluation Results Related to Usability, Support, Compatibility and License

Table 4.6 provides the results of investigations related to usability, support, compatibility and license criteria of OpenNebula.

1. Supported operating systems:	CentOS, Debian, openSUSE, and Ubuntu
2. Number of available different deployment instructions:	Two deployment guides [69, 71]
3. Number of published books about the platform:	OpenNebula 3 cloud Computing[20]
4. Commercial support provided:	**Yes,** by C12G Labs OpenNebula
5. User interfaces:	Command line Web based GUI (Sunstone)
6. Compatibility with similar technologies:	Compatible with EC2 API, OCCI (Open Cloud Computing Interface) and vCloud
7. Cloud platforms license:	Apache 2.0 license

Table 4.6 General information about OpenNebula

Regarding the information provided in Table 4.6, the following points are explained:

- The OpenStack packages are only provided for 64 bits, but from source they can be compiled for 32 bits architectures.
- Beside the instructions on deploying OpenNebula, another OpenNebula deployment guide [69, 71] for CentOS operating system is referenced on the OpenNebula web site.

- The "OpenNebula 3 cloud Computing"[20] book was found to be highly useful book with excellent instructions and providing good examples. This book makes it easy to understanding the OpenNebula platform. Unfortunately, however, some OpenNebula commands and configurations presented in the book do not match the latest release of the OpenNebula platform.
- The C12G Labs (OpenNebula.pro) provides the commercial support to businesses and organizations.

To give an easier overview of the previous tables in this chapter, Table 4.7 is created. This table is a combination of Table 4.1, Table 4.2, Table 4.3 and Table 4.5.

	X1 OpenStack	X2 OpenStack	X3 OpenStack	OpenNebula
Base OS	Ubuntu 12.10	Ubuntu 12.10	Ubuntu 12.04	Ubuntu 12.04
Packages installed	34	45	50	22
Files configured	27	52	39	4
Lines configured	228	232	212	49
Number of prerequisite packages	19	26	21	13
Time used by manual deployment	7 Hour	10 Hour	10 Hour	3 Hour
Time used by automatic deployment	30 Minutes	40 Minutes	40 Minutes	30 Minutes
Automatability Level	90%	90%	95%	80%
Number of codelines used in automating the deployment	311	413	451	70
Installation behaved as expected	No	No	Partially	No
Debugging level	Very Hard	Very Hard	Very Hard	Hard
Expertise level required	Very High	Very High	Very High	High

Table 4.7 Results of OpenStack and OpenNebula deployments in one table

As can be seen from Table 4.7 the deployment of OpenStack requires nearly twice as many packages as what does the deployment of OpenNebula. The number of files configured during the deployment of OpenNebula is a lot lower than the number of files configured during the deployment of OpenStack. Comparing the total number of lines configured during the deployment of OpenStack to the total number of lines configured during the deployment of OpenNebula, we see that the number of lines configured during the deployment of OpenStack is three to four times higher than that of OpenNebula.

The time required by the deployment of OpenStack is twice as much time used to deploy OpenNebula. Consequently depending on the amount of configuration needed, the number of codelines written to automate OpenStack is four to seven times more than the number of codelines written to automate OpenNebula. As the Automatability level of the software shows, most of the deployment process of both OpenNebula and OpenStack can be automated. It can be seen that the debugging level for OpenStack is higher than that for OpenNebula. This implies that OpenStack is much more complex than OpenNebula. Among the three OpenStack deployments, the last deployment was

partially successful, meaning that a VM instance could be created and logged into via VNC, while it could not be accessed via SSH.

Considering the total number of packages required by the deployment of both OpenStack and OpenNebula, the number of prerequisite packages used by OpenStack and OpenNebula deployments are relatively similar. It can be seen that none of the experiments behaved as expected when the cloud deployment was finished. Deploying OpenStack requires much higher level of expertise than does OpenNebula.

Comparing Table 4.4 and Table 4.6, the following comparisons are considered:

- Both OpenNebula and OpenStack provide packages for CentOS, Debian, openSUSE and Ubuntu, while in addition OpenStack provides packages for Fedora, RHEL and SLES as well. While OpenStack packages are only available for 64-bit architecture, the OpenNebula platform can be compiled for 32 bits architectures from source.
- OpenStack has fewer deployment manuals than OpenNebula.
- There exist two more books about OpenStack than OpenNebula.
- Regarding commercial support, third parties provide commercial support for OpenStack platform while OpenNebula itself provides commercial support.
- Both cloud platforms provide command line tools and web GUIs.
- Considering compatibility with other cloud technologies, both cloud platforms are compatible with EC2 and OCCI, while OpenNebula provides compatibility with vCloud as well.
- Both OpenStack and OpenNebula open source cloud platforms are distributed and licensed under the terms of the Apache License, Version 2.0.

4.3 Evaluating Complexity

Table 4.8 presents an overview of the general information about the source code of the OpenStack components and OpenNebula. This information is created by using GitStats tool, which is a statistics generator for Git repositories. The following two commands show an example of how the Gitstats tool is used to generate an overview of Quantum source code as HTML page in the *quantumStatistics* folder within the current path.

```
>git clone https://github.com/openstack/quantum.git
>gitstats quantum/ quantumStatistics
```

Project name	Cinder	Glance	Horizon	Keystone	Nova	Quantum	Swift	Open-Nebula
Age in days	391	4483	866	771	1095	878	1052	1799
Total amount of-Files	517	337	931	331	2558	783	284	1560
Total amount of Codelines	9959	82554	166424	43014	141838	229173	48920	293621
Total amount of Commits	1087	2597	2519	3261	20709	2658	2639	6683
Average commits per active day	4.0	3.9	4.4	5.6	20.2	4.9	3.8	6.6
Number of Authors	123	144	134	181	454	162	116	37

Table 4.8 OpenStack and OpenNebula source code evaluation

As Table 4.8 shows The OpenStack platform is a combination of seven components while OpenNebula platform consists of only one component.

Considering the total number of lines written in the source code, both cloud platforms provide an important parameter for evaluation of the complexity of the cloud platforms. Based on Table 4.8, the OpenStack components are listed according to their number of source codelines in descending order:

- Quantum 229173
- Nova 141838
- Horizon 166424
- Glance 82554
- Swift 48920
- Keystone 43014
- Cinder 9959
- All-together 721882

If all the OpenStack components were assumed to be as one component it would have 721882 lines of source code. This number in comparison to the number of the OpenNebula source (293621) is more than double. The previous list showed that among the OpenStack components, Quantum has the biggest number of source code lines. In fact it is nearly twice the number of source code in Nova. Cinder appears to be the smallest of all the OpenStack components. This modular architecture of the OpenStack platform makes it possible to see the source code size of different components compared to other components. Considering the size of the source code as complexity factor of a software, the OpenStack could be pointed to as more of a complex software.

As described in the Background chapter, the OpenStack platform is written in Python while the OpenNebula platform is written mostly in C++, but also in Ruby and Java. Considering the involvement of various programing languages in the development of cloud platform, OpenStack is written only in Python while three languages are used to build OpenNebula. Naturally when the development of software involves several different programing languages, the complexity of the software increases. In this aspect the OpenNebula platform would be considered more complex than the OpenStack platform.

4.4 Evaluating the User- and Developer Community

This section evaluates the community of the cloud platforms based on two studies. The first study is based on an ongoing research project [49-51] as mentioned in the Literature section in the Background chapter. This study provides the results of analyzing the user- and developer community for OpenStack, OpenNebula, Eucalyptus and CloudStack cloud platforms. It should be noted that diagrams presented from Figure 4.5 to Figure 4.14, Table 4.9 and Table 4.10 are copyrighted and used with

permission. The second study will provide the author's personal experience of interacting with the user community of the OpenStack and OpenNebula cloud platforms.

4.4.1 Evaluation Based on the First Study

The data used in this analysis, such as the total number of topics (threads), messages (posts), and participants (unique email addresses or registered members), is extracted from the communication between community members in the form of mailing lists or public forum discussions.

Figure 4.5 shows the monthly number of topics (threads) discussed through the community over several years. Figure 4.6 represents the monthly number of messages discussed through the user community of cloud platforms. The monthly number of messages represents the volume of the discussions in the user community.

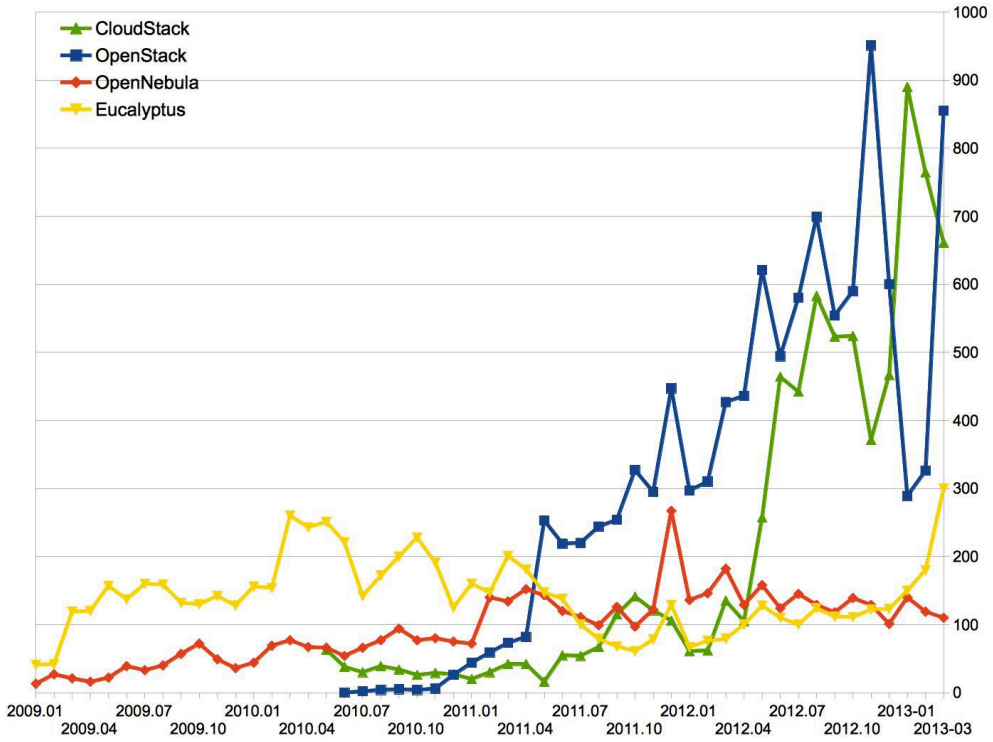

Figure 4.5 Monthly number of threads

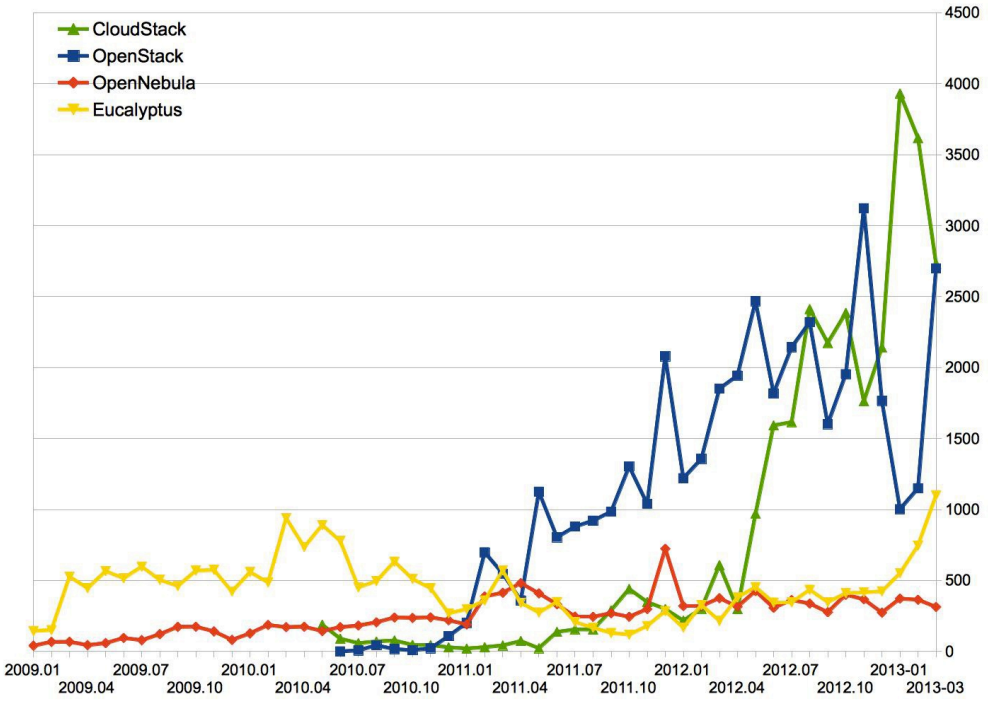

Figure 4.6 Monthly number of messages

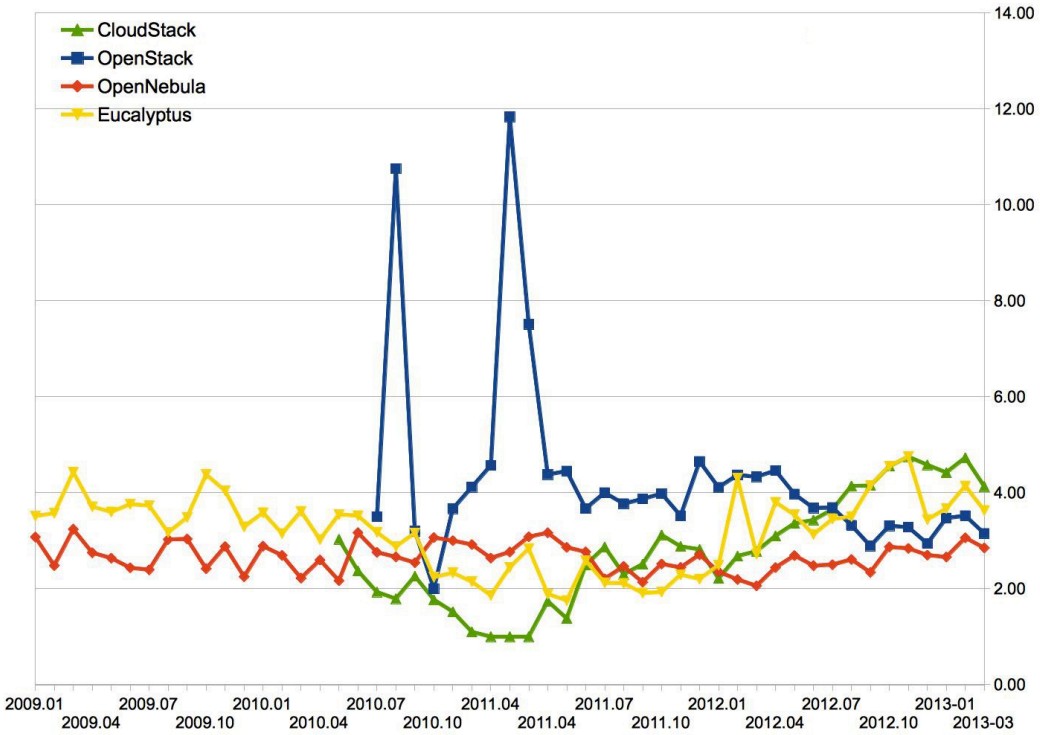

Figure 4.7 Monthly participation ratio

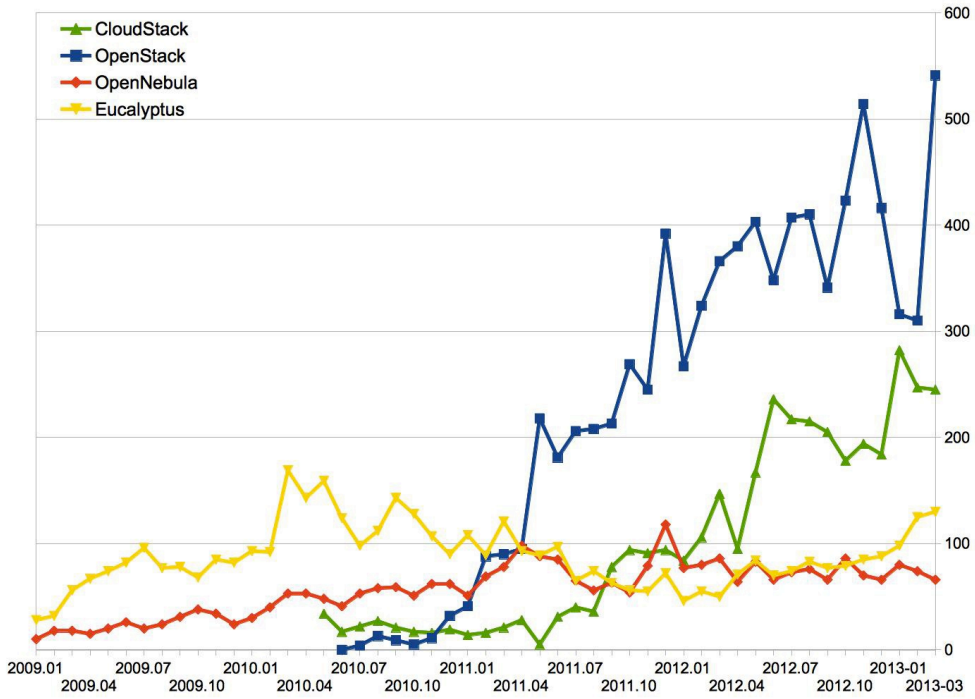

Figure 4.8 Monthly number of participants

As it appears in Figure 4.5, before October of 2012 OpenStack had the highest number of topics. Despite experiencing an extreme decline of number of topics during the last two months of 2012, it has since the beginning of 2013 up to the present time filled the gap and nearly regained its previous position. After OpenStack, CloudStack has the highest number of topics discussed. This number has decreased in the first three months of 2013. The number of topics discussed in the OpenNebula community is much lower than the number of topics discussed in the OpenStack community. It can be seen in the first three months of 2013, that the number of topics discussed in OpenNebula decreased slightly. Since the beginning of 2012, the number of topics discussed in the Eucalyptus user community has decreased; nevertheless, by the end of May 2013 it reached its peak.

Figure 4.6 indicates similar values as the diagrams in Figure 4.5, except for the fact that before October 2012 CloudStack had the highest volume of messages exchanged through the user community. While in the first three months of 2013 the number of messages exchanged through the OpenStack and the Eucalyptus user communities has increased, the volume of messages in the CloudStack and the OpenNebula user communities has decreased. Since the start of the OpenStack project, the volume of messages in the OpenStack community has been increasing until October 2012, where in last two months of 2012 it experienced a sharp decline in volume of messages. Nevertheless, OpenNebula has had the smallest volume of messages of all during the past twelve months.

Figure 4.7 represents the monthly participation rate of the online community, being the ratio between “the number of posts” and “the number of topics”. This parameter is used to measure the attention received by the community and the depth of discussion for a particular topic. For example, if the number of original posts that started a topic

were less than the number of replies, it implies that the participation of the forum or mailing list is high.

According to Figure 4.7, from the start of the OpenStack project to the second half of 2012, OpenStack had much higher participation ratio than all other communities. Since the beginning of 2012, OpenNebula has had the lowest rate of all, while the rate has gradually increased. It can be seen that from the end of 2012, first the CloudStack and then the Eucalyptus communities had the highest rate while the participation ratio for OpenStack and OpenNebula are similar to each other. We can see that in past 12 months, the participation ratio for OpenNebula has increased while the participation ratio for OpenStack has decreased.

Figure 4.8 represents the number of monthly active participants being discussed for the four projects. From Figure 4.8 it can be seen that first OpenStack and then Cloud stack had the highest number of active participants compared to the other two projects. The number of active participants for OpenStack is three to four times bigger than that of OpenNebula and Eucalyptus. We see that during the first three months of 2013, the number of monthly active participants for OpenStack and Eucalyptus increased, while the number of monthly active participants for CloudStack and OpenNebula decreased. Note that even though the number of active participants of CloudStack is less than OpenStack, Figure 4.5 and Figure 4.6 show that the volume of discussion of the two projects are on the same level. This implies that on average the active members in the CloudStack community are communicating more than those of the OpenStack community.

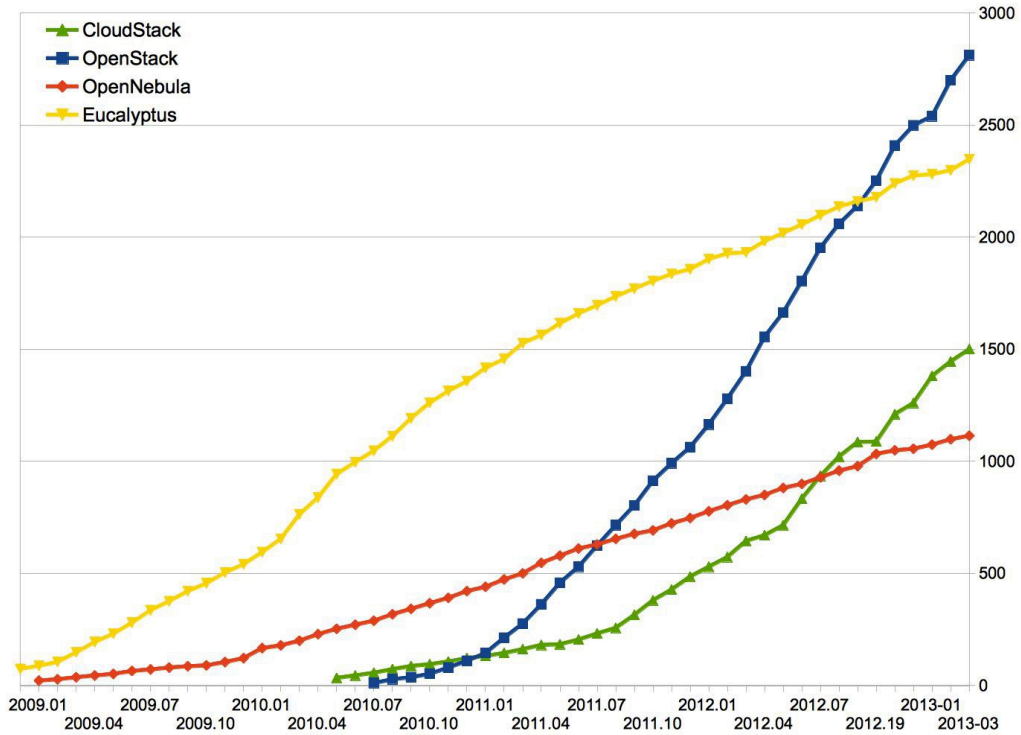

Figure 4.9 Accumulated community population

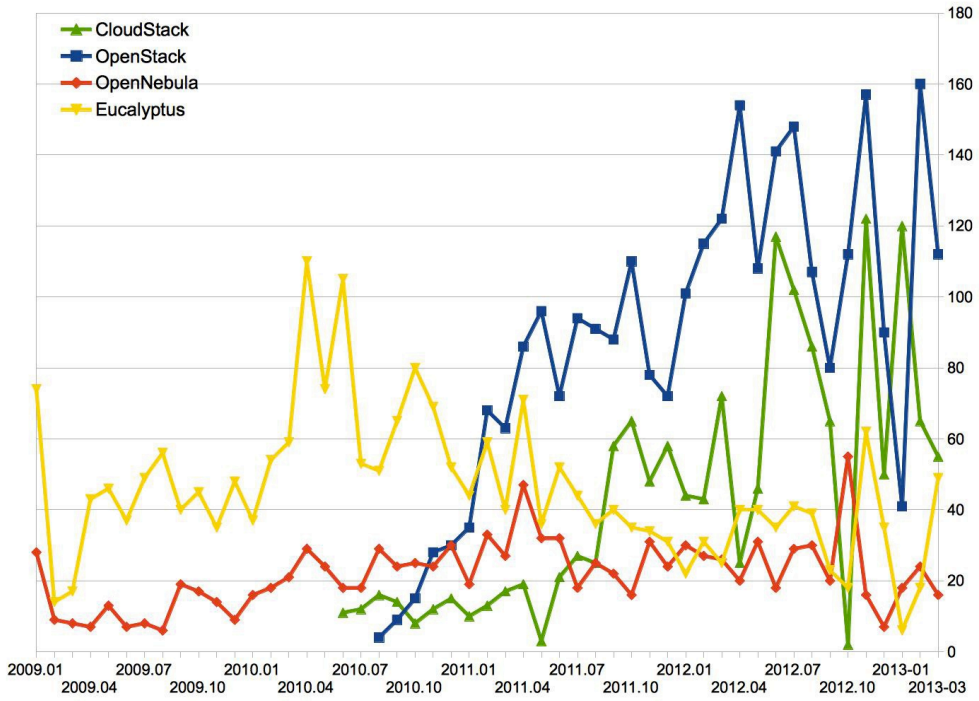

Figure 4.10 Monthly population growth

Figure 4.9 represents the total number of people who have participated in forums or mailing-list discussions. This number is referred to as the accumulated community population. This number does not include inactive users who have registered into discussion forums or mailing lists but have never participated in any open discussions.

According to Figure 4.9, before July 2012 Eucalyptus had the biggest accumulated population among all. From July 2012 to March 2013 OpenStack had the biggest accumulated population and OpenNebula had the smallest accumulated population. It can be seen that by March 2013 the accumulated population of OpenStack is one and half times higher than the accumulated population of OpenNebula.

Figure 4.10 presents the monthly population growth (new members) for four projects. As it can be seen from Figure 4.10, during the first three months of 2013, the population of Eucalyptus has continuously grown, while the population of the other three projects had a growth and then decreased. OpenStack has the highest population growth, followed by CloudStack. The populations of Eucalyptus and OpenNebula are growing at about the same pace. Eucalyptus and OpenNebula have a slow population growth compared to CloudStack and OpenStack. Note the abnormal CloudStack point in 2012-10, this is because the forum data was excluded from the analysis.

Figure 4.11 is a combining Figure 4.8 and Figure 4.10, showing that around 30% of monthly participants OpenStack and OpenNebula are new members, while around 50% of the monthly participants of CloudStack and Eucalyptus are new members. This implies that the OpenStack and OpenNebula communities are more “sticky” than the CloudStack and Eucalyptus communities.

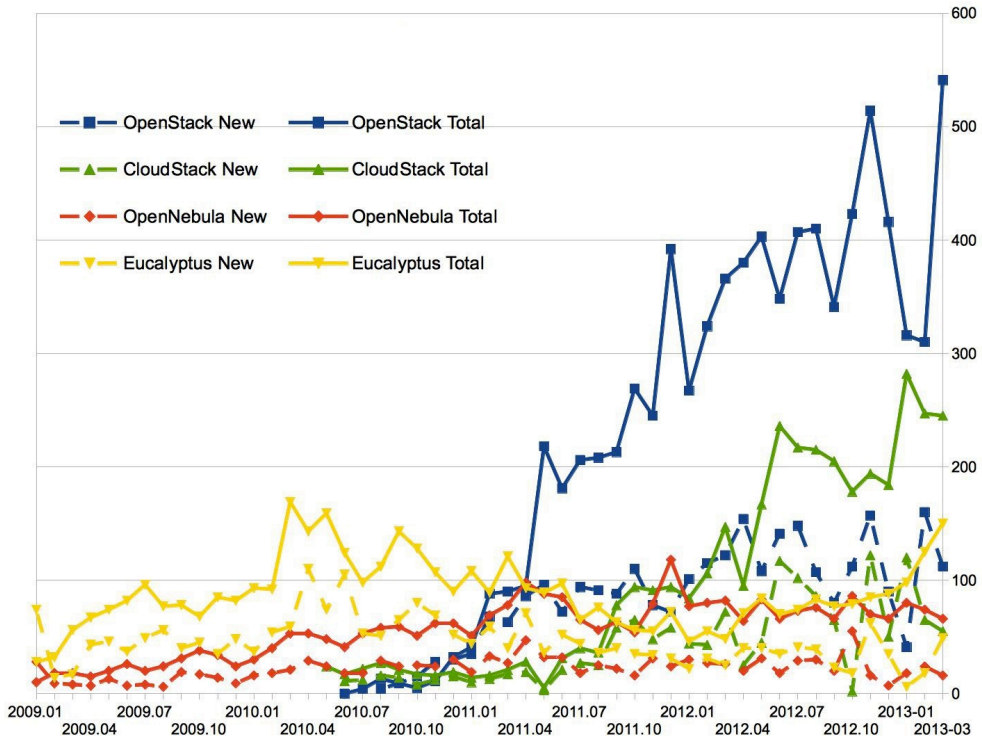

Figure 4.11 Combination of Figure 4.8 and Figure 4.10

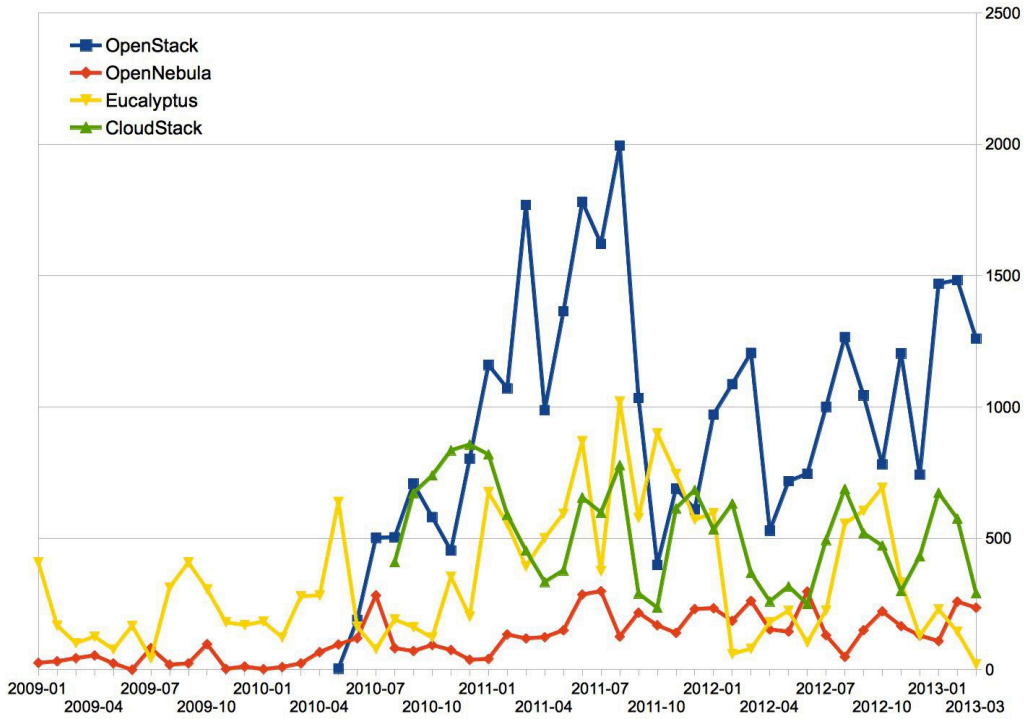

Figure 4.12 Monthly Git commits

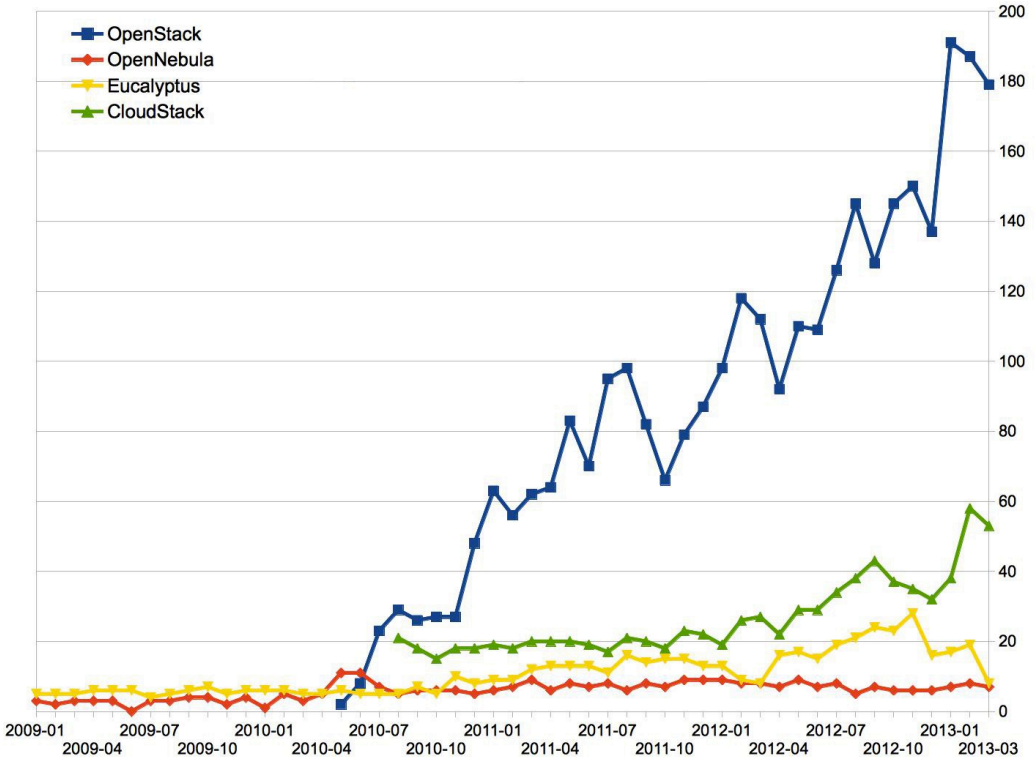

Figure 4.13 Monthly Git contributors

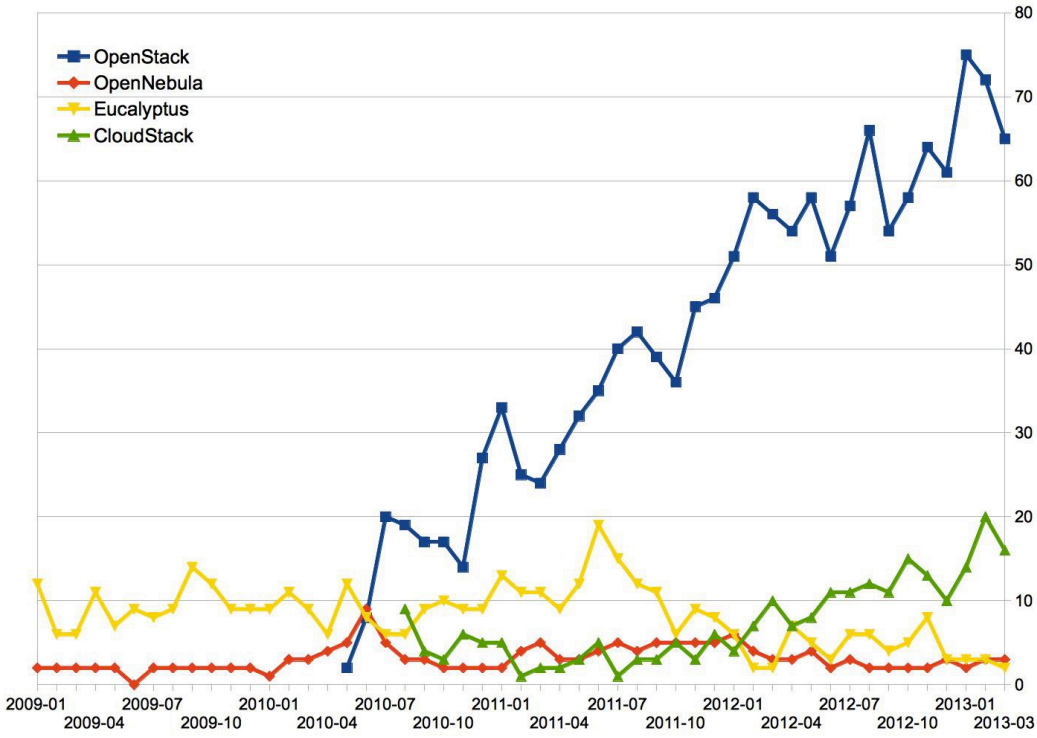

Figure 4.14 Monthly Git domains

All OpenStack, Eucalyptus, CloudStack and OpenNebula projects host their source code on github.com. Github.com [74] provides online hosting service for software development projects that use the Git [73] as their revision control system. Based on some of the data related to the development of four open-source cloud platforms, available form github.com, some basic analyses are carried out in next sections.

Figure 4.12 represents the monthly number of commit operations for four cloud platforms. As it appears in Figure 4.12, the OpenStack project, with an average number of 1000 commits per month and a peak value of 2000 commits in mid 2011, has had the highest commit frequency. The OpenNebula project, with an average of 200 commits per month, has had lowest commit frequency. It can be seen that CloudStack and Eucalyptus quite similar commit frequencies.

Figure 4.13 represents the monthly number of contributors. These contributors are identified by unique github.com accounts for these projects. As Figure 4.13 shows, OpenStack has the highest number of contributors, and the number of contributors are growing quickly. OpenNebula has the lowest number of contributors and it has hardly had any growth in last 6 months. In the last month of 2012 and the first three months of 2013, the number of Eucalyptus contributors had decreased.

Figure 4.14 represents the monthly number of unique institutes that are contributing to the projects. These contributors are identified by the domain name of the contributor's email address. As it appears in Figure 4.14, the number of contributing institutes for OpenStack and CloudStack is growing but OpenStack has the largest number of contributing institutes among all other projects. The number of contributing institutes for OpenNebula is nearly the same while it has had a slight decrease since the beginning of 2012. Between October 2012 and March 2013 the number of contributing institutes for Eucalyptus has decreased.

OpenNebula		**OpenStack**	
Domain	**%**	**Domain**	**%**
opennebula.org	99	review.openstack.org	48
nostalgeek.com	0.6	redhat.com	9
gmail.com	0.2	gmail.com	9
djurdjevic.ch	0.1	linux.vnet.ibm.com	4
c12g.com	0.1	openstack.org	3
		us.ibm.com	3
		hp.com	2
		nicira.com	2
		cloudscaling.com	1
		intel.com	1

Table 4.9 Main contributors to OpenStack and OpenNebula

CloudStack		**Eucalyptus**	
Domain	**%**	**Domain**	**%**
citrix.com	39	eucalyptus.com	85

apache.org	26	gmail.com	13
gmail.com	14	fedoraproject.org	2
cloud.com	6		
bestservers.com	3		
schubergphilis.com	3		
widodh.nl	2		
zonker.net	1		
tcloudcomputing.com	1		
gnsa.us	1		

Table 4.10 Main contributors to CloudStack and Eucalyptus

Table 4.9 and Table 4.10 list the institutes that make the most contributions to OpenNebula, OpenStack, CloudStack and Eucalyptus platforms' development during the first three months of 2013. These numbers are based on the number of commit operations that different institutes contribute with to each project on github.com. According to Table 4.9 and Table 4.10, 99% of the contribution to OpenNebula is made by the OpenNebula institute and 85% of the contribution to Eucalyptus is made by the Eucalyptus institute, while several institutes contribute to the CloudStack and OpenStack projects. Since the majority of OpenStack commits are from a code review system (review.openstack.org) with a single github.com account (jenkins@review.openstack.org) , it makes is difficult to determine the influence of Rackspace on the OpenStack. Also we see redhat.com with 9% of the commits and ibm.com with 7% of the commits are the next major contributors of OpenStack.

Release	Date
OpenStack Grizzly	04.04.2013
OpenStack Folsom	27.09.2012
OpenStack Essex	05.04.2012
OpenStack Diablo	22.09.2011
OpenStack Cactus	15.04.2011
OpenStack Bexar	03.02.2011
OpenStack Austin	21.10.2010

Table 4.11Overview of OpenStack releases

Release	Date
OpenNebula 4.0	08.05.2013
OpenNebula 3.8	22.10.2012
OpenNebula 3.6	09.07.2012
OpenNebula 3.4	10.04.2012
OpenNebula 3.2	17.01.2012
OpenNebula 3.0	03.10.2011
OpenNebula 2.2	28.03.2011
OpenNebula 2.0	24.10.2010
OpenNebula 1.4	16.12.2009
OpenNebula 1.2	06.02.2009
OpenNebula 1.0	24.07.2008

Table 4.12 Overview of OpenNebula releases

Table 4.11 and Table 4.12 provide an overview of the OpenStack platform and the OpenNebula platform releases. From these tables it can be seen that since 2011 OpenStack has had a new release every six months, and Grizzly, the latest release of the OpenStack platform, was published 04.04.2013. While the OpenNebula platform has had a new release every three months in the period of 2011 to 2012, the latest version of OpenNebula 4.0, published 08.05.2013, was released six months after the previous one. This shows the OpenNebula has changed the release time.

4.4.2 Evaluation Based on the Second Study

During the process of writing this thesis several messages were posted into the user communities' mailing list of both the OpenStack platform and the OpenNebula platform. The reaction of both user communities were as following:

- Out of the six messages posted on the OpenStack user community, three of them were replied to within one or two days, the rest remaining unreplied. From the three replies, none of them were useful.
- Out of the nine messages posted on the OpenNebula user community, all were replied to within the timeframe of a couple of hours to one day. Four of the replies were useful.

Although this evidence is too personal and not substantial enough in order to draw any conclusion on the user community of an open-source cloud platform, it can be mentioned that in this thesis' author's experience, the OpenNebula user community appeared more helpful and more responsive than the OpenStack user community.

Chapter 5

5. Discussion and Future Work

This chapter discusses the various aspects of this thesis, in context of evaluation criteria, designing the cloud deployments, outcome of the experiments and future work. Finally based on the outcome of the experiments, answers to the initial problem statements are proposed.

5.1 Evaluation Criteria

Naturally investigating different aspects of a software system would depend on the evaluation criteria defined in the investigation regarding the software system. Evaluating a cloud platform can be a difficult task, depending on the complexity of the framework and amount of criteria to be investigated. It should be noted that even defining the criteria could be a difficult job and requires a high level of expertise. Time and costs also apply as important factors in the quality of evaluating a framework. There were 37 criteria defined in 7 categories related to deployment process, support, usability, compatibility, license, platform complexity and user- and developer community. The evaluation criteria defined for the purpose of this thesis are based on a combination of both generic- and software specific characteristics of the open-source cloud platform. A factor that played an important role in choosing the criteria to be investigated in this project was the time limit. Investigating the chosen criteria had to fit within the time limit dedicated to this project.

The Background chapter presented some previous works related to open-source cloud platform evaluation. Regarding the evaluation criteria investigated in this thesis, it should be noted that although most of the cloud evaluation criteria investigated have not been investigated in previous related works, some of the criteria in previous works are repeated in the criteria of this thesis. This is because the open source cloud platforms are continuously growing projects and the evaluation criteria should be investigated according the latest progress of the cloud platform projects. The following list discusses some of the criteria investigated in both previous works and this thesis:

- An example regarding the necessity of evaluation of license of the cloud platform in the CloudStack was published under GPLv3 license while one year later it was donated to the Apache Software Foundation and published under Apache License 2.
- As the cloud platforms evolve they may add more compatibility with other similar technologies.

- The cloud platforms may provide support for new operating systems.
- Cloud platforms may make some changes in their architecture. For example was OpenStack's current storage component (Cinder) added after the Essex release of OpenStack.

5.2 Designing the Cloud Infrastructure

A cloud infrastructure can be built on several machines or on a single node. Depending on the purpose of the cloud deployment, the cloud infrastructure design may vary from a single node cloud deployment to a multi node cloud deployment. Both OpenStack and OpenNebula cloud platforms provide single node cloud deployment guides for testing purposes. OpenNebula even provides a single node OpenNebula cloud deployment included in a CentOS virtual machine image. For the purpose of basing this thesis on authentic cloud deployment, simple alternatives have been avoided.

The Result chapter presented the process of three separate OpenStack deployments, while only one cloud deployment was carried out for OpenNebula. One of the reasons behind several experiments for OpenStack was to deploy OpenStack with or without some components like the OpenStack networking component (Quantum). Another reason was that each OpenStack deployment experiment provided different results, which showed the differences of the cloud deployment guides.

5.3 Results and Outcomes

In the Result chapter the answers required by the evaluation criteria for both OpenStack and OpenNebula cloud platforms were provided and measured against each other. The comparisons showed that both cloud platforms have advantages and disadvantages. This section discusses the results provided in the Results chapter as a whole picture.

As the process of configuring both cloud platforms showed in the Results chapter, deploying an OpenNebula cloud was more convenient than deploying an OpenStack cloud from several aspects, such as number of packages to be installed, number of files to be configured, amount of time required for the cloud deployment, debugging difficulty level and expertise level. The configuration work required by OpenNebula deployment was much less than the deployment of the OpenStack platform. For example, when configuring databases for OpenNebula only one database was created and the credentials to that database were given to OpenNebula, while each OpenStack component required its own database. It should be considered that when the amount of configuration increases in the deployment process, the risk of misconfiguration increases as well. This is why the automation of the cloud deployment would be very helpful.

We noticed that OpenNebula platform could be deployed on 32-bit operating systems while the OpenStack only provides support for 64-bit operating systems. This gives the OpenNebula users more flexibility of deploying the cloud, where they do not have to change their current hardware and software for deploying the cloud.

Considering the number of books and deployment guides, there were more books and deployment guides for OpenStack than there were for OpenNebula. However, as the deploying of several experiments showed in this thesis, deployment guides had differences in qualities such as readability, clarity of the instructions, being error free and having successful results. Among three deployment guides and one book used to deploy OpenStack cloud, one deployment guide was useful. The existence of various deployment guides and books naturally increases the knowledge about the cloud platform, and they complete each other by revealing each other's mistakes.

As presented in the Results chapter there were third parties that provided support for the OpenStack platform. Some of these third parties were Red Hat, Rackspace, Canonical and SUSE. As explained in the Background chapter, Rackspace is one of the main cofounders of OpenStack and they provide cloud services based on OpenStack. OpenNebula institute provides commercial support for OpenNebula platform.

Regarding the complexity of both OpenNebula and OpenStack platforms, some statistics about the source code of both platforms showed that more than double the amount of source code lines involved in OpenNebula source code are involved in OpenStack platform source code. Naturally, when the size of the source code rises the complexity of the software rises as well. Thus, in this sense the OpenStack platform has higher complexity than that of the OpenNebula platform.

When evaluating the user- and developer community of the cloud platforms, the evaluation was expanded to evaluate CloudStack and Eucalyptus cloud platforms beside the evaluation of OpenStack and OpenNebula cloud platforms. Having the opportunity to compare more cloud platforms provides a better picture when trying to understand the characteristics of the cloud platforms. According to the user- and developer community evaluation results presented in the Result chapter, the OpenStack project has the most growing community among all other cloud projects evaluated, followed by the CloudStack and Eucalyptus platforms. All the criteria investigating the user- and developer communities show that the OpenNebula project has the least growth and development rate compared to other evaluated cloud platform projects. Besides, the general evaluation of the personal experience of the author with the OpenNebula and OpenStack user communities was explained in the result chapter. Based on the author's experience, the OpenNebula community was more responsive and effective than the OpenStack community.

The parameters that evaluate the developer community the for four projects, showed that OpenStack had the highest commit frequency, highest number of contributors and the highest number of contributing institutes among all other projects. The OpenNebula developer community had the lowest of all of these parameters. This means that OpenStack software is developing faster than OpenNebula software. The last part of the analysis of the developer community showed that most of the contribution to OpenNebula and Eucalyptus is done by the OpenNebula institute while several institutes contribute to CloudStack and OpenStack projects. This may indicate that there is a bigger investment in OpenStack and CloudStack than in the other two cloud platforms.

Generally speaking, it seems that open source cloud platforms may be free in terms of buying licenses, but they are still expensive in terms of deployment and maintenance.

5.4 Future Work

As discussed earlier, evaluating cloud software is a heavy task and this thesis covers only a part of it. In the future this thesis can be continued with the following tasks:

- As the OpenStack and OpenNebula projects are growing new features and functionalities are added. Additionally the bugs found in previous releases may be fixed. Thus the cloud platform deployment may provide different results than the results presented in this thesis. Hence evaluating the new releases of the cloud platforms for the same criteria can be done as future work.

- As discussed earlier there exist several more open source cloud platforms on the market. Evaluating all of these cloud platforms could not be fulfilled within the timeframe of this thesis, hence evaluating the rest of open source cloud platform like CloudStack and Eucalyptus would be a natural next step.

- The open source cloud platforms provide support when using different virtualization platforms like Xen, KVM and VMware in cloud infrastructures. How a cloud platform performs under each named virtualization platform, could be done as future work.

- There are other aspects of the cloud platforms that need to be evaluated, for example performance in various system conditions and security of the system. Evaluating cloud platforms from these aspects would also be very important.

Chapter 6

6. Conclusions

The problem statement of this thesis aimed to investigate evaluation criteria in order to make a thorough comparison of cloud computing platforms with respect to the deployment process, activeness of the user community and complexity of the system. Two major open-source private cloud platforms, OpenNebula and OpenStack, were investigated according to all the criteria defined in the thesis. Some of the leading open-source cloud platforms and related technologies were introduced in the Background chapter. In the Methodology chapter, several criteria were defined to evaluate the cloud platforms, including deployment process, usability, as well as user community and developer community aspects.

Each cloud platform was studied carefully using the available resources like official documentation, books, scientific articles and webpages. Different cloud infrastructures were designed to test the deployment of OpenStack and OpenNebula cloud platforms. After preparing each physical cloud infrastructure, the cloud platforms were installed and configured. The cloud deployments were automated as well to evaluate the deployment process against automatability level and provide the techniques useful for future works. When setting up the cloud systems, several attempts were made to debug the systems and for this purpose various system failures were posted on user community of both cloud platforms. A user community and developer community analysis of four major open source cloud platforms was included in this thesis.

According to all cloud evaluation results displayed in the Results chapter and discussed in The Discussion and Future Work chapter, the research question asked in the problem statement are answered as follows:

- For the purpose of this thesis 37 criteria were defined to investigate several aspects of a cloud platform.
- Deploying the OpenNebula cloud platform was easier than deploying the OpenStack cloud platform.

- When evaluating the user community, it showed that the OpenStack user- and developer communities were the most active and growing among the communities of other cloud platform projects.
- The OpenStack cloud platform was found more complex than that of the OpenNebula cloud platform.

The evaluations presented in this thesis revealed some important characteristics of the OpenStack and OpenNebula cloud platforms, which can help the reader of this thesis to determine which cloud platform fits best to their needs.

7. Bibliography

1. *Cloud Computing study for Microsoft shows dramatic reduction in carbon emissions | 5th Nov 2010*. 19.04.2013]; Available from: http://www.wspenvironmental.com/newsroom/news-2/view/cloud-computing-study-for-microsoft-shows-dramatic-reduction-in-carbon-emissions-235.
2. Carr, G. *6 Reasons Why You Shouldn't Believe Everything Proprietary Cloud Vendors Tell You*. 2011 05.04.2013]; Available from: http://www.itproportal.com/2011/12/22/6-reasons-why-you-shouldnt-believe-everything-proprietary-cloud-vendors-tell-you/.
3. Brown, E. *Final Version of NIST Cloud Computing Definition Published*. 2011 20.01.2013]; Available from: http://www.nist.gov/itl/csd/cloud-102511.cfm.
4. Wikipedia. *Cloud computing*. 19.01.2013]; Available from: http://en.wikipedia.org/wiki/Cloud_computing.
5. Cantu, A. *The History and Future of Cloud Computing*. 05.04.2013]; Available from: http://www.forbes.com/sites/dell/2011/12/20/the-history-and-future-of-cloud-computing/.
6. Ammirati, S. *Infographic: Data deluge - 8 zettabytes of Data by 2015*. [cited 05.04.2013; Available from: http://readwrite.com/2011/11/17/infographic-data-deluge---8-ze.
7. Loeffler, B. *What is Infrastructure as a Service*. 21.01.2013]; Available from: http://social.technet.microsoft.com/wiki/contents/articles/4633.what-is-infrastructure-as-a-service.aspx.
8. *What are Service Models in Cloud Computing?* 04.05.13]; Available from: http://www.cloud-competence-center.com/understanding/cloud-computing-service-models/.
9. Wikipedia. *Platform as a service*. 05.04.2013]; Available from: http://en.wikipedia.org/wiki/Platform_as_a_service.
10. Hurwitz, J., *Hybrid Cloud For Dummies*, ed. J. Hurwitz.
11. Nolle, T. *Introducing the key cloud computing platforms*. 28.01.2013]; Available from: http://searchcloudcomputing.techtarget.com/tip/Introducing-the-key-cloud-computing-platforms.
12. Beal, V. *30 Private Cloud Computing Vendors to Consider*. [cited 14.02.2013; Available from: http://www.webopedia.com/DidYouKnow/private-cloud-computing-vendors-to-consider.html.

13. Wikipedia. *OpenStack*. 20.01.2013]; Available from: http://en.wikipedia.org/wiki/OpenStack.
14. OpenStackTeam. *Open source software for building private and public clouds.* 21.01.2013]; Available from: http://www.openstack.org/.
15. OpenStackTeam. *OpenStack Image Service*. 20.01.2013]; Available from: http://www.openstack.org/projects/image-service/.
16. Kavit Munshi, O. *OpenStack keystone identity service*. [cited 20.01.2013; Available from: http://www.slideshare.net/openstackindia/openstack-keystone-identity-service.
17. Kashaba, S. *OpenStack Cloud Storage Services: First Look at Folsom's Cinder project*. [cited 20.01.2013; Available from: http://www.mirantis.com/blog/openstack-cinder-first-look/.
18. *Research and Innovation in Cloud Computing*. 15.04.2013]; Available from: http://dsa-research.org/doku.php.
19. OpenNebula.org. *About the OpenNebula.org Project*. 21.01.2013]; Available from: http://opennebula.org/about:about.
20. Toraldo, G., *OpenNebula 3 Cloud Computing*. First ed2012: Packt Publishing.
21. apache.org. *What is CloudStack?* [cited 22.01.2013; Available from: http://incubator.apache.org/cloudstack/.
22. WikiPedia. *CloudStack*. 20.01.2013]; Available from: http://en.wikipedia.org/wiki/CloudStack.
23. *CloudStack 3.0 Installation Guide*. 12.05.2013]; Available from: http://support.rightscale.com/@api/deki/files/1241/CloudStack3.0InstallGuide.pdf.
24. *CloudStack Official web site*. 12.03.2013]; Available from: http://cloudstack.apache.org/.
25. *CloudStack Reference Architecture*. 12.05. 2013]; Available from: http://support.rightscale.com/09-Clouds/CloudStack/CloudStack_Reference_Architecture.
26. Eucalyptos. *Eucalyptus Systems Extends Lead in AWS-compatible Services from a Private Cloud*. 20.05.2013]; Available from: http://www.eucalyptus.com/news/eucalyptus-systems-extends-lead-aws-compatible-services-private-cloud.
27. Eucalyptos-team. *Eucalyptus FAQ*. 20.05.2013]; Available from: http://www.eucalyptus.com/faq - q2.
28. Eucalyptos-team. *Eucalyptus Components*. 20.05.2013]; Available from: http://www.eucalyptus.com/docs/3.1/ig/euca_components.html.
29. *Eucalyptus (computing)*. [cited 2013 20.05.2013]; Available from: http://en.wikipedia.org/wiki/Eucalyptus_%28computing%29.
30. Eucalyptos-team. *Eucalyptos-Home on GitHub*. 20.05.2013]; Available from: https://github.com/eucalyptus/eucalyptus/wiki.
31. Service, A.w. *Amazon EC2 Instance Types*. 24.01.2013]; Available from: http://aws.amazon.com/ec2/instance-types/.
32. Amazon.com, I. *Amazon Elastic Compute Cloud*. 24.01.2013]; Available from: http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud.
33. Amazon.com, I. *Amazon Elastic Compute Cloud (Amazon EC2)*. 24.01.2013]; Available from: http://aws.amazon.com/ec2/.

34. Wikipedia. *Amazon S3*. 22.01.2013]; Available from: http://en.wikipedia.org/wiki/Amazon_S3.
35. Amazon.com, I. *Amazon Simple Storage Service (Amazon S3)*. 24.01.2013]; Available from: http://aws.amazon.com/s3/.
36. Wikipedia. *Cloud computing comparison*. 10.04.2013]; Available from: http://en.wikipedia.org/wiki/Cloud_computing_comparison.
37. Albertson, L. *Comparing open source private cloud platforms*. 10.04.2013]; Available from: http://www.slideshare.net/OReillyOSCON/comparing-open-source-private-cloud-platforms.
38. *OpenQRM Community Resources*. 20.05.2013]; Available from: http://www.openqrm-enterprise.com.
39. *oVirt*. 20.05.2013]; Available from: http://www.ovirt.org/Home.
40. *OpenShift is Red Hat's free, auto-scaling Platform* 20.05.2013]; Available from: https://http://www.openshift.com/.
41. *Nimbus is cloud computing for science*. 20.05.2013]; Available from: http://www.nimbusproject.org/.
42. *Cloud Foundry*. 20.05.2013]; Available from: http://cloudfoundry.com/.
43. Mahjoub, M., et al. *A Comparative Study of the Current Cloud Computing Technologies and Offers*. in *Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on*. 2011.
44. Sempolinski, P. and D. Thain. *A Comparison and Critique of Eucalyptus, OpenNebula and Nimbus*. in *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*. 2010.
45. von Laszewski, G., et al. *Comparison of Multiple Cloud Frameworks*. in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. 2012.
46. Wind, S. *Open source cloud computing management platforms: Introduction, comparison, and recommendations for implementation*. in *Open Systems (ICOS), 2011 IEEE Conference on*. 2011.
47. Cordeiro, T., et al. *Open Source Cloud Computing Platforms*. in *Grid and Cooperative Computing (GCC), 2010 9th International Conference on*. 2010.
48. Xiaolong, W., et al. *Comparison of open-source cloud management platforms: OpenStack and OpenNebula*. in *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*. 2012.
49. *About Qingye Jiang (John)*. 10.03.2013]; Available from: http://www.qyjohn.net/?page_id=275.
50. qyjohn. *CY12-Q4 Community Analysis — OpenStack vs OpenNebula vs Eucalyptus vs CloudStack*. 2012 23.04.2013]; Available from: http://www.qyjohn.net/?p=2733.
51. qyjohn. *CY13-Q1 Community Analysis — OpenStack vs OpenNebula vs Eucalyptus vs CloudStack*. 2013 23.04.2013]; Available from: http://www.qyjohn.net/?p=3120.
52. Ubuntu.com. *Build your own private cloud*. 15.02.2013]; Available from: http://www.ubuntu.com/cloud/private-cloud.
53. Wheeler, D.A. *How to Evaluate Open Source Software / Free Software (OSS/FS) Programs*. 2011 15.04.2013]; Available from: http://www.dwheeler.com/oss_fs_eval.html.
54. Voras, I., et al. *Evaluating open-source cloud computing solutions*. in *MIPRO, 2011 Proceedings of the 34th International Convention*. 2011.

55. Hauge, O., et al. *An empirical study on selection of Open Source Software - Preliminary results*. in *Emerging Trends in Free/Libre/Open Source Software Research and Development, 2009. FLOSS '09. ICSE Workshop on*. 2009.
56. Voras, I., B. Mihaljevic, and M. Orlic. *Criteria for evaluation of open source cloud computing solutions*. in *Information Technology Interfaces (ITI), Proceedings of the ITI 2011 33rd International Conference on*. 2011.
57. Bitner, S. *10 ways to evaluate your IT management software*. 2009 22.04.2013]; Available from: http://www.zdnet.com/news/10-ways-to-evaluate-your-it-management-software/361560.
58. Esquirol, M. *How to Evaluate Software*. 2010 22.04.2013]; Available from: http://www.softcity.com/article/os-utilities/how-to-evaluate-software/0YTO3MzN.
59. Kelly, B. *Top Tips For Selecting Open Source Software*. 2006 22.04.2013]; Available from: http://www.ukoln.ac.uk/qa-focus/documents/briefings/briefing-60/html/.
60. Silva, C.d. *10 questions to ask when selecting open source products for your enterprise*. 2009 22.04.2013]; Available from: http://www.techrepublic.com/blog/10things/10-questions-to-ask-when-selecting-open-source-products-for-your-enterprise/1232.
61. openstack.com. *How To Get Started With OpenStack*. 17.02.2013]; Available from: http://www.openstack.org/software/start/.
62. Msekni, B. *OpenStack Folsom Install Guide* 02.02.2013]; Available from: https://github.com/mseknibilel/OpenStack-Folsom-Install-guide/blob/master/OpenStack_Folsom_Install_Guide_WebVersion.rst.
63. VanDuyn, Z. *Basic OpenStack Folsom Install Guide*. 17.02.2013]; Available from: http://openstack-folsom-install-guide.readthedocs.org/en/latest/.
64. *OpenStack Basic Installation Guide for Ubuntu 12.04 (LTS)* [cited 22.01.2013; Available from: http://docs.openstack.org/grizzly/basic-install/apt/content.
65. Campbell, K. *Installing OpenStack on Ubuntu 12.04 LTS in 10 Minutes*. 28.01.2013]; Available from: http://www.stackgeek.com/guides/gettingstarted.html.
66. Jackson, K., *OpenStack Cloud Computing Cookbook*. First published: September 2012 ed: Packet.
67. Pepple, K., *Deploying OpenStack*. July 2011 ed: O'Reilly Media.
68. Tom Fifield, D.F., Anne Gentle, Lorin Hochstein, Adam Hyde, Jonathan Proulx, Everett Toews and Joe Topjian, *OpenStack Operations Guid*: First.
69. team, O. *Installing the OpenNebula 4.0*. 16.04.2013]; Available from: http://opennebula.org/documentation:rel4.0:ignc.
70. documentation, U.O. *Ubuntu OpenNebula*. 15.04.2013]; Available from: https://help.ubuntu.com/10.04/serverguide/opennebula.html.
71. CentOS.org. *CentOS - OpenNebula 3.8.1*. [cited 14.05.2013 16.04.2013]; Available from: http://wiki.centos.org/Cloud/OpenNebula/QuickStart.
72. OpenNebula. *OpenNebula Download* 16.04.2013]; Available from: http://opennebula.org/software:software.
73. *Git*. 20.05.2013]; Available from: http://git-scm.com/.
74. *GitHub*. [cited 27.05.2013; Available from: https://github.com/.
75. team, O. *OpenStack Basic Install*. 2013.

8. Appendix

A OpenStack deployment experiment 1

A.1 Controller Node Automation Script

```
#!/bin/bash -e

echo "\nEnter the IP Address for Controller Node"
read IP_ControllerNode
#echo $IP_ControllerNode

if ! [[ "$IP_ControllerNode" =~ (^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}$) ]] ; then
   exec >&2; echo "Bad IP"; exit 1
fi

x=1
echo "--"$(( x++ ))"---------------------------------update----------------
----------------------"
#After you complete the Ubuntu 12.10 installation
apt-get -y update
echo "--"$(( x++ ))"---------------------------------ugrade----------------
----------------------"

apt-get -y upgrade
echo "--"$(( x++ ))"---------------------------------dist-upgrade---------
----------------------------"
apt-get -y dist-upgrade
echo "--"$(( x++ ))"-----------------------NTP-----------------------------
-----------"
#install and configure the NTP service
apt-get  -y install ntp
sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver
127.127.1.0\nfudge 127.127.1.0 stratum 10/g' /etc/ntp.conf
service ntp restart

echo "--"$(( x++ ))"-------------------------MYSQL-----------------------
--------------------"
#install mysql

debconf-set-selections <<< 'mysql-server-5.5.29 mysql-server/root_password
```

```
password Cloud@2013'
debconf-set-selections <<< 'mysql-server-5.5.29 mysql-
server/root_password_again password Cloud@2013'

apt-get -y install mysql-server

apt-get -y install python-mysqldb

echo "--"$(( x++ ))"--------------------------------edit my.conf------------
-----------------------------"
#in this file /etc/mysql/my.cnf comment out the below:
sed -i 's/bind-address/#bind-address/g' /etc/mysql/my.cnf
service mysql restart
echo "--"$(( x++ ))"---------------------------------Creat data bases------
---------------------------------"
mysql -uroot -pCloud@2013 << EOF
DELETE FROM mysql.db WHERE Db LIKE 'test%';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'Cloud@2013';
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystoneUser'@'%' IDENTIFIED BY 'Keystone2013';
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glanceUser'@'%' IDENTIFIED BY 'Glance2013';
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'novaUser'@'%' IDENTIFIED BY 'Nova2013';
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinderUser'@'%' IDENTIFIED BY 'Cinder2013';
EOF

echo "--"$(( x++ ))"----------------------install rabbitmq-server----------
---------------------------------------"
#Install RabbitMQ    very important:
apt-get -y install rabbitmq-server

echo "--"$(( x++ ))"----------------------------install vlan bridge-utils-
---------------------------------------"
#Install VLAN, Bridge-Utils, and setup IP Forwarding
apt-get -y install vlan bridge-utils
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
#run systcl with the updated configuration:
sysctl -p
echo "--"$(( x++ ))"---------------------------------install keystone------
---------------------------------"
#Install the Keystone identity service:

apt-get -y install keystone

sed -i "s/connection =.*/connection =
mysql:\/\/keystoneUser:Keystone2013@$IP_ControllerNode\/keystone/g"
/etc/keystone/keystone.conf
service keystone restart
keystone-manage db_sync

echo "--"$(( x++ ))"----------------------------------dl keystone
Keystone_Scripts-----------------------------------"
wget -P /tmp/ https://raw.github.com/nimbula/OpenStack-Folsom-Install-
guide/master/Keystone_Scripts/With%20Quantum/keystone_basic.sh
wget -P /tmp/ https://raw.github.com/nimbula/OpenStack-Folsom-Install-
guide/master/Keystone_Scripts/With%20Quantum/keystone_endpoints_basic.sh
chmod +x  /tmp/keystone_basic.sh
```

```
chmod +x  /tmp/keystone_endpoints_basic.sh

sed -i "s/HOST_IP=.*/HOST_IP=$IP_ControllerNode/" /tmp/keystone_basic.sh

path="/tmp/keystone_endpoints_basic.sh"
sed -i "s/HOST_IP=.*/HOST_IP=$IP_ControllerNode/g" $path
sed -i "s/EXT_HOST_IP=.*/EXT_HOST_IP=$IP_ControllerNode/g" $path
sed -i "s/MYSQL_PASSWORD=keystonePass/MYSQL_PASSWORD=Keystone2013/g" $path

./keystone_basic.sh
./keystone_endpoints_basic.sh

echo "--"$(( x++ ))"------------------------------make cred----------------
-------------------------"
#Create our OpenStack credential file and load it so we won't be bothered
later:
PATH0="/creds"
echo  -e "export OS_NO_CACHE=1
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL=\"http://$IP_ControllerNode:5000/v2.0/\"" > $PATH0

source $PATH0

echo "--"$(( x++ ))"---------------------install open ssl curl-------------
-------------------------------------"
apt-get -y install curl openssl
# test to see if Keystone is up:
curl http://$IP_ControllerNode:35357/v2.0/endpoints -H 'x-auth-token:
ADMIN' | python -m json.tool

echo "--"$(( x++ ))"----------------------------install glance-------------
------------------------------"
apt-get -y install glance
echo "--"$(( x++ ))"-------------------------------edit glance-api-paste---
-----------------------------------"
for ((  i = 0 ;  i < 1;  i++  ));do sed -i '$d' /etc/glance/glance-api-
paste.ini; done
echo -e "auth_host = $IP_ControllerNode \nauth_port = 35357\nauth_protocol
= http\nadmin_tenant_name = service\nadmin_user = glance\nadmin_password =
service_pass" >> /etc/glance/glance-api-paste.ini

#for ((  i = 0 ;  i < 6;  i++  ));do sed -i '$d' /etc/glance/glance-
registry-paste.ini; done
echo -e "auth_host = $IP_ControllerNode \nauth_port = 35357\nauth_protocol
= http\nadmin_tenant_name = service\nadmin_user = glance\nadmin_password =
service_pass" >> /etc/glance/glance-registry-paste.ini

sed -i "s/.*sql_connection =.*/sql_connection =
mysql:\/\/glanceUser:Glance2013@$IP_ControllerNode\/glance/g"
/etc/glance/glance-api.conf
sed -i 's/.*flavor=.*/flavor = keystone/g' /etc/glance/glance-api.conf

sed -i "s/.*sql_connection =.*/sql_connection =
mysql:\/\/glanceUser:Glance2013@$IP_ControllerNode\/glance/g"
/etc/glance/glance-registry.conf
sed -i 's/.*flavor=.*/flavor = keystone/g' /etc/glance/glance-registry.conf
```

```
service glance-api restart
service glance-registry restart
#Sync databases
glance-manage db_sync

service glance-api restart
service glance-registry restart

echo "--"$(( x++ ))"-----------------------------create image -------------
------------------------------"
#test the Glance installation by installing the cirros cloud image from the
Launchpad mirror
wget -P /tmp/ wget
https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-
disk.img

glance image-create --name NimbulaTest --is-public true --container-format
bare --disk-format qcow2 < /tmp/cirros-0.3.0-x86_64-disk.img

echo "--"$(( x++ ))"-----------------------------List image---------------
---------------------------"
glance image-list

#SetUp Networking
#for ((  i = 0 ;  i < 8;  i++  ));do sed -i '$d'
/etc/network/interfaces;done
#echo -e 'auto br100\niface br100 inet static\n\taddress \n\tnetmask
255.255.255.0\n\tgateway 128.39.73.1 \n\tnetwork 128.39.73.0\n\tbroadcast
128.39.73.255\n\tdns-nameservers 128.39.89.8\n\tbridge_ports
eth0\n\tbridge_stp off\n\tbridge_maxwait 0\n\tbridge_fd 0' >>
/etc/network/interfaces

brctl addbr br100
service networking restart

echo "--"$(( x++ ))"-----------------------------Install NOVA + ...-------
----------------------------------"
#Install and configure Nova¶
apt-get install -y nova-api nova-cert novnc nova-consoleauth nova-scheduler
nova-novncproxy nova-network

PP=$(keystone endpoint-list |grep 9696 |awk '{ print $2; }')

keystone endpoint-delete $PP

PP=$(keystone service-list | grep quantum |awk '{ print $2; }')
keystone service-delete $PP

for ((  i = 0 ;  i < 7;  i++  )) do sed -i '$d' /etc/nova/api-paste.ini;
done
echo -e "auth_host = $IP_ControllerNode\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = nova\nadmin_password =
service_pass\nsigning_dirname = /tmp/keystone-signing-nova" >>
/etc/nova/api-paste.ini

echo "--"$(( x++ ))"-----------------------------edit nova.conf----------
------------------------------"
> /etc/nova/nova.conf
wget -P /tmp/ https://dl.dropbox.com/u/92798871/thesis/openstack/nova.conf
```

```
cat /tmp/nova.conf >> /etc/nova/nova.conf
sed -i "s/CLTIP/$IP_ControllerNode/g" /etc/nova/nova.conf
sed -i 's/IPRANGE/192.168.1.0/g' /etc/nova/nova.conf

nova-manage db sync

cd /etc/init.d/; for i in $(ls nova-*); do sudo service $i restart; done

nova-manage service list

echo "--"$(( x++ ))"----------------------------------install cinder-------
------------------------------"
apt-get -y install cinder-api cinder-scheduler cinder-volume iscsitarget
iscsitarget-dkms

for ((  i = 0 ;  i < 9;  i++  ));do sed -i '$d' /etc/cinder/api-paste.ini;
done
echo -e "service_protocol = http\nservice_host =
$IP_ControllerNode\nservice_port = 5000\nauth_host =
$IP_ControllerNode\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = cinder\nadmin_password =
service_pass" >> /etc/cinder/api-paste.ini

#for ((  i = 0 ;  i < 9;  i++  ));do  sed -i '$d' /etc/cinder/cinder.conf;
done
> /etc/cinder/cinder.conf
echo -e
"[DEFAULT]\nrootwrap_config=/etc/cinder/rootwrap.conf\nsql_connection =
mysql://cinderUser:Cinder2013@$IP_ControllerNode/cinder\napi_paste_confg =
/etc/cinder/api-paste.ini\niscsi_helper=ietadm\nvolume_name_template =
volume-%s\nvolume_group = cinder-volumes\nverbose = True\nauth_strategy =
keystone\n#osapi_volume_listen_port=5900" >> /etc/cinder/cinder.conf

echo "--"$(( x++ ))"----------------------------------make cinder volume---
----------------------------------"
cinder-manage db sync
dd if=/dev/zero of=/root/cinder-volumes bs=1 count=0 seek=2G
losetup /dev/loop2 /root/cinder-volumes

(echo n; echo p; echo 1; echo ; echo ; echo t; echo 8e; echo w) |fdisk
/dev/loop2

pvcreate /dev/loop2
vgcreate cinder-volumes /dev/loop2
sed -i  '/exit 0/i \losetup /dev/loop2 /root/cinder-volumes' /etc/rc.local

echo "--"$(( x++ ))"--------------------install dashboard memchashed-------
--------------------------------------------"
apt-get -y install openstack-dashboard memcached

export DIR1=/etc/openstack-dashboard/local_settings.py
sed -i 's/.*Comment these lines/#Comment these lines/g' $DIR1
sed -i 's/.*Enable the Ubuntu theme if it is present./#Enable the Ubuntu
theme if it is present./g' $DIR1
sed -i 's/.*try:/#try:/g' $DIR1
sed -i 's/.*from ubuntu_theme import */#   from ubuntu_theme import */g'
$DIR1
sed -i 's/.*except ImportError:/#except ImportError:/g' $DIR1
sed -i 's/.*pass/#   pass/g' $DIR1
```

```
#Horizon now requires a reboot to authenticate properly. Reboot and when
the machine is ready, start all of your nova services:
reboot
```

A.2 Compute Node Automation Script

```
#!/bin/bash

echo -e "\nEnter the IP Address for Compute Node"
read IP_Compute
if ! [[ "$IP_Compute" =~ (^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$)
]] ; then
   exec >&2; echo "Bad Compute Node IP"; exit 1
fi
echo -e "\nEnter the IP Address for Controller Node"
read IP_ControllerNode

if ! [[ "$IP_ControllerNode" =~ (^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-
9]{1,3}$) ]] ; then
   exec >&2; echo "Bad Controller Node IP"; exit 1
fi

x=1
echo "--"$(( x++ ))"------------------UPDATE-------------------------------
--------------"
sudo su
apt-get -y update

echo "--"$(( x++ ))"-------------------UPGRADE-----------------------------
---------------"
apt-get -y upgrade

echo "--"$(( x++ ))"-------------------DIST-UPGRADE------------------------
--------------------"
apt-get -y dist-upgrade

echo "--"$(( x++ ))"-----------------------NTP----------------------------
-----------"
#nstall the NTP service:
apt-get -y install ntp
#Configure the NTP server to follow the Controllerler node:
sed -i "s/server ntp.ubuntu.com/server $IP_ControllerNode/" /etc/ntp.conf
service ntp restart

echo "--"$(( x++ ))"------------------------VLAN--------------------------
------------"
#Setup vlan, bridge-utils, and KVM
apt-get -y install vlan bridge-utils

echo "--"$(( x++ ))"-----------------------------edit /etc/sysctl.conf-----
-----------------------------"
#Enable IP_Forwarding
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
#run systcl with the updated configuration:
sysctl -p

echo "--"$(( x++ ))"--------------------------check for KVM SUPPORT--------
-----------------------------"
#Next, check if you can install KVM on your machine:
```

```
apt-get -y install cpu-checker
PS=$(kvm-ok)
if ! [[ "$PS" =~ (^.*KVM acceleration can be used$) ]] ; then exec >&2;
echo -e "NO SUPPORT FOR KVM--Program exits";fi

echo "--"$(( x++ ))"----------------------KVM-----------------------------
-----------"
apt-get -y install kvm libvirt-bin pm-utils

echo "--"$(( x++ ))"-------------------------edit /etc/libvirt/qemu.conf--
-----------------------------------"

sed -i "s/.*cgroup_device_acl =.*/cgroup_device_acl = [/"
/etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/null\", \"\/dev\/full\",.*/\"\/dev\/null\",
\"\/dev\/full\", \"\/dev\/zero",/' /etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/random\",.*/\"\/dev\/random\", \"\/dev\/urandom",/'
/etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/ptmx\",.*/\"\/dev\/ptmx\", \"\/dev\/kvm\",
\"\/dev\/kqemu\",/' /etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/rtc\",.*/\"\/dev\/rtc\", \"\/dev\/hpet\",
\"\/dev\/net\/tun\"/' /etc/libvirt/qemu.conf
sed -i 's/#]/]/' /etc/libvirt/qemu.conf

echo "--"$(( x++ ))"---------------Delete the default virtual bridge-------
-----------------------------"
#Delete the default virtual bridge:
virsh net-destroy default
virsh net-undefine default

echo "--"$(( x++ ))"------------------------edit
/etc/libvirt/libvirtd.conf--------------------------------------"
path="/etc/libvirt/libvirtd.conf"
sed -i "s/.*listen_tls =.*/listen_tls = 0/" $path
sed -i "s/.*listen_tcp =.*/listen_tcp = 1/" $path
sed -i "s/.*auth_tcp =.*/auth_tcp = \"none\"/" $path

sed -i "s/.*env libvirtd_opts.*/env libvirtd_opts=\"-d -l\"/g"
/etc/init/libvirt-bin.conf
sed -i "s/.*libvirtd_opts=.*/libvirtd_opts=\"-d -l\"/g"
/etc/default/libvirt-bin

#Restart the libvirt service to apply the changes:

service libvirt-bin restart

echo "--"$(( x++ ))"--------------------Install and configure nova-network-
--------"
apt-get -y install nova-network

brctl addbr br100
service networking restart

echo "--"$(( x++ ))"-------------------install nova-api-------------------
-----------------------"
#Install and configure nova-api and nova-compute
apt-get -y install nova-api-metadata nova-compute-kvm

for ((  i = 0 ;  i < 7;  i++  )); do sed -i '$d' /etc/nova/api-
```

```
paste.ini;done
echo -e "auth_host = $IP_ControllerNode\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = nova\nadmin_password =
service_pass\nsigning_dirname = /tmp/keystone-signing-nova\n" >>
/etc/nova/api-paste.ini


echo "--"$(( x++ ))"-------------------------edit nova-compute.conf--------
------------------------------"
sed -i "s/.*libvirt_type=.*/libvirt_type=kvm/" /etc/nova/nova-compute.conf

echo "--"$(( x++ ))"---------------------------edit nova.conf-------------
----------------------"
> /etc/nova/nova.conf
wget -P /tmp/
https://dl.dropbox.com/u/92798871/thesis/openstack/novaCompute.conf
cat /tmp/novaCompute.conf >> /etc/nova/nova.conf
sed -i "s/CLTIP/$IP_ControllerNode/g" /etc/nova/nova.conf
sed -i 's/IPRANGE/192.168.1.0/g' /etc/nova/nova.conf
sed -i "s/COMPUTE_IP/$IP_Compute/g" /etc/nova/nova.conf #<----------Chnage
to compute node IP
nova-manage db sync

echo "--"$(( x++ ))"----------------------run nova services---------------
-------------------------"
#Restart services to update changes:
cd /etc/init.d/; for i in $(ls nova-*); do sudo service $i restart; done

echo "--"$(( x++ ))"----------------------list nova services--------------
--------------------------"
nova-manage service list

echo "REMEBER TO CREAT THE network"
echo "nova-manage network create --label=NimbulaNetwork --
fixed_range_v4=10.33.14.0/24 --bridge=br100 --
project_id=<InsertProjectIDHere> --num_networks=1 --multi_host=T"
echo "--"$(( x++ ))"----------------------------finished------------------
---------------"
#ELAPSED_TIME=$(($SECONDS - $START_TIME))
#echo "executed in "$ELAPSED_TIME" seconds"
```

A.3 Nova.conf on Controller Node

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=128.39.73.66
ec2_host=128.39.73.66
ec2_dmz_host=128.39.73.66
rabbit_host=128.39.73.66
cc_host=128.39.73.66
metadata_host=128.39.73.66
metadata_listen=0.0.0.0
nova_url=http://128.39.73.66:8774/v1.1/
sql_connection=mysql://novaUser:Nova2013@128.39.73.66/nova
```

```
ec2_url=http://128.39.73.66:8773/services/Cloud
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://128.39.73.66:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=128.39.73.66:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=128.39.73.66
vncserver_listen=0.0.0.0

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=128.39.73.66
public_interface=br100
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
#Note the different pool, this will be used for instance range
fixed_range=192.168.1.0/24
# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

A.4 Nova.conf on Compute Node

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=128.39.73.66
ec2_host=128.39.73.66
ec2_dmz_host=128.39.73.66
rabbit_host=128.39.73.66
cc_host=128.39.73.66
metadata_host=128.39.73.67
metadata_listen=0.0.0.0
nova_url=http://128.39.73.66:8774/v1.1/
sql_connection=mysql://novaUser:Nova2013@128.39.73.66/nova
ec2_url=http://128.39.73.66:8773/services/Cloud
```

```
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://128.39.73.66:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=128.39.73.66:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=128.39.73.67
vncserver_listen=0.0.0.0

# NETWORK
network_manager=nova.network.manager.FlatDHCPManager
force_dhcp_release=True
dhcpbridge=/usr/bin/nova-dhcpbridge
dhcpbridge_flagfile=/etc/nova/nova.conf
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
# Change my_ip to match each host
my_ip=128.39.73.67
public_interface=br100
vlan_interface=eth0
flat_network_bridge=br100
flat_interface=eth0
#Note the different pool, this will be used for instance range
fixed_range=192.168.1.0/24

# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

A.5 Networking Configuration on Controller Node

```
auto br100
iface br100 inet static
address 128.39.73.66
netmask 255.255.255.0
gateway 128.39.73.1
dns-nameservers 128.39.89.8
dns-search mtv.nimbula.org
bridge_ports eth0
bridge_stp off
bridge_maxwait 0
bridge_fd 0
```

A.6 Network Configuration on Compute Node

```
auto br100
iface br100 inet static
address 128.39.73.67
netmask 255.255.255.0
gateway 128.39.73.1
```

```
dns-nameservers 128.39.89.8
dns-search mtv.nimbula.org
bridge_ports eth0
bridge_stp off
bridge_maxwait 0
bridge_fd 0
```

A.7 Create a Network

```
nova-manage network create --label=NimbulaNetwork --
fixed_range_v4=10.33.14.0/24 --bridge=br100 --
project_id=<InsertProjectIDHere> --num_networks=1 --multi_host=T
```

B OpenStack deployment experiment 2

B.1 Controller Node Automation Script

```
#!/bin/bash -e
x=0
echo "--"$(( x++ ))"---------------------------------START-----------------
----------------------"
echo "--"$(( x++ ))"---------------------------------update----------------
----------------------"
#After you complete the Ubuntu 12.10 installation
apt-get -y update
echo "--"$(( x++ ))"---------------------------------ugrade----------------
----------------------"
apt-get -y upgrade
echo "--"$(( x++ ))"---------------------------------dist-upgrade---------
----------------------------"
apt-get -y dist-upgrade
echo "--"$(( x++ ))"----------------------NTP---Node synchronization------
-------------------------------"
#install and configure the NTP service
apt-get  -y install ntp
sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com\nserver
127.127.1.0\nfudge 127.127.1.0 stratum 10/g' /etc/ntp.conf
service ntp restart
echo "--"$(( x++ ))"-------------------------MYSQL-----------------------
--------------------"
#install mysql

debconf-set-selections <<< 'mysql-server-5.5.29 mysql-server/root_password
password Cloud@2013'
debconf-set-selections <<< 'mysql-server-5.5.29 mysql-
server/root_password_again password Cloud@2013'

apt-get -y install mysql-server
apt-get -y install python-mysqldb
echo "--"$(( x++ ))"-------------------------------edit my.conf------------
-----------------------------"
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart

echo "--"$(( x++ ))"---------------------------------Creat DataBases-------
--------------------------------"
mysql -uroot -pCloud@2013 << EOF
```

```
DELETE FROM mysql.db WHERE Db LIKE 'test%';
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'Cloud@2013';
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystoneUser'@'%' IDENTIFIED BY 'keystonePass';
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glanceUser'@'%' IDENTIFIED BY 'glancePass';
CREATE DATABASE quantum;
GRANT ALL ON quantum.* TO 'quantumUser'@'%' IDENTIFIED BY 'quantumPass';
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'novaUser'@'%' IDENTIFIED BY 'novaPass';
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinderUser'@'%' IDENTIFIED BY 'cinderPass';
DELETE FROM mysql.db WHERE Db LIKE 'test%';
DROP USER ''@'localhost';
EOF

echo "--"$(( x++ ))"----------------------install rabbitmq-server----------
---------------------------------------"
#Install RabbitMQ    very important:
apt-get -y install rabbitmq-server

echo "--"$(( x++ ))"----------------------------install vlan bridge-utils-
-----------------------------------------"
#Install VLAN, Bridge-Utils, and setup IP Forwarding
apt-get -y install vlan bridge-utils
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/g' /etc/sysctl.conf
#run systcl with the updated configuration:
sysctl net.ipv4.ip_forward=1
sysctl -p

echo "--"$(( x++ ))"---------------------------------install keystone------
--------------------------------"
#Install the Keystone identity service:

apt-get -y install keystone

sed -i "s/connection =.*/connection =
mysql:\/\/keystoneUser:keystonePass@192.168.0.1\/keystone/g"
/etc/keystone/keystone.conf
service keystone restart
keystone-manage db_sync

echo "--"$(( x++ ))"---------------------------------Fill up the keystone
database using the two scripts-----------------"
wget https://raw.github.com/mseknibilel/OpenStack-Folsom-Install-
guide/master/Keystone_Scripts/With%20Quantum/keystone_basic.sh
wget https://raw.github.com/mseknibilel/OpenStack-Folsom-Install-
guide/master/Keystone_Scripts/With%20Quantum/keystone_endpoints_basic.sh
chmod +x keystone_basic.sh
chmod +x keystone_endpoints_basic.sh
sed -i "s/HOST_IP=.*/HOST_IP=192.168.0.1/" keystone_basic.sh
sed -i "s/HOST_IP=.*/HOST_IP=192.168.0.1/" keystone_endpoints_basic.sh
sed -i "s/EXT_HOST_IP=.*/EXT_HOST_IP=192.168.0.1/"
keystone_endpoints_basic.sh
#sed -i "s/MYSQL_HOST=.*/MYSQL_HOST=192.168.0.1/"
keystone_endpoints_basic.sh
echo "--"$(( x++ ))"----------------------------run keystone_basic.sh------
---------------------------------------"
./keystone_basic.sh
```

```
./keystone_endpoints_basic.sh
echo "--"$(( x++ ))"----------------------------run
keystone_endpoints_basic.sh-----------------------------------------"

echo 'export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=admin_pass
export OS_AUTH_URL="http://128.39.73.66:5000/v2.0/"' >> /home/behzad/creds

source /home/behzad/creds

#* To test Keystone, we use a simple curl request::
echo "--"$(( x++ ))"---------------------install open ssl curl-----To test
Keystone-----------------------------------------------"
apt-get -y install curl openssl
curl http://128.39.73.66:35357/v2.0/endpoints -H 'x-auth-token: ADMIN' |
python -m json.tool

echo "--"$(( x++ ))"----------------------------install glance---image
storage service a.k.a Glance---------------"
#* After installing Keystone, we continue with installing image storage
service a.k.a Glance::
apt-get -y install glance

echo "--"$(( x++ ))"-------------------------------edit glance-api-paste---
-------------------------------------"
#for ((  i = 0 ;  i < 1;  i++  ));do sed -i '$d' /etc/glance/glance-api-
paste.ini; done
sed -i '$d' /etc/glance/glance-api-paste.ini
echo -e 'auth_host = 192.168.0.1\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = glance\nadmin_password =
service_pass' >> /etc/glance/glance-api-paste.ini

#for ((  i = 0 ;  i < 6;  i++  ));do sed -i '$d' /etc/glance/glance-
registry-paste.ini; done
echo -e 'auth_host = 192.168.0.1\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = glance\nadmin_password =
service_pass' >> /etc/glance/glance-registry-paste.ini

sed -i 's/.*sql_connection =.*/sql_connection =
mysql:\/\/glanceUser:glancePass@192.168.0.1\/glance/g' /etc/glance/glance-
api.conf
sed -i 's/.*flavor=.*/flavor = keystone/g' /etc/glance/glance-api.conf

sed -i 's/.*sql_connection =.*/sql_connection =
mysql:\/\/glanceUser:glancePass@192.168.0.1\/glance/g' /etc/glance/glance-
registry.conf
sed -i 's/.*flavor=.*/flavor = keystone/g' /etc/glance/glance-registry.conf

echo "--"$(( x++ ))"-------------------------------Restart glance-api
glance-registry-----------------------------------------"
service glance-api restart
service glance-registry restart
echo "--"$(( x++ ))"-------------------------------Sync databases----------
-------------------------------------"
#Sync databases
glance-manage db_sync
echo "--"$(( x++ ))"-------------------------------Restart glance-api
glance-registry-----------------------------------------"
```

```
#Restart the services again to take into account the new modifications:
service glance-registry restart
service glance-api restart

echo "--"$(( x++ ))"----------------------------Test Glance installation
download an image and upload it to glance store-----------------"
mkdir /home/behzad/images
#test the Glance installation by installing the cirros cloud image from the
Launchpad mirror
wget -P /home/behzad/images/
https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-
disk.img
glance image-create --name myFirstImage --is-public true --container-format
bare --disk-format qcow2 < /home/behzad/images/cirros-0.3.0-x86_64-disk.img

echo "--"$(( x++ ))"-----------------------------List image--------------
--------------------------"
glance image-list
echo "--"$(( x++ ))"-----------------------------     Quantum       ------
--------------------------------"
echo "--"$(( x++ ))"-----------------------------Install Quantum----------
--------------------------------"
apt-get -y install quantum-server quantum-plugin-openvswitch
sed -i 's/^sql_connection = .*/sql_connection =
mysql:\/\/quantumUser:quantumPass@192.168.0.1\/quantum/g'
/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini
sed -i '
/^\[OVS\]/ a\
tenant_network_type=vlan\nnetwork_vlan_ranges = physnet1:1:4094\n
' /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini
sed -i '/filter:authtoken/{n;N;N;N;N;N;N;d}' /etc/quantum/api-paste.ini
sed -i '
/\[filter:authtoken\]/ a\
paste.filter_factory =
keystone.middleware.auth_token:filter_factory\nauth_host =
192.168.0.1\nauth_port = 35357\nauth_protocol = http\nadmin_tenant_name =
service\nadmin_user = quantum\nadmin_password = service_pass\n
' /etc/quantum/api-paste.ini
echo "--"$(( x++ ))"-----------------------------Install Quantum----------
--------------------------------"
#* Restart the quantum server::
service quantum-server restart

echo "--"$(( x++ ))"-----------------------------Install NOVA + ...-------
-----------------------------------"
#Install and configure Nova¶
#apt-get install -y nova-api nova-cert novnc nova-consoleauth nova-
scheduler nova-novncproxy nova-network
apt-get install -y nova-api nova-cert novnc nova-consoleauth nova-scheduler
nova-novncproxy

for ((  i = 0 ;  i < 7;  i++  )) do sed -i '$d' /etc/nova/api-paste.ini;
done
echo -e 'auth_host = 192.168.0.1\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = nova\nadmin_password =
service_pass\nsigning_dirname = /tmp/keystone-signing-nova' >>
/etc/nova/api-paste.ini

echo "--"$(( x++ ))"-----------------------------edit nova.conf----------
```

```
------------------------------"
wget -P /home/behzad/
https://dl.dropbox.com/u/92798871/thesis/3NodeOpenstackSetup/nova.conf
sed -i 's/192.168.0.3/192.168.0.1/g' ./nova.conf
sed -i 's/128.39.73.69/128.39.73.66/g' ./nova.conf

> /etc/nova/nova.conf
cat /home/behzad/nova.conf >> /etc/nova/nova.conf

echo "--"$(( x++ ))"------------------------------Synchronize your
database----------------------------------------"
nova-manage db sync
echo "--"$(( x++ ))"------------------------------#Restart nova-*
services:---------------------------------------"
#Restart nova-* services:
cd /etc/init.d/; for i in $( ls nova-* ); do sudo service $i restart; done
echo "--"$(( x++ ))"-----------------Check for the smiling faces on nova-*
services to confirm your installation:-----"
nova-manage service list
echo "--"$(( x++ ))"----------------------------------install Cinder-------
------------------------------"
#apt-get -y install cinder-api cinder-scheduler cinder-volume iscsitarget
iscsitarget-dkms
apt-get -y install cinder-api cinder-scheduler cinder-volume iscsitarget
iscsitarget-dkms open-iscsi
echo "--"$(( x++ ))"----------------------------------Configure the iscsi
services:-------------------------------------"
sed -i 's/false/true/g' /etc/default/iscsitarget

echo "--"$(( x++ ))"----------------------------------Restart the services:
iscsitarget-open-iscsi-------------------------"
service iscsitarget start
service open-iscsi start

echo "--"$(( x++ ))"----------------------------------Configure api-
paste.ini-------------------------"
for ((  i = 0 ;  i < 9;  i++  ));do sed -i '$d' /etc/cinder/api-paste.ini;
done
echo -e 'service_protocol = http\nservice_host = 128.39.73.66\nservice_port
= 5000\nauth_host = 192.168.0.1\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = cinder\nadmin_password =
service_pass' >> /etc/cinder/api-paste.ini

echo "--"$(( x++ ))"----------------------------------Configure
cinder.conf-------------------------"
#for ((  i = 0 ;  i < 9;  i++  ));do  sed -i '$d' /etc/cinder/cinder.conf;
done
> /etc/cinder/cinder.conf
echo -e
'[DEFAULT]\nrootwrap_config=/etc/cinder/rootwrap.conf\nsql_connection =
mysql://cinderUser:cinderPass@192.168.0.1/cinder\napi_paste_confg =
/etc/cinder/api-paste.ini\niscsi_helper=ietadm\nvolume_name_template =
volume-%s\nvolume_group = cinder-volumes\nverbose = True\nauth_strategy =
keystone\n#osapi_volume_listen_port=5900' >> /etc/cinder/cinder.conf

echo "--"$(( x++ ))"----------------------------------synchronize your
database-------------------------------------"
cinder-manage db sync
```

```
echo "--"$(( x++ ))"---------------------------------make cinder volume---
----------------------------------"

dd if=/dev/zero of=/home/cinder-volumes bs=1 count=0 seek=2G
losetup /dev/loop2 /home/cinder-volumes

(echo n; echo p; echo 1; echo ; echo ; echo t; echo 8e; echo w) | fdisk
/dev/loop2

echo "--"$(( x++ ))"------------------------- create the physical volume
then the volume group-------------------------------"
pvcreate /dev/loop2
vgcreate cinder-volumes /dev/loop2

echo "--"$(( x++ ))"-----------------------making sure volume group dont
get lost after a system reboot-----------------------------"
sed -i  '/exit 0/i \losetup /dev/loop2 /home/cinder-volumes' /etc/rc.local

echo "--"$(( x++ ))"----------------------------Restart the cinder
services:----------------------------------"
service cinder-volume restart
service cinder-api restart
echo "--"$(( x++ ))"--------------------install Horizon dashboard
memchashed-----------------------------------------------------"
apt-get -y install openstack-dashboard memcached
echo "--"$(( x++ ))"---------------------------------- Disable OpenStack
ubuntu theme-------------------"
path=/etc/openstack-dashboard/local_settings.py
sed -i 's/.*Comment these lines/#Comment these lines/g' $path
sed -i 's/.*Enable the Ubuntu theme if it is present./#Enable the Ubuntu
theme if it is present./g' $path
sed -i 's/.*try:/#try:/g' $path
sed -i 's/.*from ubuntu_theme import */#   from ubuntu_theme import */g'
$path
sed -i 's/.*except ImportError:/#except ImportError:/g' $path
sed -i 's/.*pass/#   pass/g' $path

echo "--"$(( x++ ))"---------------------------Reload Apache and
memcached:----------------------------------"
echo "--"$(( x++ ))"----------You can now access your OpenStack
**192.168.0.1/horizon** with credentials **admin:admin_pass**.-------"
service apache2 restart; service memcached restart

echo -e "\n\n Attention you may need to reboot the system for successful
login"
```

B.2 Network Node Automation Script

```
#!/bin/bash -e
x=1
echo "--"$(( x++ ))"---------------------------------START Network Node----
----------------------------------"
echo "--"$(( x++ ))"---------------------------------update---------------
----------------------"
#After you complete the Ubuntu 12.10 installation
apt-get -y update

echo "--"$(( x++ ))"---------------------------------ugrade---------------
----------------------"
```

```
apt-get -y upgrade

echo "--"$(( x++ ))"----------------------------------dist-upgrade---------
----------------------------"
apt-get -y dist-upgrade

echo "--"$(( x++ ))"----------------------INSTALL NTP---------------------
----------------"
apt-get  -y install ntp

echo "--"$(( x++ ))"--------------------Configure the NTP server to follow
the controller node::-----------------------------------"
sed -i 's/server ntp.ubuntu.com/server 192.168.0.1/g' /etc/ntp.conf
service ntp restart

echo "--"$(( x++ ))"----------------------------install vlan bridge-utils-
-----------------------------------------"
apt-get -y install vlan bridge-utils

echo "--"$(( x++ ))"----------------------------Enable IP forwarding------
-----------------------------------"
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/' /etc/sysctl.conf
sysctl net.ipv4.ip_forward=1
sysctl -p

echo "--"$(( x++ ))"----------------------------Install the openVSwitch--
-----------------------------------"
apt-get install -y openvswitch-switch openvswitch-datapath-dkms

echo "--"$(( x++ ))"-------------------------Create the bridges------------
--------------------------------"
#br-int will be used for VM integration
ovs-vsctl add-br br-int

#br-eth3 will be used for VM configuration
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1

#br-ex is used to make to VM accessible from the internet
ovs-vsctl add-br br-ex
ovs-vsctl add-port br-ex    #<----------------------------Here riseses
problem lossing the internet

echo "--"$(( x++ ))"-------------------------Install the Quantum
openvswitch agent, l3 agent and dhcp agent-----------------------"

apt-get -y install quantum-plugin-openvswitch-agent quantum-dhcp-agent
quantum-l3-agent

echo "--"$(( x++ ))"-------------------------Edit /etc/quantum/api-
paste.ini-----------------------"
sed -i '/filter:authtoken/{n;N;N;N;N;N;N;N;d}' /etc/quantum/api-paste.ini
sed -i '
/\[filter:authtoken\]/ a\
paste.filter_factory =
keystone.middleware.auth_token:filter_factory\nauth_host =
192.168.0.1\nauth_port = 35357\nauth_protocol = http\nadmin_tenant_name =
service\nadmin_user = quantum\nadmin_password = service_pass\n
' /etc/quantum/api-paste.ini
```

```
echo "--"$(( x++ ))"------------------------Edit
/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini--------------------
----"
sed -i 's/^sql_connection = .*/sql_connection =
mysql:\/\/quantumUser:quantumPass@192.168.0.1\/quantum/g'
/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini

sed -i '
/^\[OVS\]/ a\
tenant_network_type=vlan\nnetwork_vlan_ranges =
physnet1:1:4094\nbridge_mappings = physnet1:br-eth1\n
' /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini

echo "--"$(( x++ ))"----------------------------Update
/etc/quantum/l3_agent.ini --------------------------------"
sed -i '/# The Quantum user information /{n;N;N;N;N;N;d}'
/etc/quantum/l3_agent.ini
sed -i '
/# The Quantum user information / a\
auth_url = http://192.168.0.1:35357/v2.0\nauth_region =
RegionOne\nadmin_tenant_name = service\nadmin_user =
quantum\nadmin_password = service_pass\nmetadata_ip =
128.39.73.67\nmetadata_port = 8775\n
'  /etc/quantum/l3_agent.ini

echo "--"$(( x++ ))"----------------* Make sure rabbitMQ IP in
/etc/quantum/quantum.conf is set to the controller node:----------"
sed -i 's/# rabbit_host =.*/rabbit_host = 192.168.0.1/'
/etc/quantum/quantum.conf

echo "--"$(( x++ ))"------------------------ Restart all the services-----
-------------------------------------"
service quantum-plugin-openvswitch-agent restart
service quantum-dhcp-agent restart
service quantum-l3-agent restart
```

B.3 Compute Node Automation Script

```
#!/bin/bash -e
x=0
echo "--"$(( x++ ))"---------------------------------START- Compute Node---
-----------------------------------"
echo "--"$(( x++ ))"---------------------------------update---------------
----------------------"
#After you complete the Ubuntu 12.10 installation
apt-get -y update
echo "--"$(( x++ ))"---------------------------------ugrade---------------
----------------------"
apt-get -y upgrade
echo "--"$(( x++ ))"---------------------------------dist-upgrade---------
---------------------------"
apt-get -y dist-upgrade
echo "--"$(( x++ ))"----------------------INSTALL NTP--------------------
----------------"
#install and configure the NTP service
apt-get  -y install ntp

echo "--"$(( x++ ))"-------------------Configure the NTP server to follow
```

```
the controller node::----------------------------------"
sed -i 's/server ntp.ubuntu.com/server 192.168.0.1/g' /etc/ntp.conf
service ntp restart
echo "--"$(( x++ ))"----------------------------install vlan bridge-utils-
-----------------------------------------"
#Install VLAN, Bridge-Utils, and setup IP Forwarding
apt-get -y install vlan bridge-utils
sed -i 's/#net.ipv4.ip_forward=1/net.ipv4.ip_forward=1/' /etc/sysctl.conf
sysctl net.ipv4.ip_forward=1
sysctl -p

echo "--"$(( x++ ))"-------------------------check for KVM SUPPORT--------
-----------------------------"
#Next, check if you can install KVM on your machine:
apt-get -y install cpu-checker
PS=$(kvm-ok)
if ! [[ "$PS" =~ (^.*KVM acceleration can be used$) ]] ; then exec >&2;
echo "NO SUPPORT FOR KVM";fi
echo "--"$(( x++ ))"----------------------KVM-----------------------------
-----------"

apt-get -y install kvm libvirt-bin pm-utils

echo "--"$(( x++ ))"-------------------------edit /etc/libvirt/qemu.conf--
-----------------------------------"
sed -i "s/.*cgroup_device_acl =.*/cgroup_device_acl = [/"
/etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/null\", \"\/dev\/full\",.*/\"\/dev\/null\",
\"\/dev\/full\", \"\/dev\/zero",/' /etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/random\",.*/\"\/dev\/random\", \"\/dev\/urandom",/'
/etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/ptmx\",.*/\"\/dev\/ptmx\", \"\/dev\/kvm\",
\"\/dev\/kqemu\",/' /etc/libvirt/qemu.conf
sed -i 's/.*\"\/dev\/rtc\",.*/\"\/dev\/rtc\", \"\/dev\/hpet\",
\"\/dev\/net\/tun\"/' /etc/libvirt/qemu.conf
sed -i 's/#]/]/' /etc/libvirt/qemu.conf

echo "--"$(( x++ ))"--------------Delete the default virtual bridge-------
-----------------------------"
#Delete the default virtual bridge:
virsh net-destroy default
virsh net-undefine default

echo "--"$(( x++ ))"----------------------- Enable live migration by
updating /etc/libvirt/libvirtd.conf-----------------"

sed -i "s/.*listen_tls =.*/listen_tls = 0/" /etc/libvirt/libvirtd.conf
sed -i "s/.*listen_tcp =.*/listen_tcp = 1/" /etc/libvirt/libvirtd.conf
sed -i "s/.*auth_tcp =.*/auth_tcp = \"none\"/" /etc/libvirt/libvirtd.conf

sed -i "s/.*env libvirtd_opts.*/env libvirtd_opts=\"-d -l\"/"
/etc/init/libvirt-bin.conf
sed -i "s/.*libvirtd_opts=.*/libvirtd_opts=\"-d -l\"/"
/etc/default/libvirt-bin

echo "--"$(( x++ ))"----------------------- Restart the libvirt service to
apply the changes-----------------"
service libvirt-bin restart
```

```
echo "--"$(( x++ ))"----------------------- Install the openVSwitch:------
-------------------------"
apt-get install -y openvswitch-switch openvswitch-datapath-dkms

echo "--"$(( x++ ))"----------------------- Create the bridges------------
----------------------"
#br-int will be used for VM integration
ovs-vsctl add-br br-int

#br-eth1 will be used for VM configuration
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1

echo "--"$x"----------------------- Quantum ----------------------------
--------------------"
echo "--"$(( x++ ))"----------------------- Quantum openvswitch agent-----
----------------------------"
apt-get -y install quantum-plugin-openvswitch-agent

sed -i 's/^sql_connection = .*/sql_connection =
mysql:\/\/quantumUser:quantumPass@192.168.0.1\/quantum/'
/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini

sed -i '
/^\[OVS\]/ a\
tenant_network_type=vlan\nnetwork_vlan_ranges =
physnet1:1:4094\nbridge_mappings = physnet1:br-eth1\n
' /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini

echo "--"$(( x++ ))"----------------* Make sure rabbitMQ IP in
/etc/quantum/quantum.conf is set to the controller node:----------"
sed -i 's/# rabbit_host =.*/rabbit_host = 192.168.0.1/'
/etc/quantum/quantum.conf

echo "--"$(( x++ ))"----------------------- Restart quantum-plugin-
openvswitch-agent-----------------"
service quantum-plugin-openvswitch-agent restart

echo "--"$(( x++ ))"-------------------install nova-api --- nova's required
components for the compute node::
-----------------------------------------"
apt-get -y install nova-compute-kvm

for ((  i = 0 ;  i < 7;  i++  )); do sed -i '$d' /etc/nova/api-
paste.ini;done
echo -e 'auth_host = 192.168.0.1\nauth_port = 35357\nauth_protocol =
http\nadmin_tenant_name = service\nadmin_user = nova\nadmin_password =
service_pass\nsigning_dirname = /tmp/keystone-signing-nova\n' >>
/etc/nova/api-paste.ini

echo "--"$(( x++ ))"-------------------------NOVA-------------------------
------------"
echo "--"$(( x++ ))"-------------------------edit nova-compute.conf--------
------------------------------"
sed -i "s/.*libvirt_type=.*/libvirt_type=kvm/" /etc/nova/nova-compute.conf

echo "--"$(( x++ ))"---------------------------edit nova.conf--------------
----------------------"
wget -N -P /tmp/
```

```
https://dl.dropbox.com/u/92798871/thesis/3NodeOpenstackSetup/novaCompute.co
nf
> /etc/nova/nova.conf
cat /tmp/novaCompute.conf >> /etc/nova/nova.conf
sed -i 's/100.10.10.51/192.168.0.1/g' /etc/nova/nova.conf
sed -i 's/192.168.100.51/128.39.73.66/g' /etc/nova/nova.conf
sed -i 's/100.10.10.53/192.168.0.3/g' /etc/nova/nova.conf #<----------
Chnage to compute node IP
nova-manage db sync

echo "--"$(( x++ ))"----------------------run nova services---------------
-------------------------"

#Restart services to update changes:
cd /etc/init.d/; for i in $(ls nova-*); do sudo service $i restart; done

echo "--"$(( x++ ))"----------------------list nova services--------------
-------------------------"
nova-manage service list
```

B.4 Nova.conf on Controller Node

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=192.168.0.1
ec2_host=192.168.0.1
ec2_dmz_host=192.168.0.1
rabbit_host=192.168.0.1
dmz_cidr=169.254.169.254/32
metadata_host=192.168.0.1
metadata_listen=0.0.0.0
sql_connection=mysql://novaUser:novaPass@192.168.0.1/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
auth_strategy=keystone
keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
vnc_enabled=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=128.39.73.66
vncserver_listen=0.0.0.0

# Network settings
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
```

```
quantum_admin_password=service_pass
quantum_admin_auth_url=http://192.168.0.1:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver

# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

B.5 Nova.conf on Compute Node

```
[DEFAULT]
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/run/lock/nova
verbose=True
api_paste_config=/etc/nova/api-paste.ini
scheduler_driver=nova.scheduler.simple.SimpleScheduler
s3_host=192.168.0.1
ec2_host=192.168.0.1
ec2_dmz_host=192.168.0.1
rabbit_host=192.168.0.1
dmz_cidr=169.254.169.254/32
metadata_host=192.168.0.1
metadata_listen=0.0.0.0
sql_connection=mysql://novaUser:novaPass@192.168.0.1/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf

# Auth
use_deprecated_auth=false
auth_strategy=keystone
keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
# Imaging service
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# Vnc configuration
novnc_enabled=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
novncproxy_port=6080
vncserver_proxyclient_address=192.168.0.3
vncserver_listen=0.0.0.0

# Network settings
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=service_pass
quantum_admin_auth_url=http://192.168.0.1:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
```

```
# Compute #
compute_driver=libvirt.LibvirtDriver

# Cinder #
volume_api_class=nova.volume.cinder.API
osapi_volume_listen_port=5900
```

B.6 Network Configuration on Controller Node

```
auto eth1
   iface eth1 inet static
   address 128.39.73.66
   netmask 255.255.255.0
   gateway 128.39.73.1
   dns-nameservers 128.39.89.8

auto eth0
   iface eth0 inet static
   address 192.168.0.1
   netmask 255.255.255.0
```

B.7 Cinder.conf File

```
[DEFAULT]
   rootwrap_config=/etc/cinder/rootwrap.conf
   sql_connection = mysql://cinderUser:cinderPass@192.168.0.1/cinder
   api_paste_confg = /etc/cinder/api-paste.ini
   iscsi_helper=ietadm
   volume_name_template = volume-%s
   volume_group = cinder-volumes
   verbose = True
   auth_strategy = keystone
   #osapi_volume_listen_port=5900
```

B.8 Create Volume Group

>dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
>losetup /dev/loop2 cinder-volumes
>fdisk /dev/loop2

```
> dd if=/dev/zero of=cinder-volumes bs=1 count=0 seek=2G
> losetup /dev/loop2 cinder-volumes
> fdisk /dev/loop2
> (echo n; echo p; echo 1; echo ; echo ; echo t; echo 8e; echo w) | fdisk
/dev/loop2
>pvcreate /dev/loop2
>vgcreate /root/cinder-volumes /dev/loop2
```

B.9 Network Configuration on Network Node

```
# VM internet Access
   auto eth2
   iface eth2 inet static
   address 128.39.73.67
   netmask 255.255.255.0
   gateway 128.39.73.1
   dns-nameservers 128.39.89.8
```

```
# OpenStack management
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0

# VM Configuration
auto eth1
iface eth1 inet static
address 10.10.10.1
netmask 255.255.255.0
```

B.10 Network Configuration on Compute Node

```
# OpenStack management
   auto eth0
   iface eth0 inet static
   address 192.168.0.3
   netmask 255.255.255.0

   # VM Configuration
   auto eth1
   iface eth1 inet static
   address 10.10.10.2
   netmask 255.255.255.0
```

B. 11 Configuration of qemu.conf File

```
   cgroup_device_acl = [
   "/dev/null", "/dev/full", "/dev/zero",
   "/dev/random", "/dev/urandom",
   "/dev/ptmx", "/dev/kvm", "/dev/kqemu",
   "/dev/rtc", "/dev/hpet","/dev/net/tun"
   ]
```

B. 12 Configuration of libvirtd.conf File

```
#/etc/libvirt/libvirtd.conf
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"

#/etc/init/libvirt-bin.conf
env libvirtd_opts="-d -l"

#/etc/default/libvirt-bin
libvirtd_opts="-d -l"
```

B. 13 Create Bridges

```
>virsh net-destroy default
>virsh net-undefine default

>ovs-vsctl add-br br-int
>ovs-vsctl add-br br-eth1
>ovs-vsctl add-port br-eth1 eth1
```

B. 14 Configuration of ovs_ quantum_plugin.ini File

```
#Under the database section
[DATABASE]
sql_connection = mysql://quantumUser:quantumPass@128.39.73.66/quantum

#Under the OVS section
[OVS]
tenant_network_type=vlan
network_vlan_ranges = physnet1:1:4094
bridge_mappings = physnet1:br-eth1
```

B. 15 Configuration of api-paste.ini File

```
[filter:authtoken]
   paste.filter_factory = keystone.middleware.auth_token:filter_factory
   auth_host = 128.39.73.66
   auth_port = 35357
   auth_protocol = http
   admin_tenant_name = service
   admin_user = nova
   admin_password = service_pass
   signing_dirname = /tmp/keystone-signing-nova
```

B. 16 Configuration of nova-compute.conf File

```
[DEFAULT]
   libvirt_type=kvm
   libvirt_ovs_bridge=br-int
   libvirt_vif_type=ethernet
   libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
   libvirt_use_virtio_for_bridges=True
```

B. 17 Managing VMs

```
>keystone tenant-create --name project_one
>keystone user-create --name=user_one --pass=user_one --tenant-id
$put_id_of_project_one --email=user_one@domain.com

>keystone user-role-add --tenant-id $put_id_of_project_one  --user-id
$put_id_of_user_one --role-id $put_id_of_member_role

>quantum net-create --tenant-id $put_id_of_project_one net_proj_one --
provider:network_type vlan --provider:physical_network physnet1 --
provider:segmentation_id 1024

>quantum subnet-create --tenant-id $put_id_of_project_one net_proj_one
50.50.1.0/24

>quantum router-create --tenant-id $put_id_of_project_one router_proj_one

>quantum router-interface-add $put_router_proj_one_id_here
$put_subnet_id_here
```

```
>keystone tenant-list
>quantum net-create --tenant-id $put_id_of_service_tenant ext_net --
router:external=True

>quantum subnet-create --tenant-id $put_id_of_service_tenant --allocation-
pool start=128.39.73.68,end=128.39.73.70 --gateway 128.39.73.1 ext_net
128.39.73.0/24 --enable_dhcp=False

>quantum router-gateway-set $put_router_proj_one_id_here
$put_id_of_ext_net_here

>quantum port-list -- --device_id <router_proj_one_id> --device_owner
network:router_gateway

>route add -net 50.50.1.0/24 gw $router_proj_one_IP

>quantum floatingip-create --tenant-id $put_id_of_project_one ext_net

>quantum port-list
>quantum floatingip-associate $put_id_floating_ip $put_id_vm_port
```

C OpenStack deployment experiment 3

C.1 Controller Node Automation Script

```
#!/bin/bash -e
echo -e "\nIs Second hard disk formated : yes/no"
read ASK

while ! [[ "$ASK" =~ (^yes|no$) ]] ; do
echo "answer withyes or no plea";
echo -e "\nIs Second hard disk formated : yes/no"
read ASK
done

x=0
echo "--"$(( x++ ))"---------------------------------START-----------------
----------------------"
apt-get -y update
#OpenStack Essex by default, we are going to use the Ubuntu Cloud Archive
for Folsom :
apt-get install ubuntu-cloud-keyring

echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
updates/folsom main" > /etc/apt/sources.list.d/cloud-archive.list

echo "--"$(( x++ ))"---------------------------------update----------------
----------------------"
#After you complete the Ubuntu 12.10 installation
apt-get -y update && apt-get -y upgrade


sed -i
's/.*net.ipv4.conf.default.rp_filter.*/net.ipv4.conf.default.rp_filter=0/g'
/etc/sysctl.conf
```

```
sed -i 's/.*net.ipv4.conf.all.rp_filter.*/net.ipv4.conf.all.rp_filter=0/g'
/etc/sysctl.conf
service networking restart

sed -i '
/localhost$/ a\
192.168.0.1     folsom-controller\n192.168.0.2     folsom-
network\n192.168.0.3     folsom-compute
' /etc/hosts


echo "--"$(( x++ ))"----------------------NTP---Node synchronization------
------------------------------"
#install and configure the NTP service
apt-get  -y install ntp
#sed -i '/server [0-9].*/d' /etc/ntp.conf
sed -i 's/server ntp.ubuntu.com/server ntp.ubuntu.com iburst\nserver
127.127.1.0\nfudge 127.127.1.0 stratum 10/g' /etc/ntp.conf
service ntp restart

echo "--"$(( x++ ))"-------------------------MYSQL-----------------------
--------------------"
#install mysql

debconf-set-selections <<< 'mysql-server-5.5.29 mysql-server/root_password
password Cloud@2013'
debconf-set-selections <<< 'mysql-server-5.5.29 mysql-
server/root_password_again password Cloud@2013'

apt-get -y install mysql-server

apt-get -y install python-mysqldb

echo "--"$(( x++ ))"-------------------------------edit my.conf------------
-----------------------------"
sed -i 's/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf
service mysql restart

echo "--"$(( x++ ))"---------------------------------Creat DataBases-------
--------------------------------"
mysql -uroot -pCloud@2013 << EOF
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.1' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.2' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.3' IDENTIFIED BY 'password';
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE quantum;
GRANT ALL ON quantum.* TO 'quantum'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON quantum.* TO 'quantum'@'192.168.0.2' IDENTIFIED BY 'password';
GRANT ALL ON quantum.* TO 'quantum'@'192.168.0.3' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
EOF
```

```
echo "--"$(( x++ ))"----------------------install rabbitmq-server----------
----------------------------------------"
#Install RabbitMQ    very important:
apt-get -y install rabbitmq-server
#changes default password
rabbitmqctl change_password guest password


echo "--"$(( x++ ))"---------------------------------install keystone------
---------------------------------"
apt-get -y install keystone python-keystone python-keystoneclient
#apt-get -y install keystone
path="/etc/keystone/keystone.conf"
sed -i 's/.*admin_token =.*/admin_token = password/g' $path
sed -i 's/.*bind_host =.*/bind_host = 0.0.0.0/g' $path
sed -i 's/.*public_port =.*/public_port = 5000/g' $path
sed -i 's/.*admin_port =.*/admin_port = 35357/g' $path
sed -i 's/.*compute_port =.*/compute_port = 8774/g' $path
sed -i 's/.*verbose =.*/verbose = True/g' $path
sed -i 's/.*debug =.*/debug = True/g' $path
sed -i 's/.*connection =.*/connection =
mysql:\/\/keystone:password@localhost:3306\/keystone\nidle_timeout = 200/g'
$path

echo "--"$(( x++ ))"---------------------------Restart Keystone and create
the tables in the database :-----------------------------------------"
service keystone restart
keystone-manage db_sync
echo "--"$(( x++ ))"--------------------------Load environment variables--
----------------------------------------"
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=password

echo 'export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=password' >> novarc
#source ./novarc
echo "source /home/behzad/novarc">>.bashrc
echo "source /home/behzad/novarc">>.profile

echo "--"$(( x++ ))"---------------------------------Fill up the keystone
database using the two scripts-----------------"
wget https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/keystone-data.sh
wget https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/keystone-endpoints.sh

./keystone-data.sh
echo "--"$(( x++ ))"--------------------------run keystone-endpoints.sh--
----------------------------------------"
echo "If an IP address of the management network on the controller node is
different from this example, please use the following:"
```

```
echo "./keystone-endpoints.sh -K <ip address of the management network>"
./keystone-endpoints.sh

echo "--"$(( x++ ))"----------------------------Glance--------------------
-----------------------------------"
apt-get -y install glance glance-api glance-registry python-glanceclient
glance-common

echo "--"$(( x++ ))"-------------------------------edit glance-api-paste---
-----------------------------------"

path="/etc/glance/glance-api.conf"

sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = glance/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path
sed -i 's/.*sql_connection =.*/sql_connection =
mysql:\/\/glance:password@localhost\/glance/g' $path
sed -i 's/.*notifier_strategy =.*/notifier_strategy = rabbit/g' $path
sed -i 's/.*rabbit_password =.*/rabbit_password = password/g' $path

path="/etc/glance/glance-registry.conf"

sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = glance/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path
sed -i 's/.*sql_connection =.*/sql_connection =
mysql:\/\/glance:password@localhost\/glance/g' $path

echo "--"$(( x++ ))"-------------------------------Restart glance-api
glance-registry-----------------------------------------"
service glance-api restart && service glance-registry restart
echo "--"$(( x++ ))"-------------------------------Create Glance tables
into the database :-------------------------------------------------"
#Sync databases
glance-manage db_sync

echo "--"$(( x++ ))"-------------------------------creat an image in to
glance  image store-------------------------------------------------"

wget -P /home/behzad/images/ http://uec-
images.ubuntu.com/releases/12.04/release/ubuntu-12.04-server-cloudimg-
amd64-disk1.img
wget -P /home/behzad/images/
https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-
disk.img
wget -P /home/behzad/images/ http://uec-
images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img
glance image-create --is-public true --disk-format qcow2 --container-format
bare --name "Ubuntu-12.4-server" < /home/behzad/images/ubuntu-12.04-server-
cloudimg-amd64-disk1.img
glance image-create --is-public true --disk-format qcow2 --container-format
bare --name "cirros-0.3.0-x86_64" < /home/behzad/images/cirros-0.3.0-
x86_64-disk.img
glance image-create --is-public true --disk-format qcow2 --container-format
bare --name "precise-server-cloudimg-amd64" < /home/behzad/images/precise-
server-cloudimg-amd64-disk1.img
echo "--"$(( x++ ))"-------------------------------List image-------------
--------------------------"
```

```
glance image-list

echo "--"$(( x++ ))"------------------------------NOVA --------------------
----------------------"
apt-get -y install nova-api nova-cert nova-common nova-scheduler python-
nova python-novaclient nova-consoleauth novnc nova-novncproxy

path="/etc/nova/api-paste.ini"

sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = nova/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path

sed -i '/\[composite:osapi_volume\]/{N;N;N;d}' $path
sed -i '/\[composite:openstack_volume_api_v1\]/{N;N;N;N;d}' $path
sed -i '/.*vol.*/d' $path

echo "--"$(( x++ ))"-------------------------------edit nova.conf----------
------------------------------"
wget -P /tmp/
https://dl.dropbox.com/u/92798871/thesis/openstack/3NodeUbuntu12.04Setup/no
va.conf
cat /tmp/nova.conf > /etc/nova/nova.conf
#
echo "--"$(( x++ ))"-------------------------------Synchronize your
database----------------------------------------"
nova-manage db sync
echo "--"$(( x++ ))"-------------------------------#Restart nova-*
services:---------------------------------------"
#Restart nova-* services:
service nova-api restart
service nova-cert restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-novncproxy restart
echo "--"$(( x++ ))"----------------------------------install Cinder-------
------------------------------"
apt-get install -y cinder-api cinder-scheduler cinder-volume iscsitarget
open-iscsi iscsitarget-dkms python-cinderclient linux-headers-`uname -r`

sed -i 's/include.*/include \/etc\/tgt\/conf.d\/cinder_tgt.conf/'
/etc/tgt/targets.conf

echo "--"$(( x++ ))"----------------------------------Configure the iscsi
services:--------------------------------------"
sed -i 's/false/true/g' /etc/default/iscsitarget

echo "--"$(( x++ ))"----------------------------------Restart the services:
iscsitarget-open-iscsi-------------------------"
service iscsitarget start
service open-iscsi start
echo "--"$(( x++ ))"----------------------------------Configure
cinder.conf------------------------"
cp /etc/cinder/cinder.conf cinder.conf.bk
echo '
sql_connection = mysql://cinder:password@localhost:3306/cinder
rabbit_password = password
'>> /etc/cinder/cinder.conf
```

```
path="/etc/cinder/api-paste.ini"

sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = cinder/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path


echo "--"$(( x++ ))"------------------------- create the physical volume
then the volume group-------------------------------"
if [ $ASK == 'no' ] ; then
(echo n; echo p; echo 1; echo ; echo ; echo t; echo 83; echo w) | fdisk
/dev/sdb
pvcreate /dev/sdb1
vgcreate cinder-volumes /dev/sdb1
fi

echo "--"$(( x++ ))"---------------------------------synchronize your
database--------------------------------------"
cinder-manage db sync

echo "--"$(( x++ ))"---------------------------------make cinder volume---
----------------------------------"

echo "--"$(( x++ ))"----------------------------Restart the cinder
services:---------------------------------"
service cinder-api restart
service cinder-scheduler restart
service cinder-volume restart

echo "--"$(( x++ ))"------------------------------     Quantum       ------
--------------------------------"
echo "--"$(( x++ ))"-----------------------------Install Quantum----------
--------------------------------"
apt-get -y install quantum-server


path="/etc/quantum/quantum.conf"
sed -i 's/.*core_plugin.*/core_plugin =
quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2/g' $path
sed -i 's/.*auth_strategy =.*/auth_strategy = keystone/' $path
sed -i 's/.*fake_rabbit =.*/fake_rabbit = False/g' $path
sed -i 's/.*rabbit_password =.*/rabbit_password = password/g' $path

path="/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini"
sed -i 's/sql_connection =.*/sql_connection =
mysql:\/\/quantum:password@localhost:3306\/quantum/g' $path
sed -i '
/^\[OVS\]/ a\
tenant_network_type = gre\ntunnel_id_ranges = 1:1000\nenable_tunneling =
True\n
' $path

echo "It's more handy to choose tunnel mode since you don't have to
configure your physical switches for VLANs."

path="/etc/quantum/api-paste.ini"
sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = quantum/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path
```

```
echo "--"$(( x++ ))"----------------------Start the quantum services ----
-------------------------------------"
service quantum-server restart

echo "--"$(( x++ ))"-------------------install Horizon dashboard
memchashed-----------------------------------------------------"
#apt-get -y install openstack-dashboard memcached
apt-get -y install apache2 libapache2-mod-wsgi openstack-dashboard
memcached python-memcache
```

C.2 Network Node Automation Script

```
#!/bin/bash -e
x=0
echo '
echo "--"$(( x++ ))"--------------------------------Disable ip version 6 -
-----------------------------------"
echo "#disable ipv6" | sudo tee -a /etc/sysctl.conf
echo "net.ipv6.conf.all.disable_ipv6 = 1" | sudo tee -a /etc/sysctl.conf
echo "net.ipv6.conf.default.disable_ipv6 = 1" | sudo tee -a
/etc/sysctl.conf
echo "net.ipv6.conf.lo.disable_ipv6 = 1" | sudo tee -a /etc/sysctl.conf
'
echo "--"$(( x++ ))"--------------------------------START Network Node----
-----------------------------------"
echo "--"$(( x++ ))"-----------------Since Ubuntu 12.04 LTS has OpenStack
Essex by default, we are going to use Cloud Archive for Folsom-----------
------"
apt-get -y update
apt-get install ubuntu-cloud-keyring

echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
updates/folsom main" > /etc/apt/sources.list.d/cloud-archive.list

echo "--"$(( x++ ))"-----------------------------------dist-upgrade---------
---------------------------"
apt-get -y update && apt-get -y upgrade

echo "--"$(( x++ ))"---------------------------------ugrade---------------
----------------------"
sed -i
's/.*net.ipv4.conf.default.rp_filter=.*/net.ipv4.conf.default.rp_filter=0/g
' /etc/sysctl.conf
sed -i 's/.*net.ipv4.conf.all.rp_filter=.*/net.ipv4.conf.all.rp_filter=0/g'
/etc/sysctl.conf
sed -i 's/.*net.ipv4.ip_forward=.*/net.ipv4.ip_forward=1/g'
/etc/sysctl.conf
service networking restart

sed -i '
/localhost$/ a\
192.168.0.1     folsom-controller\n192.168.0.2     folsom-
network\n192.168.0.3     folsom-compute
' /etc/hosts


echo "--"$(( x++ ))"----------------------INSTALL NTP--------------------
```

```
----------------"
apt-get  -y install ntp

echo "--"$(( x++ ))"--------------------Configure the NTP server to follow
the controller node::-----------------------------------"
sed -i '/server [0-9].*/d' /etc/ntp.conf
sed -i 's/server ntp.ubuntu.com/server 192.168.0.1/g' /etc/ntp.conf
service ntp restart

echo "--"$(( x++ ))"----------------------------Open-vSwitch install
packages----------------------------------------"
apt-get -y install quantum-plugin-openvswitch-agent quantum-dhcp-agent
quantum-l3-agent

echo "--"$(( x++ ))"----------------------------Start Open-vSwitch--------
--------------------------------"
service openvswitch-switch start

echo "--"$(( x++ ))"----------------------------Creat virtual bridges-----
-----------------------------------"
ovs-vsctl add-br br-int
ovs-vsctl add-br br-ex
ovs-vsctl add-port br-ex eth2
ip link set up br-ex

path="/etc/quantum/l3_agent.ini"
sed -i 's/.*auth_url =.*/auth_url = http:\/\/192.168.0.1:35357\/v2.0/g'
$path
sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = quantum/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path
sed -i 's/.*metadata_ip =.*/metadata_ip = 192.168.0.1/g' $path
sed -i 's/.*use_namespaces =.*/use_namespaces = False/g' $path

path="/etc/quantum/api-paste.ini"
sed -i 's/.*auth_host =.*/auth_host = 192.168.0.1/g' $path
sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = quantum/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path

path="/etc/quantum/quantum.conf"
sed -i 's/.*core_plugin.*/core_plugin =
quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2/g' $path
sed -i 's/.*auth_strategy =.*/auth_strategy = keystone/' $path
sed -i 's/.*fake_rabbit =.*/fake_rabbit = False/g' $path
sed -i 's/.*rabbit_host =.*/rabbit_host = 192.168.0.1/g' $path
sed -i 's/.*rabbit_password =.*/rabbit_password = password/g' $path

path="/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini"
sed -i 's/sql_connection =.*/sql_connection =
mysql:\/\/quantum:password@192.168.0.1:3306\/quantum/g' $path
sed -i '
/^\[OVS\]/ a\
tenant_network_type = gre\ntunnel_id_ranges = 1:1000\nenable_tunneling =
True\nintegration_bridge = br-int\ntunnel_bridge = br-tun\nlocal_ip =
10.10.10.1\n
' $path

#echo "It's more handy to choose tunnel mode since you don't have to
```

```
configure your physical switches for VLANs."

path="/etc/quantum/dhcp_agent.ini"
sed -i 's/.*use_namespaces .*/use_namespaces = False/g' $path

echo "--"$(( x++ ))"----------------------------Start services------------
------------------------------"
service quantum-plugin-openvswitch-agent restart
service quantum-dhcp-agent restart
service quantum-l3-agent restart


echo "--"$(( x++ ))"----------------------------Create Virtual Networking-
----------------------------------------"

export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://192.168.0.1:5000/v2.0/"
export SERVICE_ENDPOINT="http://192.168.0.1:35357/v2.0"
export SERVICE_TOKEN=password

echo 'export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://192.168.0.1:5000/v2.0/"
export SERVICE_ENDPOINT="http://192.168.0.1:35357/v2.0"
export SERVICE_TOKEN=password
' > novarc
#source novarc
echo "source novarc">>.bashrc
echo "source novarc">>.profile

echo "--"$(( x++ ))"----------------------------Create Virtual Networking-
----------------------------------------"
#rm /tmp/quantum-networking.*
wget -P /tmp/ https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/quantum-networking.sh
chmod +x /tmp/quantum-networking.sh
path="/tmp/quantum-networking.sh"
sed -i 's/7\.7\.7\.130/128\.39\.73\.68/' $path
sed -i 's/7\.7\.7\.150/128\.39\.73\.70/' $path
sed -i 's/7\.7\.7\.0/128\.39\.73\.0/' $path
sed -i 's/7\.7\.7\.8/128\.39\.73\.67/' $path
sed -i 's/7\.7\.7\.1/128\.39\.73\.1/' $path
sed -i 's/8\.8\.8\.8/128\.39\.89\.8/' $path

/tmp/quantum-networking.sh

# gateway_external_network_id =
path=" /etc/quantum/l3_agent.ini"

gatwayid=$(quantum net-list |grep ext_net |awk '{print $2}')
echo $gatwayid
sed -i "s/.*gateway_external_network_id =.*/gateway_external_network_id =
$gatwayid/" $path

routerid=$(quantum router-list|grep provider-router |awk '{print $2}')
echo $routerid
```

```
sed -i "s/.*router_id =.*/router_id = $routerid/" $path

service quantum-l3-agent restart
```

C.3 Compute Node Automation Script

```
#!/bin/bash -e
x=0
echo "--"$(( x++ ))"---------------------------------START- Compute Node---
-----------------------------------"

echo "--"$(( x++ ))"-----------------Since Ubuntu 12.04 LTS has OpenStack
Essex by default, we are going to use Cloud Archive for Folsom------------
------"
#After you complete the Ubuntu 12.10 installation
#OpenStack Essex by default, we are going to use the Ubuntu Cloud Archive
for Folsom :
apt-get -y update
apt-get install ubuntu-cloud-keyring

echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
updates/folsom main" > /etc/apt/sources.list.d/cloud-archive.list

echo "--"$(( x++ ))"-----------------------------------dist-upgrade---------
----------------------------"
apt-get -y update && apt-get -y upgrade


echo "--"$(( x++ ))"---------------------------------ugrade----------------
-----------------------"

sed -i
's/.*net.ipv4.conf.default.rp_filter=.*/net.ipv4.conf.default.rp_filter=0/g
' /etc/sysctl.conf
sed -i 's/.*net.ipv4.conf.all.rp_filter=.*/net.ipv4.conf.all.rp_filter=0/g'
/etc/sysctl.conf
service networking restart

sed -i '
/localhost$/ a\
192.168.0.1     folsom-controller\n192.168.0.2     folsom-
network\n192.168.0.3     folsom-compute
' /etc/hosts
echo "folsom-compute" > /etc/hostname

echo "--"$(( x++ ))"----------------------INSTALL NTP--------------------
----------------"
apt-get  -y install ntp

echo "--"$(( x++ ))"--------------------Configure the NTP server to follow
the controller node::-----------------------------------"
sed -i '/server [0-9].*/d' /etc/ntp.conf
sed -i 's/server ntp.ubuntu.com/server 192.168.0.1/g' /etc/ntp.conf
service ntp restart


echo "--"$(( x++ ))"-------------------------check for KVM SUPPORT--------
----------------------------"
Next, check if you can install KVM on your machine:
```

```
apt-get -y install cpu-checker
PS=$(kvm-ok)
if ! [[ "$PS" =~ (^.*KVM acceleration can be used$) ]] ; then exec >&2;
echo "NO SUPPORT FOR KVM";fi

echo "--"$(( x++ ))"----------------------Hypervisor----------------------
-----------------"
apt-get install -y kvm libvirt-bin pm-utils

echo "--"$(( x++ ))"--------------------------edit /etc/libvirt/qemu.conf--
-----------------------------------"
path="/etc/libvirt/qemu.conf"
sed -i "s/.*cgroup_device_acl =.*/cgroup_device_acl = [/" $path
sed -i 's/.*\"\/dev\/null\", \"\/dev\/full\",.*/\"\/dev\/null\",
\"\/dev\/full\", \"\/dev\/zero",/' $path
sed -i 's/.*\"\/dev\/random\",.*/\"\/dev\/random\", \"\/dev\/urandom",/'
$path
sed -i 's/.*\"\/dev\/ptmx\",.*/\"\/dev\/ptmx\", \"\/dev\/kvm\",
\"\/dev\/kqemu\",/' $path
sed -i 's/.*\"\/dev\/rtc\",.*/\"\/dev\/rtc\", \"\/dev\/hpet\",
\"\/dev\/net\/tun\"/' $path
sed -i 's/#]/]/' $path

echo "--"$(( x++ ))"--------------Delete the default virtual bridge-------
-----------------------------"
#Delete the default virtual bridge:
virsh net-destroy default
virsh net-undefine default

echo "--"$(( x++ ))"----------------------- Enable live migration by
updating /etc/libvirt/libvirtd.conf-----------------"

path="/etc/libvirt/libvirtd.conf"
sed -i "s/.*listen_tls =.*/listen_tls = 0/" $path
sed -i "s/.*listen_tcp =.*/listen_tcp = 1/" $path
sed -i "s/.*auth_tcp =.*/auth_tcp = \"none\"/" $path

path=" /etc/init/libvirt-bin.conf"
sed -i "s/.*env libvirtd_opts.*/env libvirtd_opts=\"-d -l\"/" $path

path="/etc/default/libvirt-bin"
sed -i "s/.*libvirtd_opts=.*/libvirtd_opts=\"-d -l\"/" $path

echo "--"$(( x++ ))"----------------------- Restart the libvirt service to
apply the changes-----------------"
service libvirt-bin restart
echo "--"$(( x++ ))"-----------------NOVA---------------------------------
-----"
apt-get -y install nova-compute-kvm

path="/etc/nova/api-paste.ini"
sed -i 's/.*auth_host =.*/auth_host = 192.168.0.1/g' $path
sed -i 's/.*admin_tenant_name =.*/admin_tenant_name = service/g' $path
sed -i 's/.*admin_user =.*/admin_user = nova/g' $path
sed -i 's/.*admin_password =.*/admin_password = password/g' $path

path="/etc/nova/nova-compute.conf"
echo "libvirt_ovs_bridge=br-int
libvirt_vif_type=ethernet
```

```
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
libvirt_use_virtio_for_bridges=True" >> $path


echo "--"$(( x++ ))"------------------------------edit nova.conf----------
------------------------------"
wget -P /tmp/
https://dl.dropbox.com/u/92798871/thesis/openstack/3NodeUbuntu12.04Setup/no
va.conf.compute
cat /tmp/nova.conf.compute > /etc/nova/nova.conf
service nova-compute restart
echo "--"$x"----------------------- Quantum ----------------------------
--------------------"
echo "--"$(( x++ ))"----------------------- Quantum openvswitch ----------
-----------------------"
#apt-get -y install quantum-plugin-openvswitch-agent
apt-get install -y openvswitch-switch
service openvswitch-switch start

echo "--"$(( x++ ))"-----------------------Configure Virtual Bridging-----
----------------------------"
ovs-vsctl add-br br-int

echo "--"$(( x++ ))"----------------------- Quantum openvswitch ----------
-----------------------"
apt-get -y install quantum-plugin-openvswitch-agent


path="/etc/quantum/quantum.conf"
sed -i 's/.*core_plugin.*/core_plugin =
quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2/g' $path
sed -i 's/.*auth_strategy =.*/auth_strategy = keystone/' $path
sed -i 's/.*fake_rabbit =.*/fake_rabbit = False/g' $path
sed -i 's/.*rabbit_host =.*/rabbit_host = 192.168.0.1/g' $path
sed -i 's/.*rabbit_password =.*/rabbit_password = password/g' $path

path="/etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini"
sed -i 's/sql_connection =.*/sql_connection =
mysql:\/\/quantum:password@192.168.0.1:3306\/quantum/g' $path
sed -i '
/^\[OVS\]/ a\
tenant_network_type = gre\ntunnel_id_ranges = 1:1000\nenable_tunneling =
True\nintegration_bridge = br-int\ntunnel_bridge = br-tun\nlocal_ip =
10.10.10.2\n
' $path

echo "--"$(( x++ ))"----------------------- Start the Agent--------------
---------------"
service quantum-plugin-openvswitch-agent restart

#nova-manage service list
echo '#!/bin/bash'> restartSer.sh
cat 3Node-ComputeNode.sh |grep start >> restartSer.sh
chmod +x restartSer.sh
```

C.4 Nova.conf on Controller Node

```
[DEFAULT]
# MySQL Connection #
sql_connection=mysql://nova:password@192.168.0.1/nova
```

```
# nova-scheduler #
rabbit_password=password
scheduler_driver=nova.scheduler.simple.SimpleScheduler

# nova-api #
cc_host=192.168.0.1
auth_strategy=keystone
s3_host=192.168.0.1
ec2_host=192.168.0.1
nova_url=http://192.168.0.1:8774/v1.1/
ec2_url=http://192.168.0.1:8773/services/Cloud
keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
api_paste_config=/etc/nova/api-paste.ini
allow_admin_api=true
use_deprecated_auth=false
ec2_private_dns_show_ip=True
dmz_cidr=169.254.169.254/32
ec2_dmz_host=192.168.0.1
metadata_host=192.168.0.1
metadata_listen=0.0.0.0
enabled_apis=ec2,osapi_compute,metadata

# Networking #
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.0.1:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver


# Cinder #
volume_api_class=nova.volume.cinder.API

# Glance #
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# novnc #
novnc_enable=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
vncserver_proxyclient_address=192.168.0.1
vncserver_listen=0.0.0.0

# Misc #
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=true
```

C.5 Nova.conf on Compute Node

```
[DEFAULT]
```

```
# MySQL Connection #
sql_connection=mysql://nova:password@192.168.0.1/nova

# nova-scheduler #
rabbit_host=192.168.0.1
rabbit_password=password
scheduler_driver=nova.scheduler.simple.SimpleScheduler

# nova-api #
cc_host=192.168.0.1
auth_strategy=keystone
s3_host=192.168.0.1
ec2_host=192.168.0.1
nova_url=http://192.168.0.1:8774/v1.1/
ec2_url=http://192.168.0.1:8773/services/Cloud
keystone_ec2_url=http://192.168.0.1:5000/v2.0/ec2tokens
api_paste_config=/etc/nova/api-paste.ini
allow_admin_api=true
use_deprecated_auth=false
ec2_private_dns_show_ip=True
dmz_cidr=169.254.169.254/32
ec2_dmz_host=192.168.0.1
metadata_host=192.168.0.1
metadata_listen=0.0.0.0
enabled_apis=metadata

# Networking #
network_api_class=nova.network.quantumv2.api.API
quantum_url=http://192.168.0.1:9696
quantum_auth_strategy=keystone
quantum_admin_tenant_name=service
quantum_admin_username=quantum
quantum_admin_password=password
quantum_admin_auth_url=http://192.168.0.1:35357/v2.0
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
linuxnet_interface_driver=nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver

# Compute #
compute_driver=libvirt.LibvirtDriver
connection_type=libvirt

# Cinder #
volume_api_class=nova.volume.cinder.API

# Glance #
glance_api_servers=192.168.0.1:9292
image_service=nova.image.glance.GlanceImageService

# novnc #
novnc_enable=true
novncproxy_base_url=http://128.39.73.66:6080/vnc_auto.html
vncserver_proxyclient_address=192.168.0.3
vncserver_listen=0.0.0.0

# Misc #
logdir=/var/log/nova
state_path=/var/lib/nova
lock_path=/var/lock/nova
```

```
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=true
```

C.6 Install Ubuntu Cloud Archive

```
>apt-get install ubuntu-cloud-keyring
> echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-
updates/folsom main" > /etc/apt/sources.list.d/cloud-archive.list
>apt-get update && apt-get upgrade
```

C.7 Network Configuration on Controller Node

```
# Management Network
    auto eth0
        iface eth0 inet static
        address 192.168.0.1
        netmask 255.255.255.0

    # API + Public Network
        auto eth1
        iface eth1 inet static
        address 128.39.73.66
        netmask 255.255.255.0
        gateway 128.39.73.1
        dns-nameservers 128.39.89.8
```

C.8 Creating Databases

```
>mysql -uroot -pCloud@2013 << EOF
CREATE DATABASE nova;
GRANT ALL ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.1' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.2' IDENTIFIED BY 'password';
GRANT ALL ON nova.* TO 'nova'@'192.168.0.3' IDENTIFIED BY 'password';
CREATE DATABASE cinder;
GRANT ALL ON cinder.* TO 'cinder'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE glance;
GRANT ALL ON glance.* TO 'glance'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE keystone;
GRANT ALL ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE quantum;
GRANT ALL ON quantum.* TO 'quantum'@'localhost' IDENTIFIED BY 'password';
GRANT ALL ON quantum.* TO 'quantum'@'192.168.0.2' IDENTIFIED BY 'password';
GRANT ALL ON quantum.* TO 'quantum'@'192.168.0.3' IDENTIFIED BY 'password';
FLUSH PRIVILEGES;
EOF
```

C.9 Configuration of keystone.conf File

```
    [DEFAULT]
    admin_token = password
    bind_host = 0.0.0.0
    public_port = 5000
    admin_port = 35357
    compute_port = 8774
    verbose = True
    debug = True
    log_file = keystone.log
    log_dir = /var/log/keystone
    log_config = /etc/keystone/logging.conf
```

```
[sql]
connection = mysql://keystone:password@localhost:3306/keystone
idle_timeout = 200
```

C.10 Setting Enviromnet Variables

```
>echo 'export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://localhost:5000/v2.0/"
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
export SERVICE_TOKEN=password' >> novarc
>source novarc
>echo "source novarc">>.bashrc
```

C.11 Install Ubuntu Cloud archive

```
https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/keystone-data.sh
https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/keystone-endpoints.sh
```

C.12 Configuaration of glance-registry.conf File

```
In /etc/glance/glance-api.conf

sql_connection = mysql://glance:password@localhost/glance
admin_tenant_name = service
admin_user = glance
admin_password = password

In /etc/glance/glance-registry.conf

sql_connection = mysql://glance:password@localhost/glance
admin_tenant_name = service
admin_user = glance
admin_password = password
notifier_strategy = rabbit
rabbit_password = password
```

C.13 Add Image to Glance Image Store

```
>service glance-api restart && service glance-registry restart
>glance-manage db_sync
Add an image to Glance image store
> wget -P /tmp/ https://launchpad.net/cirros/trunk/0.3.0/+download/cirros-
0.3.0-x86_64-disk.img
>glance image-create --name myFirstImage --is-public true --container-
format bare --disk-format qcow2 < /tmp/cirros-0.3.0-x86_64-disk.img
>glance image-list
```

C.14 Install and Configure Nova

```
>apt-get –y install nova-api nova-cert nova-common nova-scheduler python-
nova python-novaclient nova-consoleauth novnc nova-novncproxy

#In /etc/nova/api-paste.ini file update:
admin_tenant_name = service
admin_user = nova
admin_password = password
```

#remove part concerning "nova-volume" from /etc/nova/api-paste.ini

```
>path="/etc/nova/api-paste.ini"
>sed -i '/\[composite:osapi_volume\]/{N;N;N;d}' $path
>sed -i '/\[composite:openstack_volume_api_v1\]/{N;N;N;N;d}' $path
>sed -i '/.*vol.*/d' $path
```

Synchronize with nova database

```
>nova-manage db sync
#Resrat nova services
>service nova-api restart
>service nova-cert restart
>service nova-consoleauth restart
>service nova-scheduler restart
>service nova-novncproxy restart
```

C.15 Install and configure Cinder

```
>apt-get install -y cinder-api cinder-scheduler cinder-volume iscsitarget
open-iscsi iscsitarget-dkms python-cinderclient linux-headers-`uname -r`
```

#Configure & start the iSCSI services :

```
>sed -i 's/false/true/g' /etc/default/iscsitarget
>service iscsitarget start
>service open-iscsi start
```

In */etc/cinder/cinder.conf* file update the following variables:

```
[DEFAULT]
sql_connection = mysql://cinder:password@localhost:3306/cinder
rabbit_password = password
```

In */etc/cinder/api-paste.ini* file update tenant authentication variables:

admin_tenant_name = service
admin_user = cinder
admin_password = password

#Format the second hard disk and create a linux partition on it.

```
>(echo n; echo p; echo 1; echo ; echo ; echo t; echo 83; echo w) | fdisk
/dev/sdb
```

#Create volume group names cinder-volumes on second hard disk.

```
>pvcreate /dev/sdb1
>vgcreate cinder-volumes /dev/sdb1
```

#Synchronize Cinder with data base to create tables and restart the service

```
>cinder-manage db sync
>service cinder-api restart
>service cinder-scheduler restart
>service cinder-volume restart
```

C.16 Install and Configure Quantum

#install Quantum

```
>apt-get install quantum-server

#In /etc/quantum/quantum.conf file update following variables.

core_plugin = \
 quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_password = password

#In /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini file update the following variables.

[DATABASE]
sql_connection = mysql://quantum:password@localhost:3306/quantum

[OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True

#In /etc/quantum/api-paste.ini file modify the following variables:

admin_tenant_name = service
admin_user = quantum
admin_password = password

#And finally restart the services:
>service quantum-server restart
```

C. 17 Install Apache web server

```
>apt-get install apache2 libapache2-mod-wsgi openstack-dashboard memcached
python-memcache
```

C. 18 Network configuration on Network node

```
# Management Network
    auto eth0
        iface eth0 inet static
        address 192.168.0.2
        netmask 255.255.255.0
        gateway 192.168.0.254
        dns-nameservers 8.8.8.8

    # Data Network
        auto eth1
        iface eth1 inet static
        address 10.10.10.1
        netmask 255.255.255.0

    # Public Bridge
        auto eth2
        iface eth2 inet manual
        up ifconfig $IFACE 0.0.0.0 up
        up ip link set $IFACE promisc on
        down ifconfig $IFACE down
```

C.19 Install and Configure NTP Server

```
>apt-get install -y ntp

#Configure /etc/ntp.conf file with following
>sed -i '/server [0-9].*/d' /etc/ntp.conf
>sed -i 's/server ntp.ubuntu.com/server 192.168.0.1/g' /etc/ntp.conf

Restart NTP server.
>service ntp restart
```

C.20 Network Node Configuration

```
>apt-get –y install quantum-plugin-openvswitch-agent quantum-dhcp-agent
quantum-l3-agent

#Start Open vSwitch by:
>service openvswitch-switch start

# add bridges
>ovs-vsctl add-br br-int
>ovs-vsctl add-br br-ex
>ovs-vsctl add-port br-ex eth2
>ip link set up br-ex

#In /etc/quantum/l3_agent.ini file update the following variables:

auth_url = http://192.168.0.1:35357/v2.0
admin_tenant_name = service
admin_user = quantum
admin_password = password
metadata_ip = 192.168.0.1
use_namespaces = False

#In /etc/quantum/api-paste.ini file modify the following variables:
auth_host = 192.168.0.1
admin_tenant_name = service
admin_user = quantum
admin_password = password

#Update /etc/quantum/quantum.conf file with following variables:

core_plugin = \
        quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_host = 192.168.0.1
rabbit_password = password

#In /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini file update:

[DATABASE]
sql_connection = mysql://quantum:password@192.168.0.1:3306/quantum
[OVS]
```

```
tenant_network_type = gre
tunnel_id_ranges = 1:1000
enable_tunneling = True
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.10.10.1
```

#Edit */etc/quantum/dhcp_agent.ini* file and modify the following variable:

```
use_namespaces = False
```

#Finally restart all services to update with new configuration changes:

```
>service quantum-plugin-openvswitch-agent start
>service quantum-dhcp-agent restart
>service quantum-l3-agent restart
```

C.21 Set Environmental Variables

```
>echo 'export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=password
export OS_AUTH_URL="http://192.168.0.1:5000/v2.0/"
export SERVICE_ENDPOINT="http://192.168.0.1:35357/v2.0"
export SERVICE_TOKEN=password
' > novarc
source novarc
>echo "source novarc">>.bashrc
>echo "source novarc">>.profile
```

C.22 Create Network

```
>wget -P /tmp/ https://raw.github.com/EmilienM/openstack-folsom-
guide/master/scripts/quantum-networking.sh

>chmod +x /tmp/quantum-networking.sh
```

#Modify */tmp/quantum-networking.sh* to suit our networking run it.

```
>/tmp/quantum-networking.sh
```

#Copy the external network ID

```
>quantum net-list
```

#Edit /etc/quantum/l3_agent.ini and paste the ID to the following variable:

```
gateway_external_network_id = ID
```

#Get provider router ID from following coommand:

```
>quantum router-list
```

#In same file /etc/quantum/l3_agent.ini and paste the router ID.

```
router_id = ID
```

#Finally restart L3 Agent.

```
>service quantum-l3-agent restart
```

C.23 Network configuration on Compute Node

```
# Management Network
    auto eth0
        iface eth0 inet static
        address 192.168.0.3
        netmask 255.255.255.0
        gateway 192.168.0.254
        dns-nameservers 8.8.8.8

    # Data Network
        auto eth1
        iface eth1 inet static
        address 10.10.10.2
        netmask 255.255.255.0
```

C.24 Prepare Hypervisor on Compute Node

```
>apt-get install -y kvm libvirt-bin pm-utils
```

#In */etc/libvirt/qemu.conf* file uncomment *cgroup_device_acl* array:

```
cgroup_device_acl = [
"/dev/null", "/dev/full", "/dev/zero",
"/dev/random", "/dev/urandom",
"/dev/ptmx", "/dev/kvm", "/dev/kqemu",
"/dev/rtc", "/dev/hpet", "/dev/net/tun"]
```

#disable KVM default virtual bridge.

```
>virsh net-destroy default
>virsh net-undefine default
```

#In */etc/libvirt/libvirtd.conf* file update the following variables:

```
listen_tls = 0
listen_tcp = 1
auth_tcp = "none"
```

#Update libvirtd_opts variable in */etc/init/libvirt-bin.conf* file.

```
env libvirtd_opts="-d -l"
```

in */etc/default/libvirt-bin* file update the following variable.

```
libvirtd_opts="-d -l"
```

Restart the libvirt service.

```
>service libvirt-bin restart
```

C.25 Install and Configure Nova on Compute node

```
>apt-get install nova-compute-kvm

#In /etc/nova/api-paste.ini file following variables are updated:
auth_host = 192.168.0.1
admin_tenant_name = service
admin_user = nova
admin_password = password

#In /etc/nova/nova-compute.conf file add the following variables.

[DEFAULT]
libvirt_type=kvm
libvirt_ovs_bridge=br-int
libvirt_vif_type=ethernet
libvirt_vif_driver=nova.virt.libvirt.vif.LibvirtHybridOVSBridgeDriver
libvirt_use_virtio_for_bridges=True

#Restart Nova services
>service nova-compute restart
```

C.26 Install and Configure Network Node

```
#Install the vSwitch
>apt-get install -y openvswitch-switch

#Start Open vSwitch service.
>service openvswitch-switch start

#Add configure internal bridge.
>ovs-vsctl add-br br-int

#Install the Quantum packages.

>apt-get install -y quantum-plugin-openvswitch-agent

#Edit /etc/quantum/quantum.conf file and update it with following variables:

core_plugin = \
        quantum.plugins.openvswitch.ovs_quantum_plugin.OVSQuantumPluginV2
auth_strategy = keystone
fake_rabbit = False
rabbit_host = 192.168.0.1
rabbit_password = password

#Edit /etc/quantum/plugins/openvswitch/ovs_quantum_plugin.ini file and modify:

[DATABASE]
sql_connection = mysql://quantum:password@192.168.0.1:3306/quantum
 [OVS]
tenant_network_type = gre
tunnel_id_ranges = 1:1000
integration_bridge = br-int
tunnel_bridge = br-tun
local_ip = 10.10.10.2
```

```
enable_tunneling = True

#Start the agent
>service quantum-plugin-openvswitch-agent restart
```

D OpenNebula deployment

D.1 Front-end Automation Script

```
#!/bin/bash -e

# Check for vitualization
apt-get update
apt-get -y upgrade
#egrep '(vmx|svm)' /proc/cpuinfo|wc -l
#adduser –home /var/lib/one oneadmin

wget -P /tmp/
https://dl.dropboxusercontent.com/u/92798871/thesis/openNebula/Ubuntu-
12.04-opennebula-3.8.3.tar.gz
tar xvzf /tmp/Ubuntu-12.04-opennebula-3.8.3.tar.gz -C /tmp/

sudo apt-get install rubygems libopenssl-ruby ruby-dev rake libxslt1-dev
/usr/share/opennebula/install_gems
sudo dpkg -i /tmp/opennebula-3.8.3/opennebula_3.8.3-1_amd64.deb
sudo apt-get install -f -y

#sudo passwd oneadmin
echo "----Configure Database---"
debconf-set-selections <<< 'mysql-server-5.5.29 mysql-server/root_password
password Cloud@2013'
debconf-set-selections <<< 'mysql-server-5.5.29 mysql-
server/root_password_again password Cloud@2013'
apt-get -y install mysql-server

echo "--"$(( x++ ))"----------------------------------Creat DataBases-------
--------------------------------"
mysql -uroot -pCloud@20133 << EOF
create database one;
CREATE USER 'one'@'localhost' IDENTIFIED BY 'Cloud@2013';
GRANT ALL ON one.* TO 'one'@'localhost';
FLUSH PRIVILEGES;
EOF

path="/etc/one/oned.conf"
sed -i 's/.*admin_token =.*/admin_token = password/g' $path
sed -i 's/DB = [ backend = "sqlite" ]/#DB = [ backend = "sqlite" ]/g' $path
sed -i '
/^# Sample configuration for MySQL/ a\
DB = [ backend = "mysql",\nserver  = "localhost",\nport    = 3306, \nuser
= "one",\npasswd  = "Cloud@2013"\ndb_name = "one" ]
' $path

sed -i 's/.*HOST_MONITORING_INTERVAL         = 600/HOST_MONITORING_INTERVAL
= 20/g' $path
sed -i 's/.*HOST_PER_INTERVAL               = 15/HOST_PER_INTERVAL
```

```
= 15/g' $path
sed -i 's/.*VM_POLLING_INTERVAL            = 600/VM_POLLING_INTERVAL
= 10/g' $path
sed -i 's/.*VM_PER_INTERVAL               = 5/VM_PER_INTERVAL
= 5/g' $path

echo '
HOST_HOOK = [
    name      = "error",
    on        = "ERROR",
    command   = "ft/host_error.rb",
    arguments = "$ID -r",
    remote    = "no" ]

VM_HOOK = [
   name      = "on_failAure_resubmit",
   on        = "FAILED",
   command   = "/usr/bin/env onevm resubmit",
   arguments = "$ID" ]
' >> $path

chown -R oneadmin /var/lib/one/datastores/
chown -R oneadmin /var/lib/one/images/
```

D.2 Prepare Compute Node

```
apt-get install -y dnsutils

apt-get install -y libvirt-bin qemu-kvm ruby

sed -i "s/.*libvirtd_opts=.*/libvirtd_opts=\"-d -l\"/"
/etc/default/libvirt-bin
sed -i "s/.*listen_tcp =.*/listen_tcp = 1/" /etc/libvirt/libvirtd.conf
service libvirt-bin restart
```

D.3 Front-end configurations

```
#OpenNebula main configuration “/etc/one/oned.conf”
HOST_MONITORING_INTERVAL         = 20
HOST_PER_INTERVAL               = 15

VM_POLLING_INTERVAL            = 10
VM_PER_INTERVAL               = 5
VM_DIR=/srv/nfs/images
SCRIPTS_REMOTE_DIR=/var/tmp/one

PORT = 2633
DB = [ backend = "mysql",
        server  = "localhost",
        port    = 3306,
        user    = "one",
        passwd  = "Cloud2013",
        db_name = "one" ]

VNC_BASE_PORT = 5900

DEBUG_LEVEL = 3


NETWORK_SIZE = 254
```

```
MAC_PREFIX   = "02:00"



DEFAULT_IMAGE_TYPE    = "OS"
DEFAULT_DEVICE_PREFIX = "hd"


IM_MAD = [
      name       = "im_kvm",
      executable = "one_im_ssh",
      arguments  = "-r 0 -t 15 kvm" ]
VM_MAD = [
    name       = "vmm_kvm",
    executable = "one_vmm_exec",
    arguments  = "-t 15 -r 0 kvm",
    default    = "vmm_exec/vmm_exec_kvm.conf",
    type       = "kvm" ]

TM_MAD = [
  name= "tm_ssh",
  executable = "one_tm",
  arguments  = "-t 15 -d ssh" ]

DATASTORE_MAD = [
    executable = "one_datastore",
    arguments  = "-t 15 -d fs,vmware,vmfs,iscsi,lvm"
]

HM_MAD = [
    executable = "one_hm" ]

HOST_HOOK = [
    name      = "error",
    on        = "ERROR",
    command   = "ft/host_error.rb",
    arguments = "$ID -r",
    remote    = "no" ]
VM_HOOK = [
   name      = "on_failure_resubmit",
   on        = "FAILED",
   command   = "/usr/bin/env onevm resubmit",
   arguments = "$ID" ]


AUTH_MAD = [
    executable = "one_auth_mad",
    authn = "ssh,x509,ldap,server_cipher,server_x509"
]

SESSION_EXPIRATION_TIME = 900

VM_RESTRICTED_ATTR = "CONTEXT/FILES"
VM_RESTRICTED_ATTR = "NIC/MAC"
VM_RESTRICTED_ATTR = "NIC/VLAN_ID"
VM_RESTRICTED_ATTR = "RANK"

IMAGE_RESTRICTED_ATTR = "SOURCE"
```

D.4 Network Configuration on Compute Node

```
auto lo
iface lo inet loopback

iface eth0 inet manual
auto lan0
iface lan0 inet static
  bridge_ports eth0
  bridge_stp off
  bridge_fd 0
  address 10.10.10.2
  netmask 255.255.255.0
  gateway 10.10.10.1
  dns-nameservers 128.39.89.8
```