
Sidharth Sachdev
Dept. of Computer Science & Engineering
India

Amit Mahajan
Dept. of CS & IT, University of Jammu
India

Abstract—Cloud computing is a new concept which combines all the disciplines, business models and technologies together to deliver IT capabilities which can be used together to implement easily and managed with minimum effort without any cloud provider support. The goal of this paper is to build up an open source private cloud using OpenStack (Essex) as cost effective solution for small and mid-sized organizations without spending huge amount of money on paid cloud service providers like AWS (Amazon Web Services). This paper describes the openness, adaptability and scalability which provide the flexibility to control the whole cloud infrastructure by using a dashboard service at administrator and client level. This paper also describes the establishment and deployment of cloud infrastructure (IaaS) for private network where users provide the tools for creating and managing virtual machines over the existing resources.

Keywords—OpenStack, private cloud, infrastructure, AWS, IaaS

I. INTRODUCTION

A. Cloud Computing

Cloud computing is a software platform that provides computing services with the help of internet. It allows users to make use of the software and hardware which is managed by the parties at remote locations like online store, social networking, webmail, e-commerce applications etc. With the help of a cloud computing model it provides access to information and computer resources from anywhere in the world with network and internet connection. It provides shared pool of resources that includes data storage space, networking facilities, computing processing power of specialized and user applications.

Cloud Computing Models: Cloud computing has three service models:-

1) Software as a Service (SaaS): In this type of service model, the cloud provider will provide the platform for the users to use their applications remotely like mail services, e-commerce applications etc. The user will have the common interface which helps the company to not to pay for extra charges for the licenses.

2) Platform as a Service (PaaS): In this type of service model, the cloud provider provides and manages the platform (hardware, server operating system, databases, storage, architecture, networks and Virtualization) but the client has to develop its application as per the need using these services provided by the cloud provider. The client has to manage the applications and the cloud service provider manages everything else like Google App Engine.

3) Infrastructure as a Service (IaaS): In this service model, the cloud service provider provides and manages the infrastructure (Virtualization, servers, networking and storage). This helps to avoid spending on hardware and human resources and which reduces the return of investment risk. The client can execute and manages virtual machines where it can use its applications, data, operating system, middleware and runtime like Amazon EC2, Rackspace, etc.

Characteristics of Cloud computing

There are three main characteristics of cloud computing service which differs it from traditional hosting service are:-

1) On demand availability

2) Rapid elasticity (pay per use) for users

3) Resource pooling

Categories of Cloud Computing: Cloud computing has three categories under which they can be deployed: private, public or Hybrid.

1) Private cloud: In such cloud infrastructure the cloud computing platform is operated solely for a specific organization under the control of IT department and is managed by the organization behind the organizational firewall. Private cloud offers the same features as public cloud and eliminates issues related to control, data, security etc.

2) Public cloud: In such cloud infrastructure the cloud service provider charges for their services from the organization. Like Amazon (AWS), Microsoft and Google, etc. own and operate the infrastructure of the
cloud and offer access to the services in the cloud with the help of Internet. Amazon Web Services (AWS) are the largest public cloud provider.

3) Hybrid cloud: is a cloud infrastructure which combines both private and public cloud infrastructures together. They're entities remained as separate but offering the benefits of both the deployment environments.

B. Open Source Cloud

Open source provides flexibility to the users to choose the product and even provide freedom to change the source code for own user’s need. This brings openness and makes the product more effective for further future uses. This ability of freedom and openness is encouraging more and more programmers. Who are migrating towards to work on open source cloud packages as don’t have to pay, look over proprietary issues. An open source cloud is growing and becoming more effective for the IT Industries and organizations who wanted to use the cloud facilities for hosting and other services. Not only Open Stack is growing in other open source cloud are also growing like cloudstack, opennebula, eucalyptus etc.

The figure (1) shows the trend line of the 5 major cloud computing solutions which had grown with respect to the time. Amazon AWS is the oldest among them and till date they are widely accepted and used. But the trend also shows that OpenStack an open source cloud is also emerging as one of the fastest cloud computing solution. People have started using this solution taking into account its openness and scalability to implement the cloud infrastructure in any environments (private, public or Hybrid).
The figure (2) shows that two cloud services are widely accepted in the market. The Amazon always attracted more users and is the most widely acceptable cloud but at the same time, OpenStack is also gaining popularity day by day. The statistics show past 12 months interest of the users where OpenStack has gained more popularity. Therefore, open stack is the future for the cloud especially in an open source environment.

C. OpenStack

OpenStack is an open source free cloud computing software, released under the same terms as Apache license. It has the ability to control extensive processes, storage and resources all through the data center, all supervised through a dashboard that gives the admin control while enabling the clients with procurement assets through a web interface. It is a cloud solution to provide an infrastructure as a service (IaaS). Open Stack is basically a collection of open source projects and organizations can set up and run their cloud compute and storage infrastructure using this collection of open source projects. NASA and Rackspace (RAX) started OpenStack project and NASA provided a code for Compute Part and Rackspace provided hosting / storage infrastructure. Open Stack is a non-profit organization entity which is established in September 2012, who manages and promotes distribution, development and adoption of the cloud computing software. The foundation has already attracted more than 7,000 individual members from 100 countries and 850 different organizations, over 190 companies including Rackspace®, Dell, HP, IBM, and Red Hat®, Canonical, Citrix, etc. OpenStack has the ability to work in private, public and hybrid environments and at enterprise grade level.

Open Stack has a modular architecture which provides users the flexibility to design the cloud as required, without any proprietary issues related to hardware or software requirements. It further has the ability to put together with existing and third party technologies. It uses various components to work together as a service. Which are designed to administer and mechanize team of computing resources and can work with widely available Virtualization technologies.

D. Components of OpenStack Essex:

1. Nova component is used for providing Compute Service. It acts as the Computing controller in the OpenStack Cloud. It provides management platform that supports the life cycle of instances within the OpenStack cloud. It is also used to manage and mechanize the pool of computer resources which can be used to work with various

![Diagram of OpenStack Components](image1.png)

Fig. 3 shows the Working with Open Stack Components/Services

![Diagram of System Architecture](image2.png)

Fig. 4 shows the System Architecture of the OpenStack

D. Components of OpenStack Essex:

1. Nova component is used for providing Compute Service. It acts as the Computing controller in the OpenStack Cloud. It provides management platform that supports the life cycle of instances within the OpenStack cloud. It is also used to manage and mechanize the pool of computer resources which can be used to work with various

© 2013, IJARCSSE All Rights Reserved
virtualization technologies or hypervisors. It does not provide virtualization of its own but uses Libvirt API to interact with hypervisors.

2. The Glance component also called OpenStack Imaging Service. It provides registration, delivery, discovery services for the cloud disk and server images. It also provides lookup and retrieval system for the virtual machine images.

3. Keystone component is used to provide Identity Service. Its job is to provide all authentication systems across the cloud infrastructure and can be combined with the existing services like LDAP which are running at the backend directory. It provides authorization and authentication services for all components of OpenStack. Keystone provides two ways of authentication. One is username/password based and the other is token based. A keypair consisting of private key/public key is generated to be able to start instances on OpenStack. Which are inserted into the instances to make password-less SSH access for the instance. This can be downloaded from the OpenStack Dashboard.

4. Horizon component in OpenStack provides an Administrative Web - Interface. Web based dashboards can be used to manage/administer OpenStack services. It is also used to manage the instances and Images, creating keypairs, manipulating the Swift containers and volumes to instances etc. Apart from this, dashboard gives the users to access the instance console, through which can be connected to an instance through VNC. OpenStack Dashboards can be used to manage various OpenStack services. It can be used to manage instances and images, create keypairs, attach volumes to instances, manipulate Swift containers etc. The OpenStack Dashboard is accessible via http://<ip_address of server1>

5. Swift component provides Storage Service in the OpenStack cloud. It provides virtual object store which is distributed, eventually for OpenStack. This component also has great capability to store large size files and large number of files among servers in an organized way. Swift also provides redundancy and failover management. The storage associated with the instance is non-persistent and therefore the data which is generated and stored on the disk of the instance is removed as soon as the instance gets terminated. So we create a dedicated volume and attach it to the running instance.

Nova controls all components and glance manages images, keystone provides authentication and authorization for all components of OpenStack, horizon provides web interface to manage all infrastructure manually and even remotely and swift for storage capabilities and client can login in cloud infrastructure and run a virtual machine/instance to deploy service/application for cloud users using images.

II. DEPLOYMENT OF PRIVATE CLOUD USING UBUNTU 12.04 SERVER EDITION AND OPENSTACK

A. Prerequisites Hardware Requirements for OpenStack cloud

| TABLE I. Hardware Requirements |
|-----------------|-----------------|-----------------|
| **Server 1** | **Server 2** | **Client** |
| Processor | Core i5 (x64 architecture) | Core i5 (x64 architecture) | Core i5 (x64 architecture) |
| RAM | 6 GB | 8 GB | 8 GB |
| No. of NICs | 2 | 2 | 1 |
| Hard disk | 750 GB | 750 GB | 320 GB |
| VT - Enabled | Yes | Yes | Yes |
| Operating System | 64 Bit 12.04 Ubuntu Server | 64 Bit 12.04 Ubuntu Server | 64 Bit 12.04 Ubuntu Desktop |

Fig. 5 shows the Server / Client implementation model (Same model we used to establish and for testing purpose)
B. Installation and Configuration Steps
To check the working of the Cloud using OpenStack, in this paper we set up the Private Cloud infrastructure using the minimum hardware requirements as mentioned in Table I. For the establishment of the Infrastructure, we have used three machines, out of which 2 machines have been configured as servers and 3rd machine act as a client to access web interface as described in figure (5). On Server1 with hostname MC and server2 with host NC, we installed Ubuntu Server 12.04 64 Bit Version. On Server1 also called as Cloud Controller Node, we have installed all the components of open stack i.e. Nova, Glance, Keystone, Horizon and Swift whereas on Server 2 called as compute node, has only nova-compute i.e. virtual machines/instances. On Client Machine i.e. Client node, we installed Ubuntu Desktop 12.04 64 Bit Version. This will run Dashboard Service, which is used to manage the OpenStack Cloud Infrastructure and administrator and client can create or start instances using images. Table II describes functionality and other details of servers and client. All the machines used should be 64 Bit (x64) bit and have Virtualization Technology Enabled. To enable VT we need to check this option in a computer bios setup. We have connected our servers and client to private switch. This private switch is configured with the organizational network, which is connected with the Internet.

Next section describes the steps to configure a cloud infrastructure based on OpenStack using 3 machines.

C. STEPS FOR CREATING SERVER 1, SERVER 2 AND CLIENT FOR THE DEPLOYMENT OF OPENSTACK CLOUD

Server 1:

1) Install 64 bit version of Ubuntu server 12.04 on Server 1, After installation update the Ubuntu OS using
   
   \[ \text{sudo apt-get update} \]
   \[ \text{sudo apt-get upgrade} \]

2) Create a dedicated physical volume for nova-volume and Volume Group named nova-volumes.
   \[ \text{sudo pvcreate /dev/sda6} \]
   \[ \text{sudo vgcreate nova-volumes /dev/sda6} \]

3) Install bridge-utils using \texttt{sudo apt-get install bridge-utils}

4) Open network configuration file (/etc/network/interfaces) and configure manually
   \[ \text{Assign eth0 – 192.170.1.30, eth1 - 192.168.3.1} \]

5) Install Open-SSh Server and NTP Server for Time Synchronization using
   \[ \text{sudo apt-get install openssh-server} \]
   \[ \text{sudo apt-get install ntp} \]
   Add these two lines to ntp configuration file (/etc/ntp.conf)
   \[ \text{server 192.170.1.1} \]
   \[ \text{fudge 127.127.1.0 stratum 10} \]

6) Install database packages. MySQL, PostgreSQL or SQLite database packages can be used. We used MySQL to create databases that to be used with nova, glance and keystone.
   \[ \text{sudo apt-get install mysql-server python-mysqldb} \]
   Open the MySQL configuration file (/etc/mysql/my.cnf) and change the bind-address = 0.0.0.0

7) Create database & user for nova and then grant all privileges and set password. Same can be done for Glance, Keystone and keystone
   Command to create database: \texttt{mysql -uroot -p@ -e 'CREATE DATABASE nova;'}
   Command to create user: \texttt{mysql -uroot -p@ -e 'CREATE USER novadb;'}
   Grant all privileges for novadb on the database "nova".
   \[ \text{mysql -uroot -p@ -e ”GRANT ALL PRIVILEGES ON nova.* TO 'novadb'@’%’;”} \]
   Create a password for the user "novadb".
   \[ \text{sudo mysql -uroot -p@ -e ”SET PASSWORD FOR 'novadb'@’%’ = PASSWORD('password_text');”} \]
   We used xyz in place of password_text

8) Install keystone packages using \texttt{sudo apt-get install keystone python-keystoneclient python-keystone}
   Open the keystone configuration file (/etc/keystone/keystone.conf) and change the admin_token =

### TABLE II. Shows the details of servers and client

<table>
<thead>
<tr>
<th>Server 1</th>
<th>Server 2</th>
<th>Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>Also Called Cloud Controller Node</td>
<td>Also Called compute nodes</td>
<td>Client node</td>
</tr>
<tr>
<td>Functionality</td>
<td>Nova-compute : runs instances / virtual machines</td>
<td>Image Bundling and provides a web interface (Horizon)</td>
</tr>
<tr>
<td>Network Interfaces</td>
<td>eth0 - Private N/W, eth1 - Private N/W</td>
<td>eth0 - Private N/W</td>
</tr>
<tr>
<td>IP addresses</td>
<td>eth0 – 192.170.1.30, eth1 - 192.168.3.1</td>
<td>eth0 – 192.170.1.31, eth1 - 192.168.3.2</td>
</tr>
<tr>
<td>Hostname</td>
<td>MC</td>
<td>NC</td>
</tr>
<tr>
<td>DNS servers</td>
<td>192.170.1.1</td>
<td>192.170.1.1</td>
</tr>
<tr>
<td>Gateway IP</td>
<td>192.170.1.1</td>
<td>192.170.1.1</td>
</tr>
</tbody>
</table>
admin and connection = mysql://keystonedb:xyz@192.170.1.30/keystone

9) Create two tenants - admin and service using
   keystone tenant-create --name admin
   keystone tenant-create --name service

10) Create four keystone users - admin, nova, glance and swift
    keystone user-create --name admin --pass admin --email dmintest@live.com
    same as for nova, glance and swift

11) Create two keystone roles - admin and Member using
    keystone role-create --name admin
    keystone role-create --name Member

12) Check tenants, users and roles using keystone tenant-list / user-list / role-list

13) Add Roles to Users in Tenants using
    keystone user-role-add --user (USER_ID) --role (ROLE_ID) --tenant_id (TENANT_ID).
    Check USER_ID, ROLE_ID & TENANT_ID from keystone tenant-list / user-list / role-list
    Add a role of ‘admin’ to the user ‘admin’ of the tenant ‘admin’
    Add a role of ‘admin’ to the users ‘nova’, ‘glance’ and ‘swift’ of the tenant ‘service’
    Add a role ‘Member’ role to the user ‘admin’ of the tenant ‘admin’

14) Create keystone service using
    keystone service-create --name service_name --type service_type --description 'Description of the service’
    service_name can be volume, nova, swift, glance, keystone and ec2 with service_type - volume, compute, object-store, image, identity and ec2 respectively.
    Check keystone service list using keystone service-list

15) Create Endpoints for nova-compute using
    keystone endpoint-create --region myregion --service_id 60ebb0f03d734133936c49c0b67ddd2a --publicurl 'http://192.170.1.30:8774/v2/$tenant_id' --adminurl 'http://192.170.1.30:8774/v2/$tenant_id' --internalurl 'http://192.170.1.30:8774/v2/$tenant_id'
    Same as for nova-volume, glance, swift, keystone & ec2.

16) Install Glance and glance components using
    apt-get install glance glance-client glance-api glance-registry python-glance glance-common
    Open glance configuration file (/etc/glance/glance-api-paste.ini) & (/etc/glance/glance-registry-paste.ini) and make changes

    tenant_name = service
    admin_user = glance
    admin_password = glance

    Open /etc/glance/glance-registry.conf and change

    sql_connection = mysql://glancedb:xyz@192.170.1.30/glance

    And add below mentioned two lines at the end of file for authentication for /etc/glance/glance-registry.conf
    and /etc/glance/glance-api.conf

    [paste_deploy]
    flavor = keystone

17) Create glance schema using
    sudo glance-manage version_control 0
    sudo glance-manage db_sync
    To test glance is properly installed run glance index and then echo $? ,
    If Output is 0 means Glance is working properly otherwise not.

18) Install Nova and its components
    sudo apt-get install nova-api nova-cert nova-compute nova-compute-kvm nova-doc nova-network nova-objectstore nova-scheduler nova-volume rabbitmq-server novnc nova-consoleauth

19) Configure the Nova file (/etc/nova/nova.conf)

   --s3_host=192.170.1.30
   --ec2_host=192.170.1.30
   --rabbit_host=192.170.1.30
   --cc_host=192.170.1.30
   --nova_url=http://192.170.1.30:8774/v1.1/
   --routing_source_ip=192.170.1.30
   --glance_api_servers=192.170.1.30:9292
   --iscsi_ip_prefix=192.168.4
   --sql_connection=mysql://novadb:xyz@192.170.1.30/nova
   --ec2_url=http://192.170.1.30:8773/services/Cloud
   --keystone_ec2_url=http://192.170.1.30:5000/v2.0/ec2tokens
   --novncproxy_base_url=http://192.170.1.30:6080/vnc_auto.html
   --vncserver_proxyclient_address=192.170.1.30
   --vncserver_listen=192.170.1.30
Now we need to configure our dedicated partition for nova-volume i.e sda6

20) Change the ownership of the /etc/nova folder and permissions for /etc/nova/nova.conf using

```bash
sudo chown -R nova:nova /etc/nova
sudo chmod 644 /etc/nova/nova.conf
```

Open nova configuration file (`/etc/nova/api-paste.ini`) and make changes

```ini
admin_tenant_name = service
admin_user = nova
admin_password = nova
```

21) Create nova schema using `sudo nova-manage db sync`

22) Provide a range of IPs to be associated to the instances using

```bash
sudo nova-manage network create private --fixed_range=192.168.4.32/27 --num_networks=1 --bridge=br100 --bridge_interface=eth1 --network_size=32
```

23) To Check nova services are working or not using nova-manage service list

24) Install OpenStack Dashboard using `sudo apt-get install openstack-dashboard` and then restart apache service using `service apache2 restart`

25) Install swift and its components using

```bash
sudo apt-get install swift swift-proxy swift-account swift-container swift-object xfsprogs curl python-pastedeploy
```

Create a file named swift-disk with approx. 975 MB space that will be used as looback disk for Swift storage backend using `sudo dd if=/dev/zero of=/srv/swift-disk bs=1024 count=0 seek=1000000`

Create a directory `/mnt/swift_backend` for mounting swift storage backend and add it to file system table for permanent mounting using `sudo mkdir /mnt/swift_backend`

26) Create some nodes for backend that to be used as storage devices and set ownership to `swift` user and group.

```bash
mount /mnt/swift_backend
pushd /mnt/swift_backend
sudo mkdir node1 node2 node3 node4
popd
```

Create configuration file (`/etc/swift/swift.conf`) and add random string

```ini
[swift-hash]
swift_hash_path_suffix = 796dc37f62886b45
```

27) Rsync maintains object replicas and Enable RSYNC (`/etc/default/rsync`) - Set RSYNC_ENABLE=true

28) Generate a random string using `od -t x8 -N 8 -A n < /dev/random` and this is used if we want to add more nodes in our setup

Create configuration file (`/etc/swift/swift.conf`) and add random string

```ini
[swift-hash]
swift_hash_path_suffix = 796dc37f62886b45
```

29) Now Create Swift Proxy Server (`/etc/swift/proxy-server.conf`) and configure file
allow_account_management = true
account_autocreate = true
set log_name = swift-proxy
set log_facility = LOG_LOCAL0
set log_level = INFO
set access_log_name = swift-proxy
set access_log_facility = SYSLOG
set access_log_level = INFO
set log_headers = True
account_autocreate = True

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:cache]
use = egg:swift#memcache
set log_name = cache

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
auth_protocol = http
auth_host = 127.0.0.1
auth_port = 35357
auth_token = admin
service_protocol = http
service_host = 127.0.0.1
service_port = 5000
admin_token = admin
admin_tenant_name = service
admin_user = swift
admin_password = swift
delay_auth_decision = 0

[filter:keystone]
paste.filter_factory = keystone.middleware.swift_auth:filter_factory
operator_roles = admin, swiftoperator
is_admin = true

30) Configure Account server (/etc/swift/account-server/1.conf)

[DEFAULT]
devices = /srv/node1
mount_check = false
bind_port = 6012
user = swift
log_facility = LOG_LOCAL2

[pipeline:main]
pipeline = account-server

[app:account-server]
use = egg:swift#account

[account-replicator]
vm_test_mode = no

[account-auditor]
[account-reaper]

And execute these commands to configure other Swift Container Server. These commands basically copy configurations from 1.conf to 2.conf and so on and then make changes that differentiate between them and each of file correspond to a device under /srv
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/2.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/3.conf
sudo cp /etc/swift/account-server/1.conf /etc/swift/account-server/4.conf
sudo sed -i 's/6012/6022/g;s/LOCAL2/LOCAL3/g;s/node1/node2/g' /etc/swift/account-server/2.conf
sudo sed -i 's/6012/6032/g;s/LOCAL2/LOCAL4/g;s/node1/node3/g' /etc/swift/account-server/3.conf
sudo sed -i 's/6012/6042/g;s/LOCAL2/LOCAL5/g;s/node1/node4/g' /etc/swift/account-server/4.conf

31) Configure Swift Container Server (/etc/swift/container-server/1.conf)
   Configuration is same as we have done in account server; few below mentioned lines are changed
   
   bind_port = 6011
   
   pipeline = container-server
   
   [app:container-server]
   use = egg:swift#container

   [container-replicator]

   [container-updater]

   [container-auditor]

   [container-sync]

   Copy 1.conf text to other conf file as we have done swift Account server and set the bind posts to 6021, 6031 and 6041 for other nodes.

32) Configure Swift Object Server (/etc/swift/object-server/1.conf)
   
   bind_port = 6010
   
   pipeline = object-server
   
   [app:object-server]
   use = egg:swift#object

   [object-replicator]

   vm_test_mode = no

   [object-updater]

   [object-auditor]

   Copy 1.conf text to other conf file as we have done swift Account server and set the bind posts to 6020, 6030 and 6040 for other nodes.

33) Swift Rings maintains the information about physical location of objects, their replicas and devices.
   Configure Swift Rings

   cd /etc/swift
   pushd /etc/swift
   sudo swift-ring-builder object.builder create 18 3 1
   sudo swift-ring-builder container.builder create 18 3 1
   sudo swift-ring-builder account.builder create 18 3 1

   Add zones and balance the rings using
   swift-ring-builder <builder_file> add <zone> -<ip_address><port><device> <weight>

   Change ownership of /etc/swift directory to ‘swift’ using sudo chown -R swift.swift /etc/swift

34) Check Swift using swift -v -V 2.0 -A http://127.0.0.1:5000/v2.0 -U service:swift -K swift stat
   Valid output shows sift is working properly else error gets displayed on screen.

Server 2:

1) Install 64 bit version of Ubuntu server 12.04 on Server 2, After installation update the OS using below mentioned commands
   sudo apt-get update
   sudo apt-get upgrade

2) And then we install bridge-utils using sudo apt-get install bridge-utils

3) Open network configuration file and configure manually (/etc/network/interfaces)
Assign `eth0` – 192.170.1.31, `eth1` - 192.168.3.2

4) Install Open-SSH Server and NTP Server for Time Synchronization using
   
   ```bash
   sudo apt-get install openssh-server
   sudo apt-get install ntp
   ```
   
   Add server1 IP in ntp configuration file (`/etc/ntp.conf`) for time sync from server1 to server2
   
   ```bash
   server 192.170.1.30
   ```

5) Configure Network Configurations manually. Assign `eth0` – 192.170.1.31, `eth1` - 192.168.3.2
   
   Install nova - compute using `sudo apt-get install nova-compute`

6) Open nova configuration file and make changes as we have done in server1
   
   And check that all services of nova are working or not using `sudo nova-manage service list`

**Client:**

1) Install 64 bit version of Ubuntu Desktop 12.04 on Client. After installation update the OS using below mentioned commands
   
   ```bash
   sudo apt-get update
   sudo apt-get upgrade
   ```

2) Configure Network Configurations manually (`/etc/network/interfaces`). Assign `eth0` – 192.170.1.32

3) Install NTP Client and Open-SSh Server using
   
   ```bash
   sudo apt-get install ntp
   sudo apt-get install openssh
   ```

4) Install client tools using `sudo apt-get install python-novaclient glance-client swift`

5) Install hypervisors, we used KVM so we can install KVM using `sudo apt-get install qemu-kvm`

6) Open Firefox Browser and enter IP address of Server 1 (192.170.1.30)

7) Add either customized image or cloud ready images to glance as shown in figure (6) (cloud ready images are available on internet)

   ```bash
   glance add name="cirros" is_public=true container_format=ova disk_format=qcow2 < cirros-0.3.0-x86_64.img
   ```

   and to display images added to glance run command `nova image-list`

8) Create public/private key either from dashboard or from Command User Interface using

   ```bash
   nova keypair-add --pub_key ~/.ssh/id_rsa.pub demo
   ```

   and to display keys execute command `nova keypair-list`

9) Add security group rules either from dashboard or from CUI using

   ```bash
   nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
   nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
   ```
and display security group rules added to default run **nova secgroup-list-rules default** or check from Dashboard as shown in figure (7).

10) Run Instance / Virtual Machine from image (automatically provision of IP address for instance)

**nova boot --flavor 1 --image <image_id> --key_name mykey --security_group mykey myinstance**

List Instances: nova list

Reboot Instance: nova reboot <Instance ID>

Delete Instance: nova delete <Instance ID> or we can do from Dashboard (figure 8)

Fig. 8 shows that instance / virtual machine running on our architecture with **IP 192.168.4.34 that was provided from Pool** and dedicated volume for that virtual machine / instance
11) Create a dedicated volume for instance
   `nova volume-create --display_name demoinstance`
   and to list volume using `nova volume-list` and Attach volume to that Instance from Dashboard
12) Now our infrastructure is ready and we can establish/upload particular service or application and install multiple VM on our private cloud as we are using cirros os in figure (9) & figure (10).

Fig. 9 shows Logging in cirros operating system

Fig. 10 shows ls/ (list) command on cirros operating system
III. CONCLUSIONS

In the paper we tried to build private cloud hosting infrastructure with minimum hardware requirements that can be used for small scale and medium scale developments. Such infrastructure can be used for the deployment purposes without HA (high availability). We can expand this architecture by adding more nova compute servers and other hardware relating to performance and storage. We executed instances / virtual machines on cirros OS for testing purpose and productivity. We can create multiple projects for multiple clients under one infrastructure and admin has the power to manage them. Therefore the client can launch instances from images using particular security group and keyboard and can deploy services. This OpenStack is highly beneficial for communities / organizations / institutions / research centers that do not have enough funds to deploy high grade cloud services at their own place. Especially for academic institutions and Universities that don’t have funds and they have talked to management of the resources. Like in universities, where all departments have own resources (hardware), websites and online storage space / web space. And even work independently and every department has to face failure and delay issues. And this model will eliminate all the issues and maintain centralized core cloud computing for maintenance. OpenStack Cloud Infrastructure is very cost-effective, flexible and elastic and if any organization goes to any paid cloud providers like AWS, that organization have to pay thousands of USD as subscription charges and technical support charges. And we can implement the same kind of services using OpenStack. The main benefit of OpenStack is that it is open source and day by day it is improving, many engineers from all over the world are working on it – detecting bugs, creating solutions for bugs, and making OpenStack effective.

Future Scope: Since this model is for the deployment in small and mid-sized organizations but for future scope it can be further strengthened by incorporating HA (High availability) to secure the hardware and software failure. Security of the cloud is an important issue. Since users of the cloud are not finding themselves comfortable with the security of their data. These issues need to be properly addressed for the further improvement of the private cloud infrastructure for its better adaptability.

ACKNOWLEDGMENT

We would like to thank Prof. Vibhakar Mansotra (Head of Department, Computer Science and Information Technology) for providing the opportunity to work on the cloud infrastructure by providing hardware and network access and secondly we would like to thank authors of instruction manual (CSS Corp Open Source Services) and for providing time to time support.

REFERENCES