IaaS Environment Creation Experiments with OpenStack

Silviu – Gabriel Topoloi, Eugen Borcoci
Department of Telecommunications
University POLITEHNICA of Bucharest
Bucharest, Romania
Emails: silviutopoloi@gmail.com, eugen.borcoci@elcom.pub.ro

Abstract—**OpenStack is an open cloud computing software platform that allows the users to create Infrastructure as a Service (IaaS) cloud environments suited for all types of deployments and environments (prod, pre-prod, test, dev, etc.). The platform is backed-up by a large and active community that continuously improves it, thus making it a serious competitor in today's cloud market. This paper presents a complete experimental work for IaaS environment creation with OpenStack as an alternative to others presented in the public literature. It can provide the baseline for future integration of different modules, e.g., between OpenStack and OpenDaylight (Open Source Software Defined Networking Platform), where OpenDaylight is used to create networking services, together with the default module provided by OpenStack (called Neutron). Developers, researchers, academic members and user communities can use the information in this paper as a practical guide to create their own cloud environments, allowing integration of their own work in many possible contexts: cloud, Software Defined Networking (SDN), Network Function Virtualization (NFV), Data Centers, and so on.**

Keywords - OpenStack; cloud; Software Defined Networking; Network Function Virtualization ; Infrastructure as a Service.

I. INTRODUCTION

OpenStack is a set of software tools for building and managing cloud computing platforms for public and private clouds. It is backed by some of the biggest companies in software development and hosting (AT&T, Ericsson, Huawei, Intel, Rackspace, Redhat, Suse, Tencent Cloud), as well as thousands of individual community members. OpenStack is managed by the OpenStack Foundation, a non-profit organization that oversees both development and community-building around the project [1].

OpenStack supports users to deploy Virtual Machines (VM) and other instances that handle different tasks for managing a cloud environment on the fly. It is horizontally scalable, i.e., it can concurrently serve more or fewer users on the fly by just spinning up more instances. For example, a mobile application that needs to communicate with a remote server can divide the communication work across many different instances, all communicating with one another but scaling quickly and easily as the application gains more users [1].

OpenStack is an open source software giving open access to the source code, to make any changes or modifications needed, and freely sharing these changes back out with the community at large. Consequently, it has the benefit of thousands of developers all over the world cooperating to make the product stronger, more robust and more secure [1] - [3].

As mentioned in the abstract, this paper presents a complete, pragmatic work for IaaS environment creation with OpenStack. Note that, currently, it is hard for users to find a complete view on how OpenStack should be deployed without bottlenecks in a carefully defined environment.

Different OpenStack deployment scenarios [4][5][7][8] are available on the Internet; however, after several attempts to have OpenStack installed following those steps only, the developer discovers that some additional steps and problem solving solutions are missing. In this paper, we try to fill this gap by presenting in one place a complete installation of OpenStack using Devstack, as well as the configurations problems, along with their solutions, thus guiding the developer during the process.

The structure of the paper is as follows: Sections I and II present a high level view on OpenStack and what can and has been achieved with this platform until now. Section III describes the architecture and services. Section IV elaborates a step by step OpenStack IaaS deployment. Section V presents some hints for the deployment to end users and developers.

II. OPENSTACK AS A SOLUTION ENABLER AND RELATED WORK

OpenStack is a key enabler in the adoption of cloud technology, while following Public, Private, or Hybrid models [2].

OpenStack, as is, can be adopted by any user who wants to start exploring the cloud world or by any developer who wants to have a free cloud environment, built up in minutes, to test the applications that he/she develops. The platform has a wide variety of usage scenarios.

Big industry players like Oracle and Huawei offer OpenStack also for Enterprise usage. For example, Huawei OpenStack-based platform is called Flexible Engine, while Oracle's is named Oracle OpenStack. Naturally, an enterprise platform is more elaborated and reliable than a free one, especially for the support it provides and the available structure of the information needed for different implementations. However, a significant price has to be paid if wanting an enterprise solution.

Many companies have already adopted OpenStack. According to iDatalabs [9] there are around 6,856 companies that use OpenStack; the reason is the openness and wide variety of usage scenarios.

The work that has been done until now can be marked in several fields, like: security, monitoring, cloud Service Orchestration, Networking (SDN, NFV), cloud IaaS and so on. Several references can provide further details [10]-[13].

However, given the large spread of published work related to OpenStack, it is difficult to find a thorough nutshell document presenting the development steps. This article tries to do it. In our knowledge, the practical hints and ideas provided in Section V cannot be found all in one place. Here, the practical hints are based on a real experience gathered when the user/developer actually interacts and works with OpenStack.

Therefore, this document can provide a good and helpful guide to all people wanting to use OpenStack as a development, test or production cloud Platform. It can be considered as a good and practical contribution to what has been done until now.

III. OpenStack Architecture and Services

A. *OpenStack Architecture*

This section shortly presents the OpenStack architecture and services (see Figure 1, [2]).

B. *OpenStack Services*

The above architecture presents the OpenStack Services and how they communicate. Further, the paper will speak shortly about each presented service.

OpenStack embraces a modular architecture (Figure 2) to provide a set of core services that facilitate scalability and elasticity as core design tenets [2].

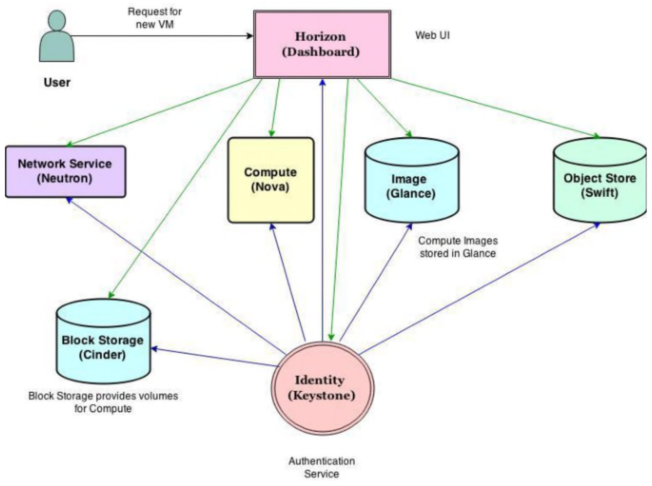


Figure 1. Loosely coupled architecture of OpenStack [6]

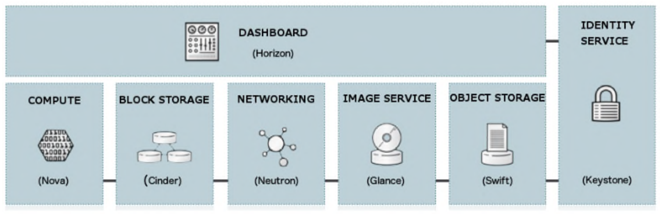


Figure 2. OpenStack modular architecture [2]

The OpenStack services are:

a. *Compute (Nova)* provides services to support the management of VM instances at scale, instances that host multi-tiered applications, dev or test environments, "Big Data" crunching Hadoop clusters, or high-performance computing [2].
b. *Object Storage (Swift)* provides support for storing and retrieving arbitrary data in the cloud [2].
c. *Block Storage (Cinder)* provides persistent block storage for Compute instances [2].
d. *Networking (neutron, previously called quantum)* provides various networking services to cloud users (tenants), such as IP address management, Domain Name Server (DNS), Dynamic Host Configuration protocol (DHCP), load balancing, and security groups (network access rules, like firewall policies) [2].
e. *Dashboard (Horizon*) provides a web-based interface for both cloud administrators and cloud tenants [2].
f. *Identity (Keystone)* is a shared service that provides authentication and authorization services throughout the entire cloud infrastructure. The Identity service has pluggable support for multiple forms of authentication [2].
g. *Image (Glance)* provides disk-image management services, including image discovery, registration, and delivery services to the Compute service, as needed [2].

Messaging is used for internal communication between OpenStack services. By default, message queues are used, based on the Advanced Messaging Queuing Protocol (AMQP). Like most OpenStack services, AMQP supports pluggable components. Today, the implementation back end could be RabbitMQ, Qpid, or ZeroMQ [2].

C. *Data Protection & Security*

The OpenStack Identity Service (Keystone) takes care of both the user's and customer's data, because the authentication uses a combination of domains, projects (tenants), users and roles. By creating tenants, logical customer (and their respective data) segregation is possible. This means that each customer has access to his/her own data and no other data. Therefore, customer data is secured.

For example, in public clouds, logical segregation is very important, because each customer is deployed in the same public cloud; however, it is logically separated from the rest of the customers with the help of tenants.

With OpenStack, private clouds can also be created for customers that do not want a logical segregation, but a physical one. Thus, if a customer opts for a private cloud, the customer can have access to a dedicated OpenStack environment that is deployed on a dedicated server, where no other customer has access to. Of course, this implies higher costs, but the data is physically separated. Either way, OpenStack assures that the customer's data is fully protected and secure.

At a user level, all the access is controlled and logged. Also, the access is segregated based on users and roles. Therefore, a user cannot see or access the data and elements for another user. Also, in this way, customer data is protected and access to it is fully controlled.

IV. Creation of an IaaS Environment in OpenStack

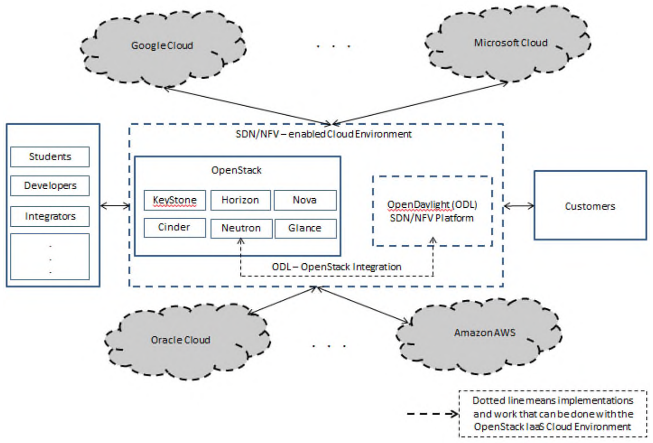
A. *Role of the Environment*

The environment that is going to be created has the role of providing a basic cloud platform for experiments in different areas like: SDN, NFV, application development, integration with other cloud platforms and more.

OpenStack services used for the platform implementation are: *Keystone, Horizon, Nova, Cinder, Neutron* and *Glance*. With the help of these services, the user will be able to create the necessary IaaS environment, meaning, Compute instances with Networking and Storage that are ready for the user's purposes.

B. *The Implemented Architecture*

The implemented architecture (Figure 3) shows the OpenStack services, as components used to implement the OpenStack cloud IaaS Environment and its capabilities. The capabilities are presented in Figure 3, with dotted-lines emphasizing future integration and work that can be done based on the created environment:



Google Cloud
. . .
Microsoft Cloud
SDN/NFV – enabled Cloud Environment
Students
Developers
Integrators
OpenStack
KeyStone
Horizon
Nova
Cinder
Neutron
Glance
OpenDaylight (ODL) SDN/NFV Platform
ODL – OpenStack Integration
Customers
Oracle Cloud
. . .
Amazon AWS
Dotted line means implementations and work that can be done with the OpenStack IaaS Cloud Environment

Figure 3. The implemented architecture

- Integrations with different platforms, like OpenDaylight;
- Integrations with different cloud providers;
- Providing cloud services to users and customers.

C. *Implementation steps*

The section will show the steps to be followed to install OpenStack and use this framework to create the cloud IaaS Environment that can be further used, as mentioned earlier, for example, as a Production environment for Application development and more.

After the creation of the IaaS Environment, the following tests can be performed:

- Ping the Compute instances from the external network (Internet Service Provider network in this case). This will be done from the Ubuntu machine;
- Connect to the Compute instances, using SSH, from the external network. This will be done from the Ubuntu machine;
- Test the connection between the Compute instances;
- Test if the Compute instances have access to the Internet. This will be tested by issuing a ping command to "google.com", from the Compute instances.

D. *The System Resources*

To support the implementation, a virtual machine image, Ubuntu 16.04.4, has been used, installed on a local machine. The hardware resources are:

Physical Host

Processor: Intel Core i5-8350U CPU @ 1.70 GHz (8 Virtual CPUs)

RAM Memory: 16 GB, HDD: 256 SSD

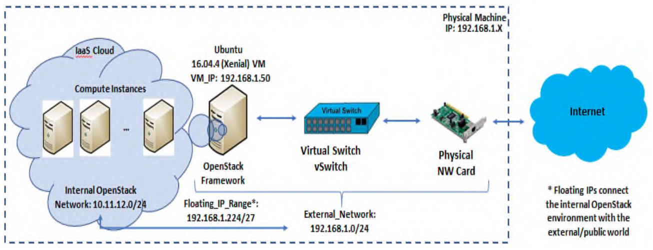
Ubuntu Virtual Box Image

Processor: 1 Virtual CPU, Memory: 4 GB

HDD: Starting from 10 GB and Dynamic Growing

The installation has been done locally, for testing purposes, on a powerful machine, but it can be scaled up easily to a datacenter environment capable of providing cloud services.

DevStack [14] has been used to automatically deploy OpenStack.



Physical Machine
IP: 192.168.1.X
IaaS Cloud
Compute Instances
...
Ubuntu
16.04.4 (Xenial) VM
VM_IP: 192.168.1.50
OpenStack Framework
Virtual Switch
Virtual Switch
vSwitch
Physical
NW Card
Internet
Internal OpenStack Network: 10.11.12.0/24
Floating_IP_Range*: 192.168.1.224/27
External_Network: 192.168.1.0/24
* Floating IPs connect the internal OpenStack environment with the external/public world

Figure 4. OpenStack deployment components

E. Installation of OpenStack Framework

1) Download and install Oracle Virtual Box;

a) Download and deploy Ubuntu 16.04.4 (Xenial) virtual box image;

b) Setup the Network to bridged mode;

c) Access the machine and install OpenStack Framework:

- First, run the following commands to pre-configure the Ubuntu environment: sudo apt-get update, sudo apt-get upgrade, sudo apt-get install openssh-server, sudo apt-get install git;
- Assign static ip to the network interface:
```
sudo nano /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback

auto enp0s3
iface enp0s3 inet static
  address 192.168.1.50
  netmask 255.255.255.0
  network 192.168.1.0
  gateway 192.168.1.1
  dns-nameservers 192.168.1.1
```
- Create "stack" user that will be used for OpenStack installation:
  `sudo useradd -s /bin/bash -d /opt/stack -m stack`
- Assign password to stack: sudo passwd stack;
- Add "stack" user to sudo group:
  `sudo usermod -aG sudo stack`
- Make sure it has sudo privileges:
  `echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack`
  Output should be:
  `stack ALL=(ALL) NOPASSWD: ALL`
- Login as stack: sudo su – stack;
- Download Openstack DevStack:
  `git clone https://github.com/openstack-dev/devstack.git -b stable/pike devstack/`
- Create local.conf file. This file will be used in the installation of OpenStack. The parameters that are setup here will be used to pre-configure the environment so that the user makes sure that the environment is fully working and accessible.
```
cat > local.conf <<EOF
[[local|localrc]]
FLOATING_RANGE=192.168.1.224/27
FIXED_RANGE=10.11.12.0/24
FIXED_NETWORK_SIZE=256
FLAT_INTERFACE=enp0s3
ADMIN_PASSWORD=secret
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
RECLONE=yes
```

Note: The Fixed Range Subnet represents the subnet that will be used for the internal OpenStack Network.

- Open and edit the file stack.sh as follows:

  Comment the below lines:
  # Start Services # ============== # Dstat # ----- # A better kind of sysstat, with the top process per time slice #start_dstat # Etcd # ----- # etcd is a distributed key value store that provides a reliable way to store data across a cluster of machines #if is_service_enabled etcd3; then # start_etcd3 #fi

  Save the above file.

- Run ./stack.sh and wait for the script to finish.
  Note that it will take some time!
  Final output should contain following elements:
```
This is your host IP address: 192.168.1.50
This is your host IPv6 address: ::1
Horizon is now available at
http://192.168.1.50/dashboard
Keystone is serving at http://192.168.1.50/identity/
The default users are: admin and demo
The password: secret
```

A fully working OpenStack environment is now available for deploying cloud environments.

F. Creating an IaaS environment in OpenStack

a) Login to the OpenStack Dashboard (192.168.1.50/dashboard) using the following credentials: user: admin & pass: secret. The User will be logged as admin, using the Project (workspace) admin.

b) First, create an internal network using the subnet specified in the local.conf file: 10.11.12.0/24.

Access Project -> Network -> Create Network and name the Network "internal".

c) Assign a subnet to the internal network using the following specifications:

Subnet name: subnet

Network address: 10.11.12.0/24

Gateway IP: 10.11.12.1

Enable DHCP

Allocation Pools: 10.11.12.2,10.11.12.10

DNS: 192.168.1.1

d) The external network, called public, is automatically created by the OpenStack stack.sh script, with the subnet specified in the local.conf file. The subnet is: 192.168.1.224/27.

e) Create a Router that will route the traffic in the internal network and will make the connection with the public network, thus the Compute instances will be able to communicate with the outside world.

Note: For the device to be able to route the traffic as mentioned before, the Router will have a Gateway interface with an IP automatically setup from the public network and an interface connected in the internal network, with an IP corresponding to the Gateway IP of the internal network (10.11.12.1).

Access Project -> Network -> Router -> Create Router

Router name: router_public

Enable Admin State: Thicked

External Network: public

Note: The Gateway is automatically created and has an IP assigned from the public network. In this case, the IP is 192.168.1.227. Still, remains to be created the Interface to the internal network, to route the internal traffic.

Access Project -> Network -> Routers -> router_public -> Add Interface

Note: Here you can see the Gateway that has been created automatically at the creation of the Router.

Parameters for the Interface to the internal network:

Subnet: Select the internal network.

IP Address: 10.11.12.1 (If the Gateway IP of the internal network is already in use, please assign another IP address).

f) Create the Floating IPs that will be assigned to the Compute instances. The Floating IPs will be used to connect the instances to the Outside world.

Access Project -> Network -> Floating IPs -> Allocate IP To Project. The IP will be allocated from the pool mentioned in the local.conf file: 192.168.1.224/27.

After clicking the Allocate IP button, a Floating IP will be automatically allocated.

g) Create the Compute instance.

Access Project -> Compute -> Images -> Launch

In the Details section:

- Instance Name: compute_2
- Availability zone: nova
- Count: 1

Note: To make sure that on Create New Volume Tab, the button No is selected. So, the volume assigned from the selected Flavor can be used. Otherwise, the image might not be created.

In the Flavor Section, select *m1.tiny* (any type of image can be selected based on the power provided by the host machine).

In the Network Section, select the internal network.

After the above steps, Launch Instance button is clicked to create the Instance.

In order to see the created Compute instance, access Project -> Compute -> Instances.

h) Assign Floating IP to be able to connect the created instance to the outside world (public network).

Access Project -> Compute -> Instances. And from the Create Snapshot dropdown list, select Associate Floating IP.

IP Address: Select the Floating IP Address that was allocated at Step 11.

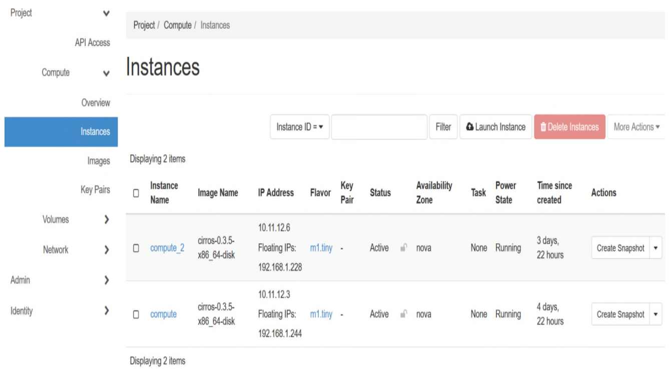
The Port to be associated is represented by the internal network IP address, which is assigned to the compute_2 instance that was created at Step 12.

After these steps, click the button called Associate.

Note: Now, the Floating IP is associated to the Compute instance and the link with the public network is done.

The process is complete. A working and public network connected Compute instance has been created.

In order for the environment to be complete, create another Compute Instance following the same steps.



Displaying 2 items

☐	Instance Name	Image Name	IP Address	Flavor	Key Pair	Status		Availability Zone	Task	Power State	Time since created	Actions
☐	compute_2	cirros-0.3.5-x86_64-disk	10.11.12.6 Floating IPs: 192.168.1.228	m1.tiny	-	Active		nova	None	Running	3 days, 22 hours	Create Snapshot ▾
☐	compute	cirros-0.3.5-x86_64-disk	10.11.12.3 Floating IPs: 192.168.1.244	m1.tiny	-	Active		nova	None	Running	4 days, 22 hours	Create Snapshot ▾

Displaying 2 items

Figure 5. Compute instances

i) Allow Internet Control Message Protocol (ICMP) and Secure Shell (SSH) Ingress traffic to be able to do PING and SSH from the Ubuntu machine towards the OpenStack Compute instances.

Access Project -> Network -> Security Groups -> Manage Rules -> Add Rule.

In the Add Rule Window, for the Rule section, select All ICMP and leave the rest of the sections default. Then, click Add.

Note: To proceed in the same manner for SSH.

G. Testing the IaaS environment

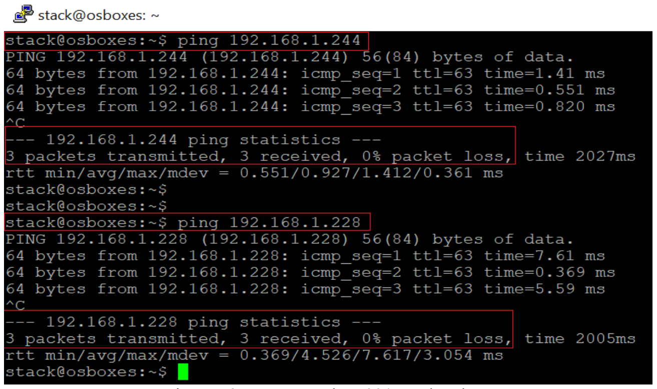
Compute Instance One:

Name: compute

Internal IP: 10.11.12.3
Floating IP: 192.168.1.244

Compute Instance Two:
Name: compute_2
Internal IP: 10.11.12.6
Floating IP: 192.168.1.228

a) Ping the Compute instances from the external network (ISP network in this case). This will be done from the Ubuntu Machine;

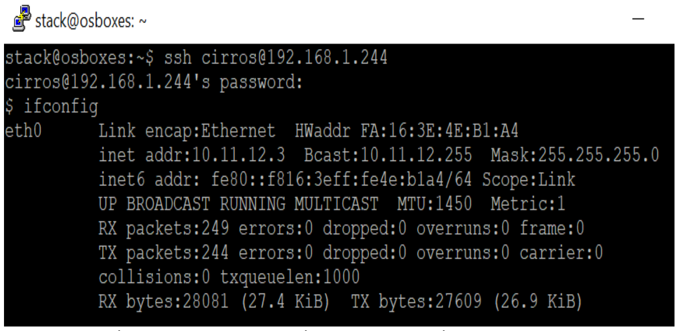


```
stack@osboxes: ~
stack@osboxes:~$ ping 192.168.1.244
PING 192.168.1.244 (192.168.1.244) 56(84) bytes of data.
64 bytes from 192.168.1.244: icmp_seq=1 ttl=63 time=1.41 ms
64 bytes from 192.168.1.244: icmp_seq=2 ttl=63 time=0.551 ms
64 bytes from 192.168.1.244: icmp_seq=3 ttl=63 time=0.820 ms
^C
--- 192.168.1.244 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2027ms
rtt min/avg/max/mdev = 0.551/0.927/1.412/0.361 ms
stack@osboxes:~$
stack@osboxes:~$
stack@osboxes:~$ ping 192.168.1.228
PING 192.168.1.228 (192.168.1.228) 56(84) bytes of data.
64 bytes from 192.168.1.228: icmp_seq=1 ttl=63 time=7.61 ms
64 bytes from 192.168.1.228: icmp_seq=2 ttl=63 time=0.369 ms
64 bytes from 192.168.1.228: icmp_seq=3 ttl=63 time=5.59 ms
^C
--- 192.168.1.228 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2005ms
rtt min/avg/max/mdev = 0.369/4.526/7.617/3.054 ms
stack@osboxes:~$
```

Figure 6. Test Result – 0% Packet loss

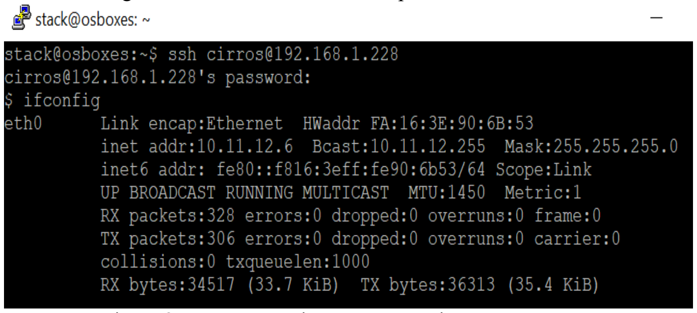
b) Connect to the Compute instances, using SSH, from the external network (this will be done from the Ubuntu Machine).

Use the following commands:
ssh cirros@192.168.1.244 (password is “cubswin:)”)
ssh cirros@192.168.1.228 (password is “cubswin:)”)



```
stack@osboxes: ~
stack@osboxes:~$ ssh cirros@192.168.1.244
cirros@192.168.1.244's password:
$ ifconfig
eth0      Link encap:Ethernet  HWaddr FA:16:3E:4E:B1:A4
          inet addr:10.11.12.3  Bcast:10.11.12.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe4e:b1a4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:249 errors:0 dropped:0 overruns:0 frame:0
          TX packets:244 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:28081 (27.4 KiB)  TX bytes:27609 (26.9 KiB)
```

Figure 7. SSH Connection to Compute instance One

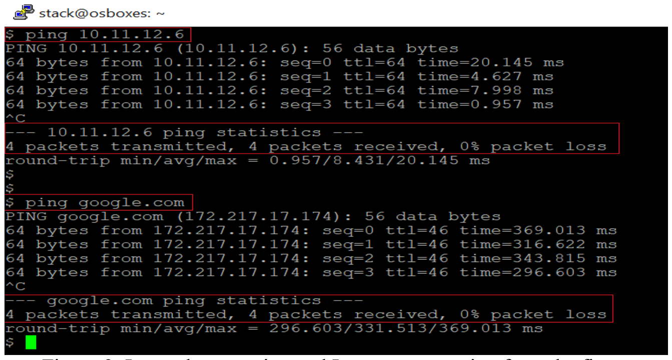


```
stack@osboxes: ~
stack@osboxes:~$ ssh cirros@192.168.1.228
cirros@192.168.1.228's password:
$ ifconfig
eth0      Link encap:Ethernet  HWaddr FA:16:3E:90:6B:53
          inet addr:10.11.12.6  Bcast:10.11.12.255  Mask:255.255.255.0
          inet6 addr: fe80::f816:3eff:fe90:6b53/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1450  Metric:1
          RX packets:328 errors:0 dropped:0 overruns:0 frame:0
          TX packets:306 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34517 (33.7 KiB)  TX bytes:36313 (35.4 KiB)
```

Figure 8. SSH Connection to Compute instance Two

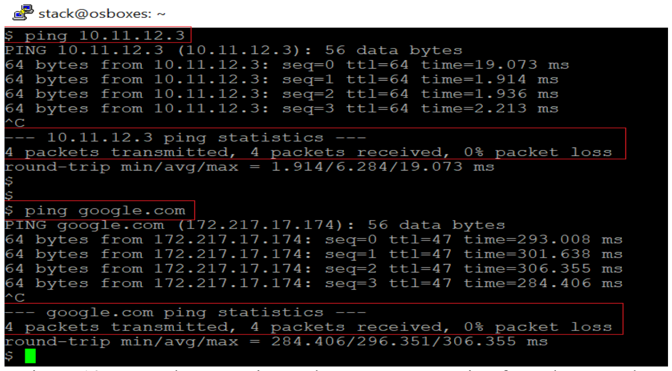
c) Test the connection between the Compute instances and the connection to the Internet.

Remain connected (using the last SSH Connection) on Compute instance One and ping Compute instance Two on the internal IP and then google.com.
Repeat the process being connected on Compute instance Two and ping Compute instance One.



```
stack@osboxes: ~
$ ping 10.11.12.6
PING 10.11.12.6 (10.11.12.6): 56 data bytes
64 bytes from 10.11.12.6: seq=0 ttl=64 time=20.145 ms
64 bytes from 10.11.12.6: seq=1 ttl=64 time=4.627 ms
64 bytes from 10.11.12.6: seq=2 ttl=64 time=7.998 ms
64 bytes from 10.11.12.6: seq=3 ttl=64 time=0.957 ms
^C
--- 10.11.12.6 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.957/8.431/20.145 ms
$
$
$ ping google.com
PING google.com (172.217.17.174): 56 data bytes
64 bytes from 172.217.17.174: seq=0 ttl=46 time=369.013 ms
64 bytes from 172.217.17.174: seq=1 ttl=46 time=316.622 ms
64 bytes from 172.217.17.174: seq=2 ttl=46 time=343.815 ms
64 bytes from 172.217.17.174: seq=3 ttl=46 time=296.603 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 296.603/331.513/369.013 ms
$
```

Figure 9. Internal connection and Internet connection from the first Compute instance



```
stack@osboxes: ~
$ ping 10.11.12.3
PING 10.11.12.3 (10.11.12.3): 56 data bytes
64 bytes from 10.11.12.3: seq=0 ttl=64 time=19.073 ms
64 bytes from 10.11.12.3: seq=1 ttl=64 time=1.914 ms
64 bytes from 10.11.12.3: seq=2 ttl=64 time=1.936 ms
64 bytes from 10.11.12.3: seq=3 ttl=64 time=2.213 ms
^C
--- 10.11.12.3 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.914/6.284/19.073 ms
$
$
$ ping google.com
PING google.com (172.217.17.174): 56 data bytes
64 bytes from 172.217.17.174: seq=0 ttl=47 time=293.008 ms
64 bytes from 172.217.17.174: seq=1 ttl=47 time=301.638 ms
64 bytes from 172.217.17.174: seq=2 ttl=47 time=306.355 ms
64 bytes from 172.217.17.174: seq=3 ttl=47 time=284.406 ms
^C
--- google.com ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 284.406/296.351/306.355 ms
$
```

Figure 10. Internal Connection and Internet Connection from the second Compute instance

All the tests have been successful. The Infrastructure as a Service environment can be used without hesitations for any type of deployment and application development.

V. Practical Hints For Development

This section suggests some practical hints that the user needs to take into consideration when deploying OpenStack.

The following hints should be considered in the development process:

a. When DevStack is used to install and configure OpenStack, one should assure that Ubuntu 16.04.4 is used as Operating System (OS). Even though DevStack supports a wide range of OSs, it is recommended to use the aforementioned OS, because it will run the smoothest and it is the most tested one. This does not mean that it cannot be installed on other OSs, but it might take a little bit more time, due to possible bottlenecks that were not tested before and need a little bit of troubleshooting from the user’s part. Like any other software, OpenStack needs to be tested on each OS and its versions.

b. When installing OpenStack, one should setup a static IP to the host. This IP is going to be embedded throughout the installation, in various locations. This means, that if the user changes the host's IP address, the user will not be able to access anymore the OpenStack Dashboard and its services and it will be very difficult to change the IP due to its wide spreading in the OpenStack environment. Most likely OpenStack will not work anymore and the installation needs to be done from the start.

c. When creating a Compute instance, in the Source section, one should select "No" as answer for the option "Create New Volume". This needs to be done to avoid an error when creating the Volume for the instance. For example, if the user selects YES and also adds a value of 2 GB to the new Volume, but the selected Flavor offers the possibility for only a 1GB Volume, then an error is issued and the Volume cannot be created, due to the fact that it is a contradiction between what the Flavor can offer and the user's selection from the Source section. Better said, OpenStack tries to create something bigger than it can be offered by the selected Flavor.

VI. Conclusion

Based on the work done during installation and configuration of the OpenStack IaaS environment, several conclusions can be drawn.

Taking advantage of OpenStack, the deployment developed in this paper can be easily replicated to another machine/server or it can be scaled to an entire Data Center, with clustering, load balancing and enhanced Disaster Recovery (DR) features, capable of providing cloud services to customers. The enhanced DR capabilities are provided with the help of SDN and NFV functions that offer the possibility to create Availability Zones and Regions, to extend Data Protection in case of a Disaster.

The implementation presented here proved that the users can actually access OpenStack resources and code to improve it or change it according to their needs and then share the results with the entire community for verification and further utilization.

The OpenStack-based system developed here can be a useful choice for users that want to start experiencing the cloud world. It can also be used for academia labs and small enterprises that want to get a competitive and at the same time affordable cloud platform.

Further developments are possible, based on this platform, in SDN and NFV combined environment, e.g. in 5G slicing management control and data planes, etc.

References

[1] What is OpenStack?. [Online]. Available from: https://opensource.com/resources/what-is-openstack 2019.02.15

[2] Introduction to OpenStack. [Online]. Available from: https://docs.openstack.org/ 2019.02.15

[3] Introduction to OpenStack. [Online]. Available from: https://docs.oracle.com/cd/E64747_01/E64749/html/osusg-openstack-what.html# 2019.02.15

[4] How to install OpenStack on your local machine using Devstack. [Online]. Available from: https://www.mirantis.com/blog/how-to-install-openstack-on-your-local-machine-using-devstack/ 2019.02.15

[5] All-In-One Single Machine. [Online]. Available from:https://docs.openstack.org/devstack/latest/guides/single-machine.html 2019.02.15

[6] Loosely coupled architecture of OpenStack. [Online]. Available from: https://www.researchgate.net/figure/Loosely-coupled-architecture-of-OpenStack_fig1_305297793 2019.02.15

[7] How to Install Single Node OpenStack on CentOS 7. [Online]. Available from: https://www.alibabacloud.com/blog/how-to-install-single-node-openstack-on-centos-7_594048 2019.02.18

[8] How to install OpenStack on Ubuntu Server with DevStack. [Online]. Available from: https://www.techrepublic.com/article/how-to-install-openstack-on-ubuntu-server-with-devstack/ 2019.02.18

[9] Companies using OpenStack. [Online]. Available from: https://idatalabs.com/tech/products/openstack 2019.02.18

[10] N. Saranya and S. Nivedha, "Implementing authentication in an Openstack environment-survey" 2016 International Conference on Computer Communication and Informatics (ICCCI) Jan. 2016, pp 1-7, DOI: 10.1109/ICCCI.2016.7479966

[11] E. Luchian, P. Docolin and V. Dobrota, "Advanced monitoring of the OpenStack NFV infrastructure: A Nagios approach using SNMP" 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC) Oct. 2016, pp 51-54, DOI: 10.1109/ISETC.2016.7781055

[12] P. Jain, A. Datt, A. Goel and S.C. Gupta, "cloud service orchestration based architecture of OpenStack Nova and Swift" 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI) Sept. 2016, pp 21-24, DOI: 10.1109/ICACCI.2016.7732425

[13] R. Cohen, K. Barabash and L. Schour, "Distributed Overlay Virtual Ethernet (DOVE) integration with Openstack" 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013) May 2013, pp 1088 – 1089, INSPEC Accession Number: 13684410

[14]DevStack. [Online]. Available from: https://docs.openstack.org/devstack/latest/ 2019.02.20