An Initial Exploration of Multi-Node Open Cloud Infrastructure Testbed

Xiaoming Liu
xliu@semo.edu

Tharinda Embuldeniya
tembuldeniya1s@semo.edu

Ziping Liu
zliu@semo.edu

Zhouzhou Li
zli2@semo.edu

Mario Alberto Garcia
mgarcia@semo.edu

Charles McAllister
cdmcallister@semo.edu
Department of Computer Science

Dana Schwieger
dschwieger@semo.edu
Department of Accounting and Management Information Systems

Southeast Missouri State University
Cape Girardeau, MO 63701-4799, USA

Abstract

Amidst the number of cloud platforms and services available in the market, the setup and the deployment of the cloud infrastructure can be challenging. This paper presents the initial exploration of a multi-node open cloud infrastructure testbed using OpenStack and various applications in the cloud. It is found that the process of selecting the suitable deployment option for OpenStack can vary based on the use-cases. After evaluated multiple deployment approaches using bare-metal provisioning and VM environment, the exploration found that OpenStack Kolla-ansible was the model deployment approach for the cloud testbed. Following the deployment, the proposed cloud testbed is further examined by running general use-cases from a modern database to a computational model on machine learning.

Keywords: Cloud Computing, Open Cloud, Cloud Testbed, OpenStack, Service Deployment, Docker.

1. INTRODUCTION

In this day and age, the cloud has significantly impacted the daily lives of society. It has benefitted society on proliferation and global outreach in content delivery, gaming, research, education, social engagement, etc. Even during natural disasters, the cloud has played a pivotal role in society for allowing businesses to operate while maintaining emergency precautions. Also, the popularity of the cloud has increased due to the progress in technology and higher performance computing needs.

Cloud computing became a universal model for enabling convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services that can be rapidly deployed with less effort from service provider interaction (Mell & Grance, 2011). Amazon has paved the way by introducing the first-ever service provider AWS (Amazon Web Services) (Miller, 2016) to deliver computing and storage services to the public since 2006. Soon after, tech leaders such as Google, Microsoft, IBM, and Rackspace Hosting also emerged to provide cloud services to commercial and public sectors (Khalid, 2010).

According to NIST (National Institute of Standards and Technology), cloud computing breaks down into four infrastructure models, namely private cloud, community cloud, public cloud, and hybrid cloud (Mell & Grance, 2011). There are numerous open-source cloud software readily available that belong to any of the infrastructure models described above. Apache Cloud Stack, OpenNebula, Tsuru, OpenStack, and OpenShift are just a few of them. OpenStack is considered an Infrastructure as a Service (IaaS) level cloud service. This paper selects the OpenStack cloud platform for the proposed multi-node open cloud infrastructure testbed as it takes the lead in reliability when it comes to deploying private cloud infrastructure to the industry (OpenStack, 2019). In the following sections, OpenStack platform and how it is deployed in a multi-node open cloud testbed are described. Section 2 gives an overview of OpenStack services and their deployment methods. Section 3 describes Kolla-Ansible services on deployment. Section 4 demonstrates deploying the multi-node OpenStack cloud testbed using the Kolla-Ansible deployment approach. Section 5 explores running cloud applications on the deployed cloud testbed. And finally Section 6 concludes the paper.

2. OPENSTACK

OpenStack is an open-source software platform for building private or public cloud infrastructure. It is a cloud platform that provides the Infrastructure as a Service (IaaS) level for controlling a scalable pool of resources from computing, storage, and networking (Pepple, 2011).

Overview of OpenStack Services

The OpenStack community has built various components and services that help manage infrastructure resources. Figure 1 shows the core services of OpenStack. Nova is the compute service, and it manages virtual instances by running through a hypervisor that is able to provision virtual machines, bare-metal servers, and system-based containers. Glance is the image service that provides disk-image management and metadata definitions. Cinder is the block storage service, and it provides persistent volumes for Nova virtual machines. Neutron is the networking service and it manages networking services, which provides Internet Protocol (IP) address management, Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), load balancing, network filtering, and access control. Keystone is a vital shared identity service that provides authentication, service discovery, and high-level authorization for managing individual services at OpenStack. Horizon Dashboard acts as the central hub for managing all the OpenStack services. Each of these services is interconnected and depends on each other to manage resources.

Multi-node Architecture

In a multi-node architecture, OpenStack services are distributed equally across available nodes. Multi-node architecture is useful for scaling, load balancing, fault tolerance, and high availability for users. Figure 2 is an example of running OpenStack services on a multi-node architecture solution using three nodes: controller node, compute node and storage node. Each node has its dedicated services.

OpenStack Deployment Method

There are several options for OpenStack to deploy the cloud environment. In our initial exploration of the three-node testbed environment, we experimented the various deployment approaches, with two types of testing environments, namely a bare-metal environment and a virtual machine environment. Some of the OpenStack deployment methods do require access to bare-metal servers for provisioning. Other deployment options can run on any type of

virtual machine that supports a Linux-based KVM (Kernel Virtual Machine) environment. Table 1 shows the deployment outcomes for this test lab environment. As shown in Table 1, Kolla-Ansible was chosen as a suitable candidate for deploying the OpenStack testbed environment for this exploration.

3. KOLLA-ANSIBLE

As described in Section 2, Kolla-Ansible is a solution that uses a container-based deployment approach for the deployment of OpenStack. It has limited functionality with bare-metal provisioning (Open Infrastructure Foundation, 2019). Therefore, it requires deploying the operating system and network configuration manually. In our experiment, we looked into Ansible and Docker dependencies used by the Kolla-Ansible deployment method. As illustrated in Figure 3, in our deployment, a PC running the Ansible service and Docker service was setup for the Kolla-Ansible deployment. Docker pulls and stores all the public container images of OpenStack services and their dependencies from the Docker hub to the deployment PC. Then, Ansible uses Docker modules and Docker API for orchestrating Docker containers. After orchestration, Ansible automates the process by deploying Docker containers into target hosts. This deployment process illustrates how the Ansible and Docker work together in deploying OpenStack using the Kolla-Ansible deployment method.

4. Deployment

For this deployment, the testbed was configured to run the core services of OpenStack, including Heat, Horizon, Nova, and Glance. Additionally, this testbed was configured to include the OpenStack Cinder, a block-storage service for managing the volumes. Figure 4 illustrates the proposed network architecture for the testbed, that is currently running at Southeast Missouri State University (SEMO)'s Cyber-Range facility.

This testbed cloud network architecture is based on the multi-node architecture that was described in Section 2. In the network architecture, each of the nodes and other equipment are connected through the 16 port L2 switch and the router provides a bridge between the OpenStack testbed and the Cyber-Range Firewall. The Cyber-Range firewall provides public network access and the internet to this testbed.

The deployment PC hosts the Kolla-Ansible service when deploying the OpenStack service to the controller node, compute node 1, and compute node 2. The controller node acts as the OpenStack resource manager of the core services of OpenStack, such as Heat, Horizon, Nova, and Glance. There is a specific network configuration required, but no other storage configuration is required for this node. Compute Node 1 and compute Node 2 act as the compute and storage services for the OpenStack. These two nodes host the Nova-compute service and Cinder storage service from OpenStack. There are a distinct network and storage configurations required for these nodes.

The testbed was set to operate within the subnet of 192.168.0.0/24. This subnet is capable of handling 254 hosts in the network. Initially, the first 50 network addresses are set aside for managing the network infrastructure. The rest of the IP addresses are set to run within the OpenStack cloud environment.

Furthermore, all the servers have full internet access to install and update their components and services as needed. As per OpenStack deployment guidelines, each server node must house at least two Network Interface Cards (NIC) for management and external communication purposes. This deployment uses two NICs for each node. The primary NIC is used for the management IP range and the other NIC is configured as a flat network for the OpenStack Neutron service.

The deployment PC and the controller node are set to use the maximum hard disk space for Kolla-Ansible services and OpenStack services, respectively. Both are configured with OS partitions. As for the other two compute nodes, they both are configured to have operating system partitions and an LVM (Logical Volume Management) partition. This LVM partition is used as the storage backend driver for Cinder block storage service.

All the server nodes (except for the deployment PC) are configured to run on Ubuntu 18.04 Server Long-Term Support (LTS) edition from Canonical Ltd. This server operating system (OS) runs on the bash terminal, command-line interface, which would provide the minimum resource overhead towards the host's hardware. As for the deployment PC, it is installed and configured to run on Ubuntu 18.04 Desktop LTS edition. This edition would be more natural to manage network resources within the OpenStack testbed. As a result, the deployment PC acts as the management node for handling the infrastructure resources. Once the OS is deployed, the next step would be to ensure all the latest security patches and updates are installed for all network nodes.

As a summary, Table 2 shows the infrastructure testbed used for this OpenStack deployment.

As described in Section 3, Ansible is an automation tool used by the Kolla-Ansible deployment process. Ansible requires a separate administrative user that can perform remote instructions given by the Ansible-playbook. This deployment uses a single administrative user called "sysadmin" that has access to each node from the deployment PC. An Ansible user can then log in to remote nodes.

The first phase is to install the deployment PC with a Docker container service (Hogan, 2018). The second phase is to install Python and the dependencies for the deployment PC (OpenStack, 2019). The last phase was to install the latest version of Ansible and Kolla-Ansible services on the deployment PC (Elligwood & Hanif, 2018).

There are three stages for the deployment process. The first stage is to run the Ansible-playbook for installing and configuring host-level dependencies such as Docker for all server nodes. The next stage of the deployment process is to verify that all the remote nodes are configured to work with the OpenStack answers file. The final stage of the deployment process is to deploy the OpenStack cloud testbed. During this process, Kolla-Ansible ensures that Docker containers are set to run each OpenStack service.

Once everything is deployed correctly, the cloud testbed would provide the OpenStack Horizon dashboard via the web browser.

5. WORKING WITH OPENSTACK

After the deployment of the OpenStack testbed using Kolla-Ansible, the testbed needs to be configured to operate and use the Nova instances in a typical production environment. The Kolla-Ansible deployment method offers a post-deployment script that allows the configuration of OpenStack services, namely, Nova, Cinder, Glance, Keystone, Heat, and Neutron for admin project at OpenStack (OpenStack, 2019).

After the post-deployment setup is completed, the Horizon dashboard shows the available resources allocated for the admin project of the cloud testbed. Figure 5 shows the admin project limitation for compute, volume, and network usages used by this testbed. Initially, compute service could run 40 instances along with 40 virtual CPUs and 93.3 GB of memory after initial installation. Network service could accommodate up to 50 floating IPs or accessible public IP addresses, 10 security groups, 100 security group rules, 100 network types, 500 network ports, and 10 network routes for this admin project.

The first instance was launched as "FirstVM" using the "cirros" image with an "m1.tiny" flavor and used the "demo-net" as the network type.

The second instance allows installation and running of a TensorFlow model (Sangaiah, 2019). In order to achieve this objective, the instance was launched with 8 VCPU cores, 10GB memory, and 102 GB block storage capacity. This instance ran on the Ubuntu 18.04 cloud server image. The first step was to download and upload the Ubuntu 16.04 cloud server image to Glance. The next step was to create the block storage capacity with 102GB with volume source image as the Ubuntu 16.04 and to upgrade the Ubuntu 16.04 to 18.04 version. Based on the official TensorFlow installation guide (Google, 2016), the TensorFlow library can be installed on a Jupyter Notebook based docker container. Therefore, the next step was to install the Docker service on the virtual instance. After installing Docker, the next step was to download and run the TensorFlow Docker container from the Docker hub. Once the TensorFlow container ran, the next step was to allow TCP port 8888 from the security group. This port is used by Jupyter Notebook software to run as a standalone web application. After creation, open ports were available for access outside the network. These open ports can comprise the whole OpenStack network. Therefore, it was ideal to delete any existing rules that might comprise the network and prevent any unwanted cyber-attacks. Later, new ingress and egress rules for TCP port 8888 were added. Once this setup was completed, TensorFlow could be accessed within the "demo-net" range via any web browser. The "demo-net" range was considered as a virtual private network created by the OpenStack init-runonce script file.

The third instance was to run a Full-Stack Web application with ASP.NET Core as the backend service. Similar to TensorFlow, this instance ran on Ubuntu 18.04 cloud image with 8GB RAM, 4VCPUs, and 80GB non-block storage space. Like the TensorFlow setup in OpenStack security groups, this running instance also required incoming and outgoing access for TCP port 80. Once this setup was complete, the Full-Stack Web application could be accessed via any web browser through the instance's IP address.

In the last instance, MongoDB was installed to test database service. With the current OpenStack architecture, the cloud testbed does not support MongoDB service as a standalone service. Therefore, it required an operating

system such as Ubuntu or CentOS cloud image to act as the host environment for MongoDB service. Like TensorFlow and ASP.NET Core instances, MongoDB instance will accommodate on Ubuntu 18.04 cloud image with 8GB memory, 102 GB block storage space, and 2 VCPUs. Like configuring HTTP's TCP port 80 and Jupyter's TCP port 8888, this instance also required access for TCP port 27017 from OpenStack security groups. MongoDB uses this port for remote access. After the security group setup was completed, the Mongo database could be accessed from any client within the "demo-net" range.

6. CONCLUSIONS

This paper described how the OpenStack core service works, available deployment strategies, and OpenStack dependencies. For the deployment approach, the paper described the Kolla-Ansible deployment option for OpenStack. Kolla-Ansible coordinates with two dependencies, namely Ansible and Docker. Knowing about the Ansible automation tool and the Docker container benefitted the deployment of the cloud testbed on OpenStack. The basis of the current host environment has managed to improvise a multi-node architecture plan that includes one deployment PC, one controller node, and two compute nodes.

After the deployment, the paper described post-deployment set up and launching instances using the OpenStack Horizon Dashboard. Furthermore, the paper described running a TensorFlow instance, ASP.NET Core instance, and a MongoDB instance on the cloud testbed.

7. REFERENCES

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing Recommendations. *National Institute of Standards and Technology Special Publication, 53,* 1-7.

Miller, R. (2016). How AWS came to be. TechCrunch. Retrieved May 6, 2020 from https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/

Khalid, A. (2010). Cloud computing: Applying issues in small business. In 2010 IEEE International Conference on Signal Acquisition and Processing, 278-281.

OpenStack. (2019). Build the future of Open Infrastructure. Retrieved June 18, 2020 from https://www.openstack.org/

Pepple, K. (2011). Deploying OpenStack: Creating Open Source Clouds. O'Reilly Media, Inc., CA.

Open Infrastructure Foundation. (2019). Deploying OpenStack - what options do we have? Streaming Service. https://www.youtube.com/watch?v=8ODdvCogwl8&t=12s

Hogan, B. (2018). How To Install and Use Docker on Debian 9. DigitalOcean. Retrieved from https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04

OpenStack. (2019). OpenStack Docs: Welcome to Kolla-Ansible's documentation, Retrieved June 19, 2020 from https://docs.openstack.org/kolla-ansible/latest/

Elligwood, J., & Hanif, J. (2018). How To Install and Configure Postfix on Ubuntu 18.04, Retrieved June 20, 2020 from https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-postfix-on-ubuntu-18-04

Sangaiah, K. (2019). Deep Learning and Parallel Computing Environment for Bioengineering Systems. Academic Press, MA.

Google. (2016). Docker TensorFlow, Retrieved June 25, 2020 from https://www.tensorflow.org/install/docker

Appendices and Annexures

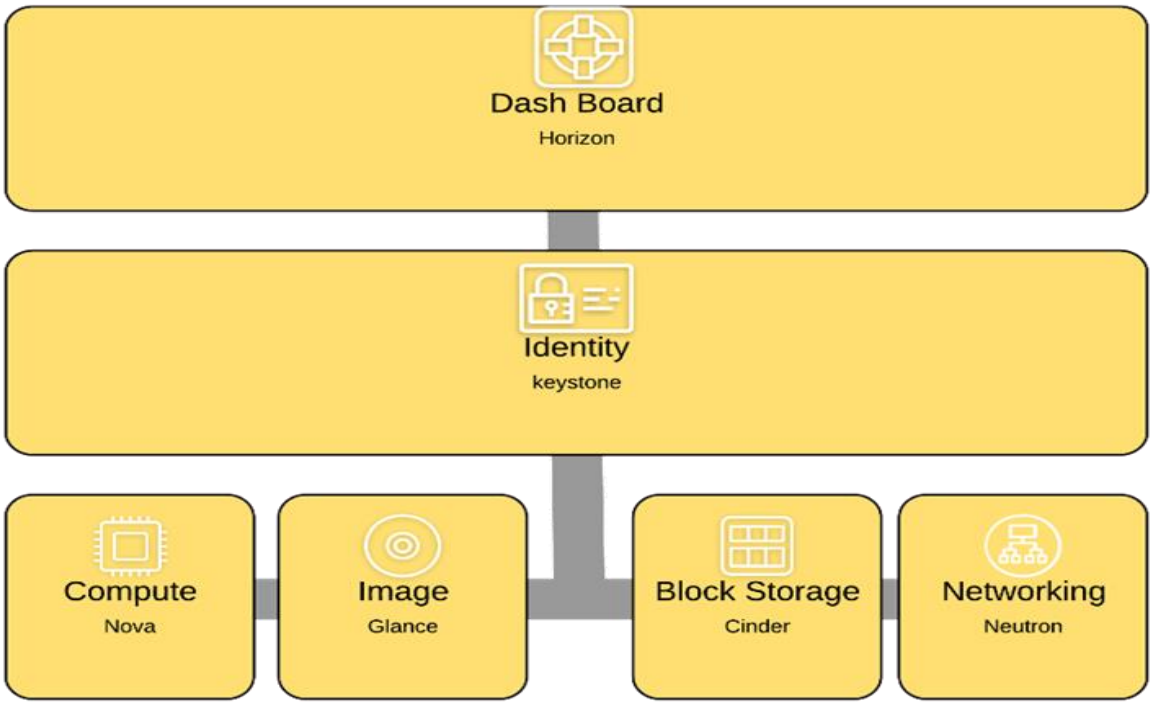

Figure 1. OpenStack Core Services

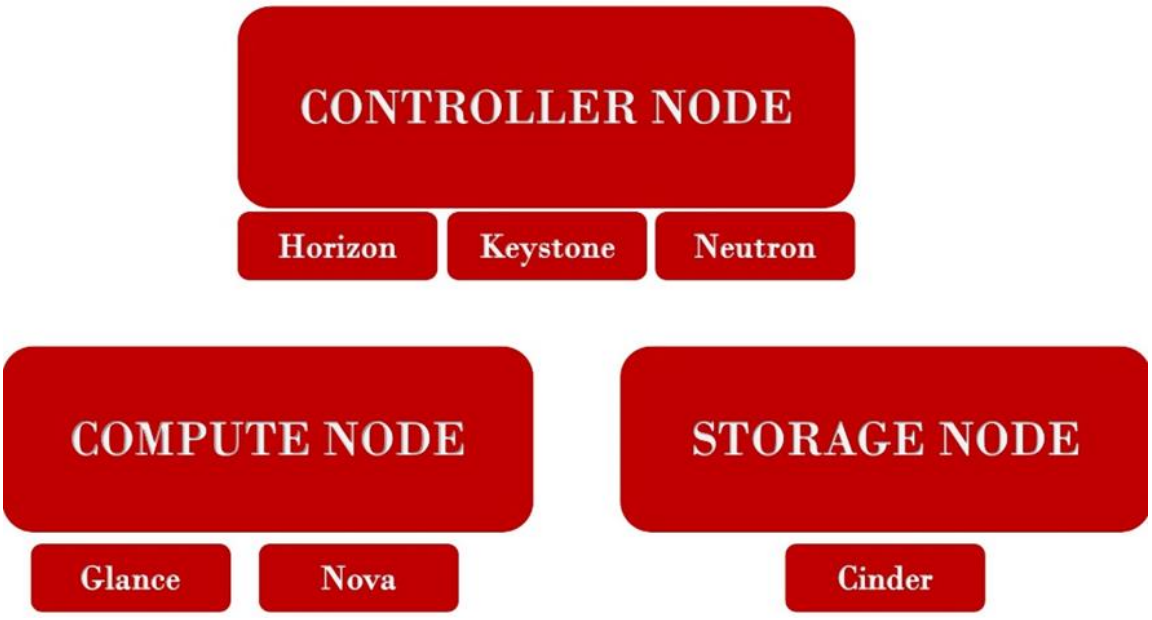

Figure 2. Three Tier Web Architecture

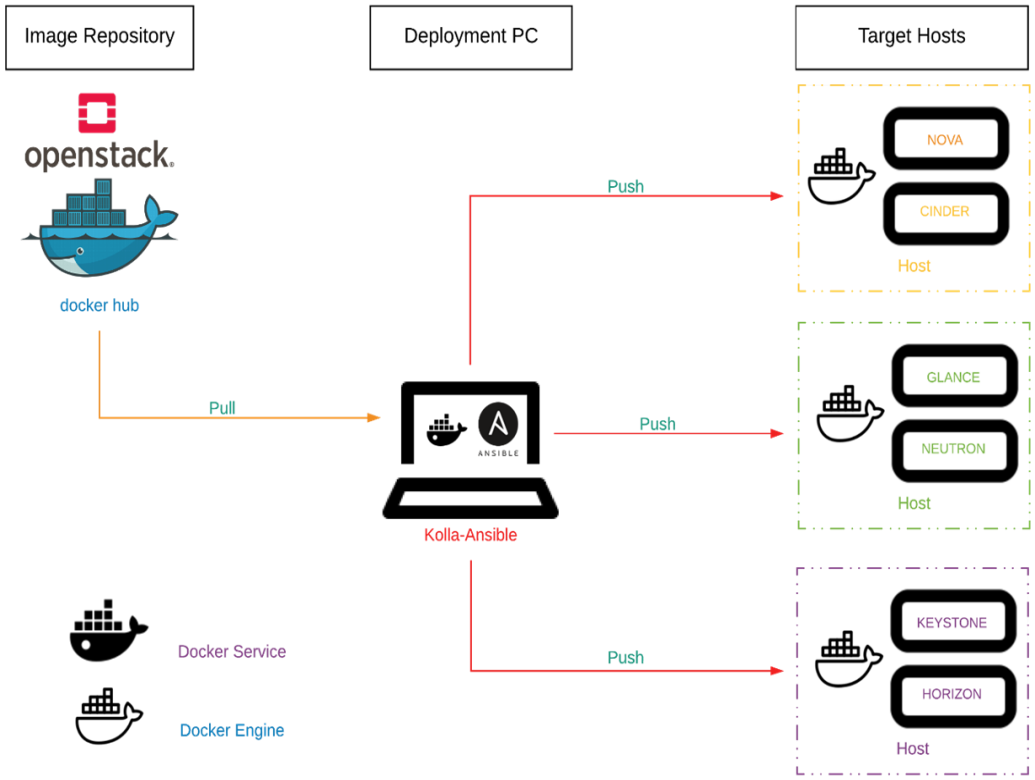

Figure 3. Inner Workings of Kolla-Ansible

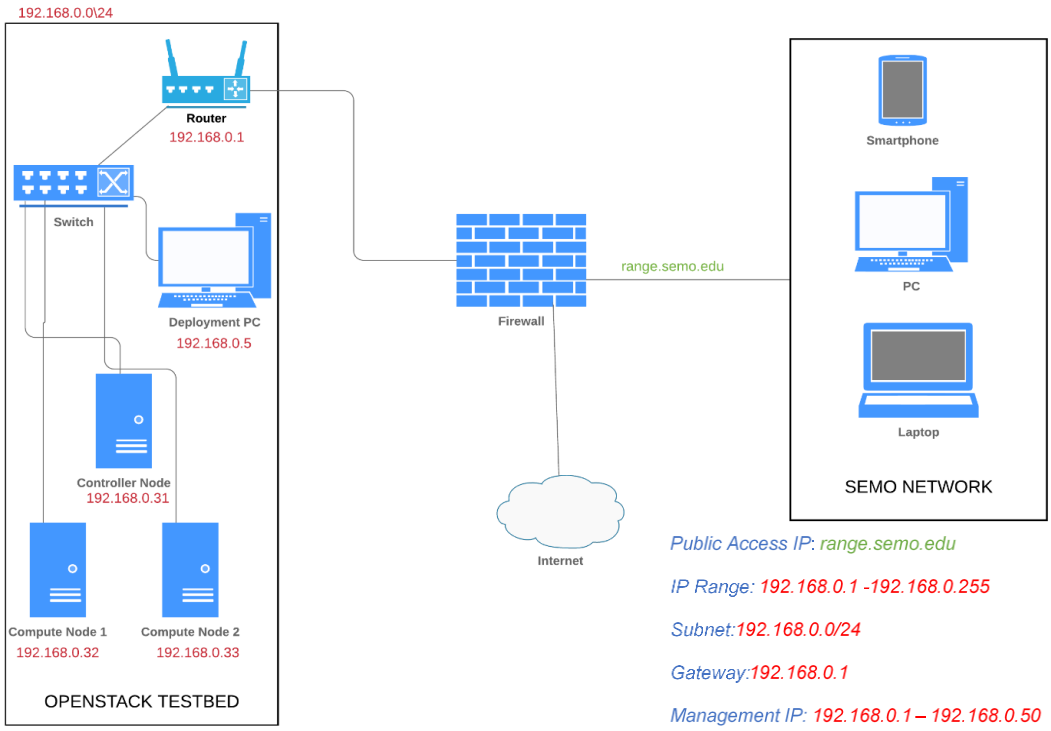

Figure 4. Cloud Testbed Network Architecture

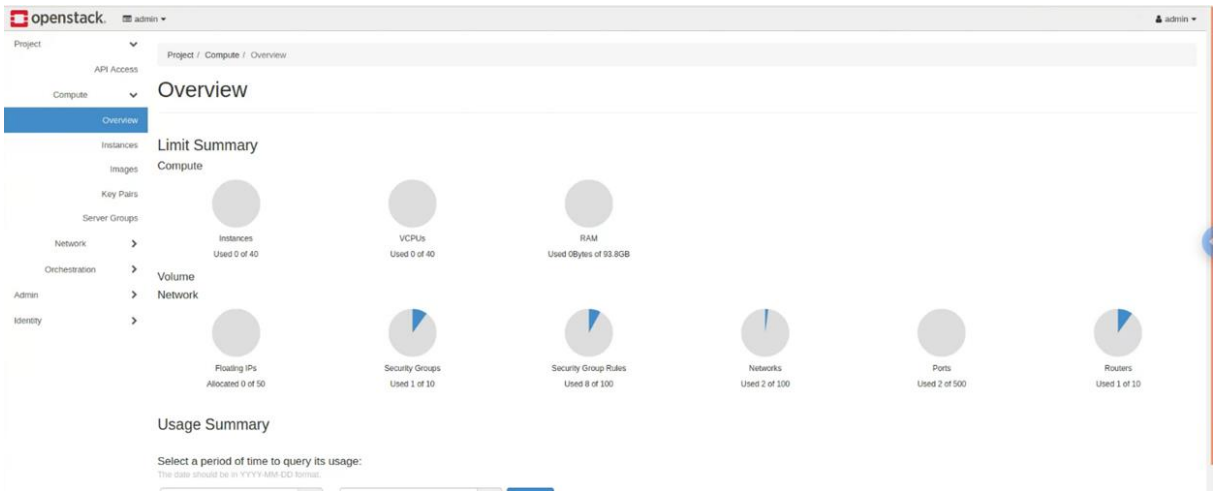

Figure 5. Overview of OpenStack cloud testbed

DEPLOYMENT OPTION	TEST LAB ENVIRONMENT	OUTCOME
MANUAL DEPLOYMENT	Virtual Machines	During the installation of each OpenStack service, there were many configuration errors. There were insufficient materials for troubleshooting with the current version of OpenStack.
DEVSTACK	BareMetal Environment	DevStack works well with the all-in-one architecture. However, network instability and slower response time caused this deployment to fail for multi-node architecture. This deployment was not optimal to work with in the production environment. It was sufficient to learn about OpenStack Services.
OPENSTACK CHARMS	BareMetal Environment	Insufficient resources caused the failure of this deployment. By default, Cinder is using Ceph as a backend drive for deployment. Ceph is a backend drive for managing volume storage requiring at least two disks for each node.
PACKSTACK AND RDO PROJECT	Virtual Machine	Absence of information and not up-to-date with the current version of OpenStack
TRIPLEO		Complex to configure. This deployment requires another OpenStack Environment.
OPENSTACK ANSIBLE	Virtual Machine	Deployment requires extra configuration from the host machine
OPENSTACK KOLLA-ANSIBLE	Virtual Machine	Successfully deployed the OpenStack Environment. This deployment method requires knowledge about Docker and Ansible setup.

Table 1. Deployment Outcomes

NODE NAME	DEPLOYMENT PC	CONTROLLER	COMPUTE NODE 1	COMPUTER NODE 2
HOSTNAME	cssemomgr	cs-semo-controller	cs-semo-compute1	cs-semo-compute2
ANSIBLE USER	sysadmin	sysadmin	sysadmin	sysadmin
CPU	2 Cores	8 Cores	16 Cores	16 Cores
RAM (MEMORY)	8192 MB	16384 MB	32768 MB	32768 MB
HARD DISK (SINGLE DISK)	250GB	1TB	1TB	1TB
PARTITION MAP	Only OS partition	Only OS partition	OS partition and 1 LVM partition	1 root partition and 1 LVM partition
NIC 1(ENO1)	192.168.0.5	192.168.0.31	192.168.0.32	192.168.0.32
NIC 2(ENO2)	none	FLAT	FLAT	FLAT
OPERATING SYSTEM	Ubuntu 18.04 Desktop LTS	Ubuntu 18.04 Server LTS	Ubuntu 18.04 Server LTS	Ubuntu 18.04 Server LTS

Table 2. Summary of Hardware Configuration