ISBN: 978-628-96613-1-6. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LACCEI2025.1.1.2392

Platform as a Service (PaaS) Private Cloud Implementation with OpenStack

Nia Newell[1]; Joshua Waithe[2]
Faculty Mentor: Aparicio Carranza[3], PhD
Co-Mentor: Harrison Carranza[4], MS
[1,2]*Vaughn College of Aeronautics and Technology, USA,* Nia.Newell@vaughn.edu, Joshua.Waithe@vaughn.edu
[3]*Vaughn College of Aeronautics and Technology, USA,* Aparicio.Carranza@vaughn.edu
[4]*Bronx Community College - CUNY, USA,* Harrison.Carranza@bcc.cuny.edu

Abstract– OpenStack is a library of open source software projects, developed to be integrated forming the foundations of Cloud Computing platforms for both Private and Public cloud distributions. Cloud computing relies on remote servers hosted on the Internet rather than a local server/computer. OpenStack consist of many different modular core segments (Nova, Neutron, Glance, Keystone, Horizon, and Ceilometer). Neutron is the networking service which implements services and associated libraries to provide an on-demand, scalable and technology-agnostic network abstraction layer. In this paper we compare and contrast the advantages and disadvantages of implementing OpenStack on a personal computer Virtual Machine (VM) using Ubuntu Server OS. Our main focus is on Neutron (network service) in OpenStack and their limitations.

Keywords-- Cloud Computing, Neutron, OpenStack, Physical Machine, Virtual Machine.

I. Introduction

Before cloud computing, networks were set up in the traditional physical sense with servers, switches, routers, storage systems and security appliances. To create an infrastructure powerful enough for large computing networks, large-scale physical networks needed to be created and maintained. Cloud computing allows us to create large-scale networks without the need of a large physical network. Cloud computing is the practice of using various network of remote servers hosted on the Internet rather than on a physical network. Cloud computing can be implemented in three different broad service categories: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) and *Software as a Service* (SaaS). Infrastructure as a Service is a form of cloud computing that provides virtualized computing resources over the Internet [1]. Some IaaS providers are IBM Cloud Platform, Rackspace Open Cloud, Google Cloud Computing (GCC), and Amazon Web Services (AWS).

While users are able to deploy virtualized desktops on a policy-based service with dynamic scaling and automation of administrative tasks; Platform as a Service is a model that delivers applications over the Internet [2]. Google App Engine, Heroku, Mendix, and Appear IQ are examples of PaaS. The users use the platform through a web browser interface while the PaaS companies will host the hardware and software on its own infrastructure. Software as a Service is a software distribution model in which a third party provider hosts applications and makes them available to customers over the Internet [3]. Examples of SaaS are Google Apps, Salesforce, Oracle, SAP and Microsoft. OpenStack is mostly deployed as an IaaS. Users either manage it through a web-based dashboard, through command-line tools, or through a RESTful Application Programming Interface (API).

OpenStack began in 2010 as a joint project of Rackspace Hosting and NASA. Since it is an open source code, OpenStack is software as defined by the company "*a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface*" [4]. OpenStack consist of many different modular core segments (Nova, Neutron, Glance, Keystone, Swift, Cinder, Horizon, and Ceilometer). Each segment provides a certain function or service. The services separate into different groups, which are storage, shared services, high-level services and general services.

General services are Horizon (*Dashboard*), Nova (*Compute*), and Neutron (*Networking*). The storage component are Swift (*Object Storage*) and Cinder (*Block Storage*). Shared services on OpenStack include Keystone (*Identity Service*), Glance (*Image Service*), and Ceilometer (*Telemetry*). OpenStack considers heat (*Orchestration*) and Trove (*Database Service*). Nova is the OpenStack compute service and is the primary source or brain of OpenStack that allows users to create and manage large numbers of virtual machine servers and other instances to handle computing tasks. Horizon is a dashboard providing a web-based portal to interact with all the OpenStack services. Neutron is OpenStack Networking backbone providing network connectivity for OpenStack and its many services in a simple and efficient manner [5]. Swift is an storage system for object and files, unstructured data object via RESTful.

Cinder is a block storage that allows persistent running of instances. This provides a more traditional way of accessing files and allows the creation and management of block storage devices. Keystone is the identity service showing a catalog of endpoints for all OpenStack services, a multiple means of access. Ceilometer is a telemetry service, a service that is the automatic measurement and wireless transmission of data from a remote source. Glance stores, retrieves "*images*" or virtual copies of hard disk, and allows the images to be used as templates when making new virtual machines. Heat orchestrates multiple composite cloud applications using either HOT (*Heat Orchestration Template*) or AWS Cloud Formation template format through both REST API and a Cloud Formation-compatible Query API. Trove is a database as a service designed to run entirely on OpenStack with the functionality for both relational and non-relational database engines [6]. This conceptual architecture is shown in Fig. 1.

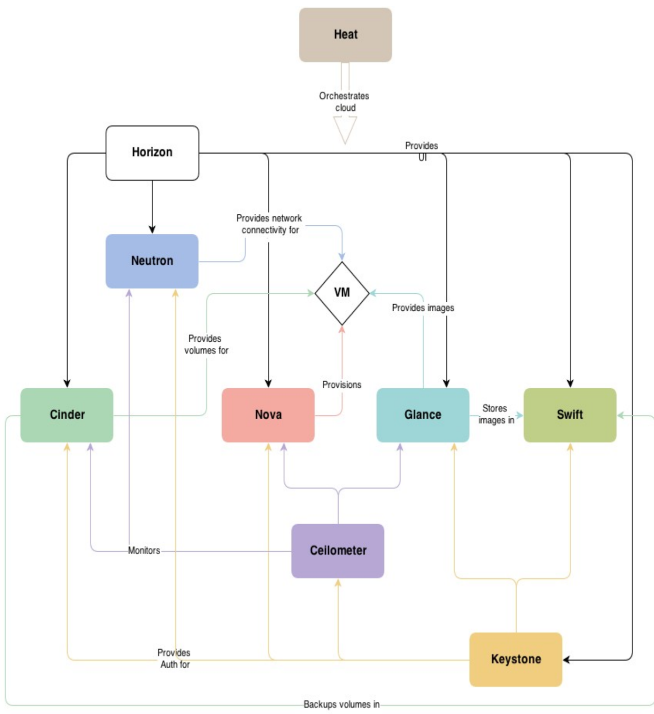

Fig. 1 Conceptual Architecture Of OpenStack

Neutron offers an API for users to define or customize networks and the attachments with a pluggable architecture that supports many popular networking vendors and technologies. It supports the OpenFlow software networking protocol and plugins. OpenFlow is an open standard way to send and receive data. It is an standalone service that can be installed and configured independently by multiple or single host. Neutron provides a framework for users to build on and enhance the capabilities of the cloud with functionalities such as switching, routing, load balancing, firewalling, and virtual private networks.

II. Theoretical Framework

Neutron is a standalone service that deploys with several sub processes across a number of nodes, as seen in Fig. 2. These processes communicate within the Neutron Umbrella as well as with other OpenStack services. The main process is the neutron-server; a Python daemon that exposes Neutron is an API and passes a tenant requests to a suite of plug-ins for additional processing. The components in Neutron are neutron server, plugin agent, Dynamic Host Configuration Protocol (DHCP) agent, L3 agent, and network provider services. Neutron server (*neutron-server and neutron-*-plugin*) runs on the network node to service the Networking API and its extensions. It also enforces the desired network model and IP addressing of each port given deployment specification. The neutron-server and plugin agents requires access to an SQL Database for persistent storage and access to a message queue for inter-communication between the L3 agent, DHCP agent and the Plugin agent. Plugin agent (*neutron-*-agent*) runs on each compute node to manage local Virtual Switch (vSwitch) configuration. The plug-in that is used determines which agents are run.

This service requires message queue access and depends on the plugin used. Some plugins like Open Daylight (ODL) and Open Virtual Network (OVN) do not require any python agents on compute nodes. DHCP agent (*neutron-dhcp-agent*) provides DHCP services to tenant networks. This agent is the same across all plug-ins and is responsible for maintaining DHCP configuration. The *neutron-dhcp-agent* requires message queue access, optionally depending on plug-in. L3 agent (*neutron-l3-agent*) provides L3/NAT forwarding for external network access of VMs on tenant networks, requires message queue access. This option depends on plug-in. Network provider services.

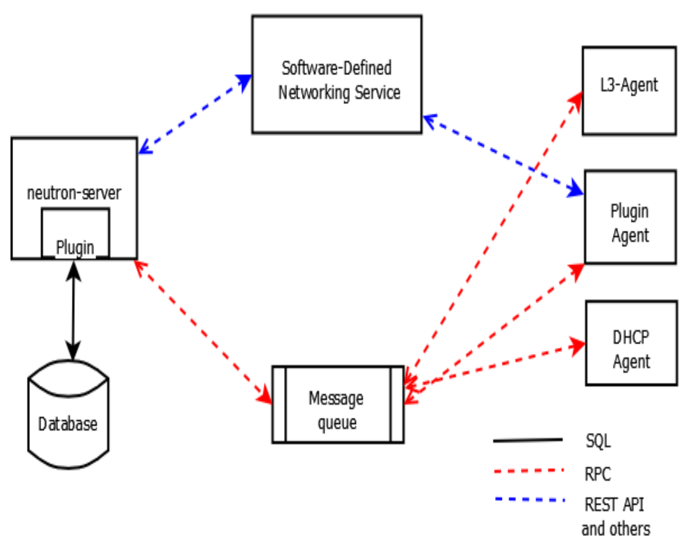

Fig. 2 Architectural and Network Flow Diagram of OpenStack Network Components

Software Defined Networking (SDN) server/services provides additional networking services to tenant networks. These SDN services may interact with neutron-server, neutron-plugin, and plugin-agents through communication channels such as REST APIs [7]. Neutron allows configuration of multi-tier network topologies with advanced network capabilities to build advanced network services that can plug into the OpenStack network with a GUI support (*Horizon Dashboard*) and API extensibility framework.

Ubuntu is an open source Debian based Linux Operating System that offers Ubuntu cloud images that are pre-installed disk images, which have been customized by Ubuntu to run on cloud platforms similar to OpenStack. Minimum of five machines, a switch, and router are required to deploy an Ubuntu OpenStack cloud with one being for the Metal as a Service (*MAAS*) server that treats physical servers as virtual machines to allow flexibility of managing and total automation over physical servers as a cloud like resource. A machine for OpenStack Autopilot, a dynamic OpenStack that allows easy installation, expansion, and everyday operations. The other three servers are used for the cloud with a dedicated switch for a private cloud LAN and router to access the Internet [8]. Since we do not have the required equipment, we opted for a single server with a 16 GB of RAM to install and deploy OpenStack.

III. METHODOLOGY

We first downloaded Ubuntu Server 22.04 LTS Server as the basis for OpenStack to work on a clean hard drive. Once Ubuntu has been installed, and a network system is setup for Internet access we proceed to install OpenStack. In the command line, the server is updated and upgraded to make sure it is up-to-date. Then we install ZFS (*combined file system and logical volume manager*), and LXD (*container hypervisor*), as seen in Figs. 3 and 4.

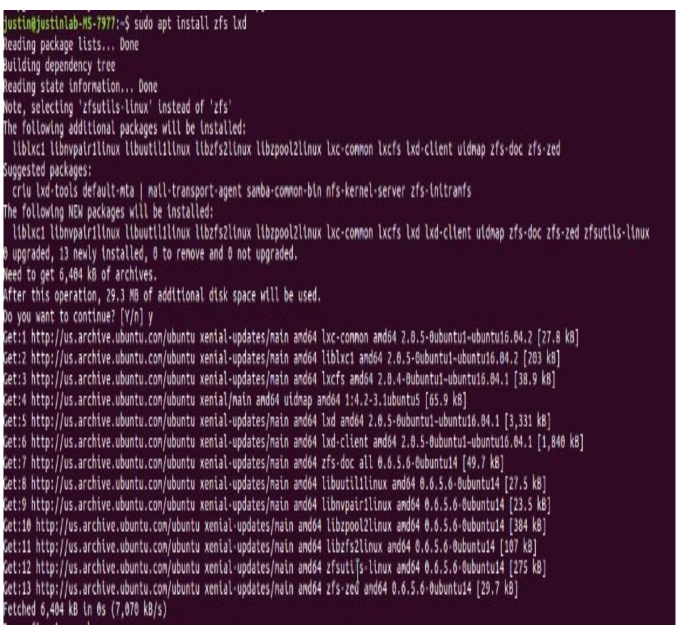

Fig. 3 Installing ZFS

ZFS is a highly resilient file system that is simple to configure and manage with strong redundancy with no additional software that integrates volume management features for large amounts of data with a set of data services and supports a variety of storage protocols. ZFS allows Cinder (*OpenStack's block storage*) and Swift (*object storage*) to be used if needed [9]. LXD is a container "*hypervisor*" with three components LXD, LXC and *nova-compute-lxd*. LXD is a system wide daemon that exports a REST API locally and over the network. LXC is the command line client that handles and connects to multiple container hosts over the network. *Nova-compute-lxd* is an OpenStack nova plugin that allow us to use LXD hosts as compute nodes and run workloads on containers rather than virtual machines that allows more scalability [10].

Fig. 4 Configuring LXD

After installing ZFS LXD a reboot of the computing system is required in order for the configuration of the LXD using ZFS to function as needed. Furtheremore, we configure the package to a network bridge which connects the LXD for outside network connectivity. Then, we generate a client certificate and add a repository to conjure up spells, also known as software packages, as seen in Figs. 5, 6 and 7. Juju must also be installed onto the server. Juju is an application as well as a service. This particular service is Ubuntu's unique distribution of the Kubernetes open-source system which provides the ability to quickly model, configure, deploy, and manage containerized applications, in an on-premise, hybrid or even public cloud environment. With juju, users can efficiently manage the lifecycle of applications, enabling quick and seamless orchestration and automation of container-based workloads across diverse infrastructurre. In the case of this particular project, only a few commands are necessary for OpenStack via conjure-up [11]. After the repository and juju has been installed, the system needs an update to ensure all components in Ubuntu are up to date.

Fig. 5 Installation og Conjure-up

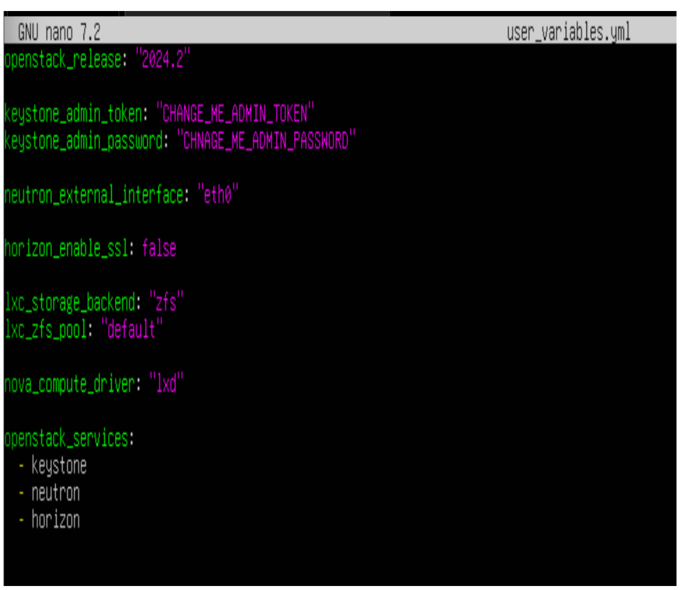

Fig. 6 Configuration of user_variables.vml file

Fig. 7 Install JUJU/devel

Before the installation of OpenStack, conjure-up needs to be installed. Conjure-up command line tool, spanning Juju, MAAS and LXD allows you to summon OpenStack. It provides an easy solution of processing scripts that gives the flexibility for network interfaces to Neutron services, import images into Glance and more [12]. We conjure-up OpenStack and launch the ncurses interface to setup OpenStack. Ncurces is freely distributed type of cursor optimization library that manages an application's display on character-cell terminals [13]. We Select the local host as our cloud and choose OpenStack with Nova-LXD as our spell. Nova-LXD provides a Nova driver for managing full system containers using LXD as part of an OpenStack cloud [14], [15]. The system needs Ceph cluster; a distribution storage and network file system. We decided to deploy all 14 remaining applications with bundle defaults and go through the process of importing services and instances of OpenStack to the server, as seen in Fig. 8.

Fig. 8 Installing Services for Open Stack

Once the machines installed the Horizon, to view the configuration summary, make sure all steps are checked. Also make sure to access Horizon in the server and the outside world.

Fig. 9, shows the important parts taken, it shows the image of the machine. It allows us to have access to the horizon with these two package been installed we are able to access these two machine locally from our system. In addition, another import part is the SSH key pairs which allows to have communication between nodes.

Note that using a different node table we will not be able to connect to different node locally. We were also able to configure neutron, this is where we configure the edge router to be able to connect to the horizon that gives the dash permission to access the internal resources, as shown in Fig. 10.

Fig. 9 Summary of Configuration

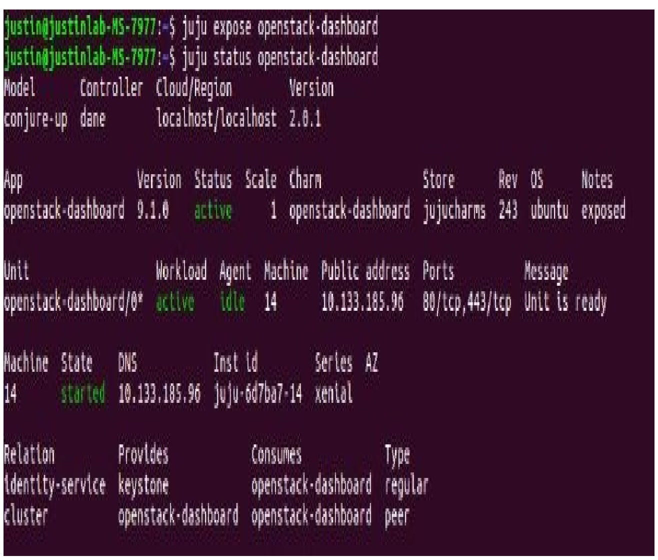

Fig. 10 Horizon access

When all containers are running OpenStack, we can then login to the Ubuntu OpenStack dashboard and view all network information for the cloud.

IV. RESULTS

For the initial deployment and configuration of OpenStack, a container-based variant was the best choice. The particular deployment required the use of a tool called "*Conjure-up*". This particular tool allows us to deploy a "*big-software*" stack via the use of a "*spell*". Conjure-up is a thin-layer that uses juju, MAAS and LXD as the fundamental underlying technologies to perform the intended functions. As seen in Fig. 11, a local deployment requires installation of ZFS, LXC, juju and conjure-up packages.

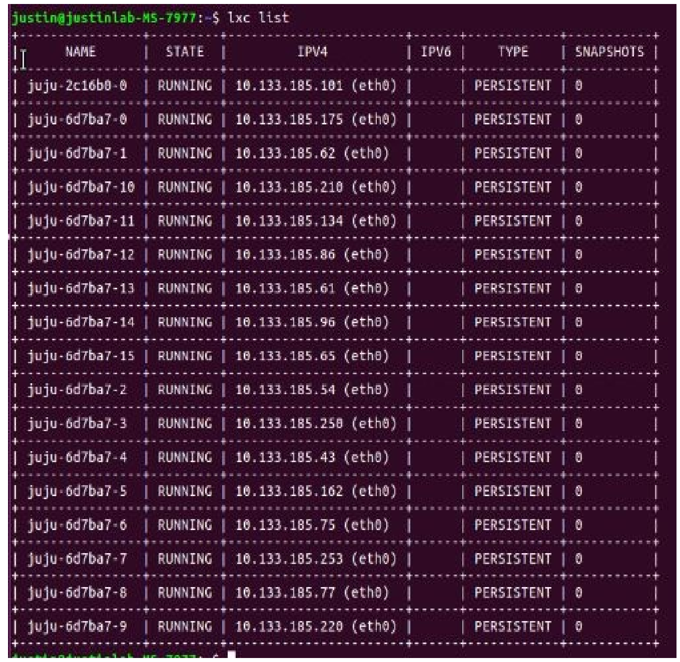

Fig. 11 JUJU instances

The installation of ZFS provides the necessary file system where LXD can be configured and deployed. Eventually juju is used to model, configure, deploy and manage the service (*machine instances*) necessary for an OpenStack deployment. When fully configured the machine instances that are created using during this process, network wise, are managed by the Neutron project where all aspect of the network infrastructure is software defined. Figs. 12-to-19 show the results of our implementations.

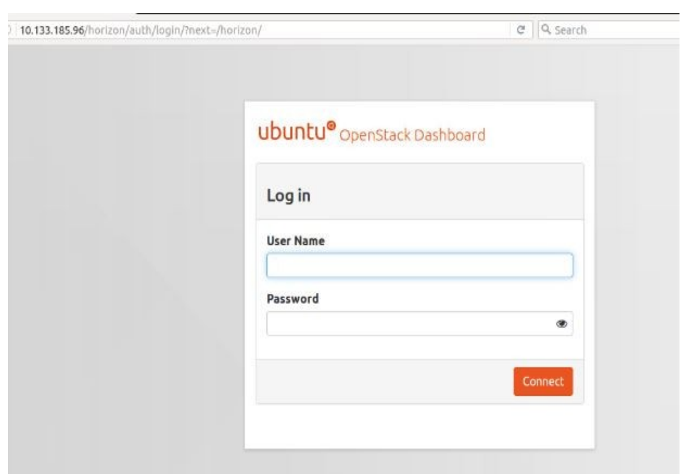

Fig. 12 Open Stack login

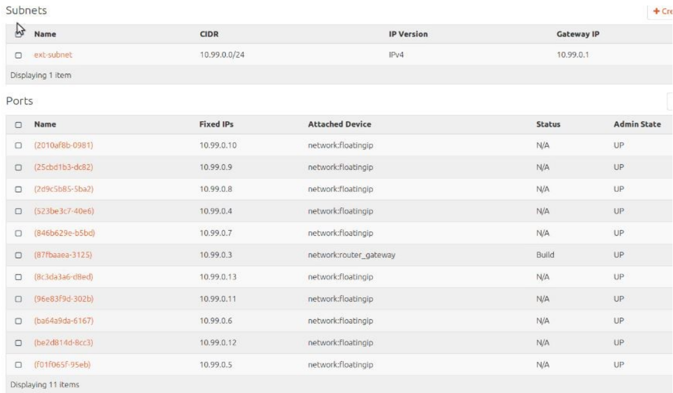

Fig. 13 Network Ports

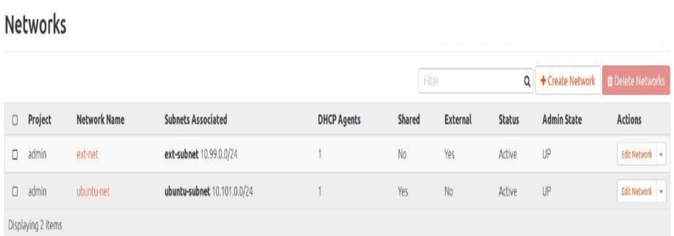

Fig. 14 View of Networks

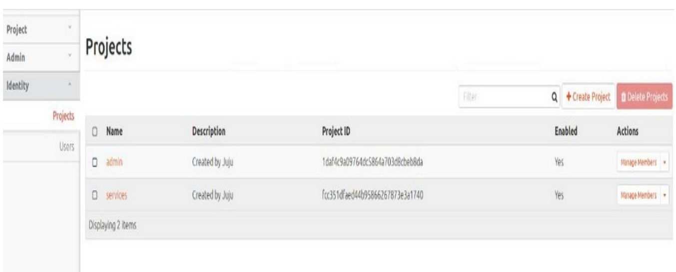

Fig. 15 View of Dashboard

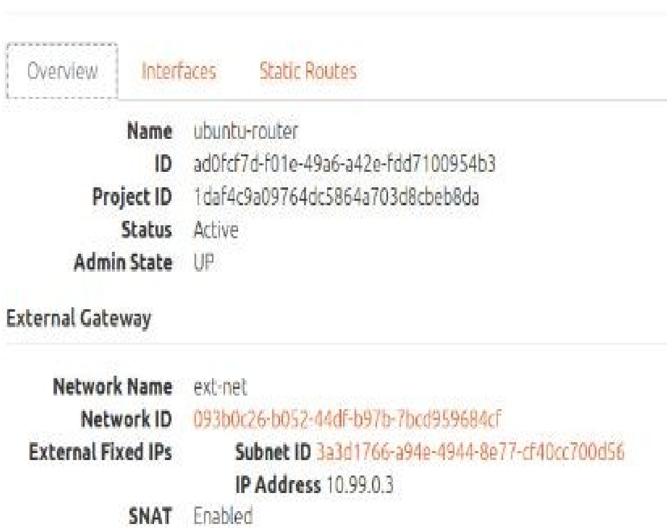

Fig. 16 Status of DHCP Agents

Fig. 17 Router Overview

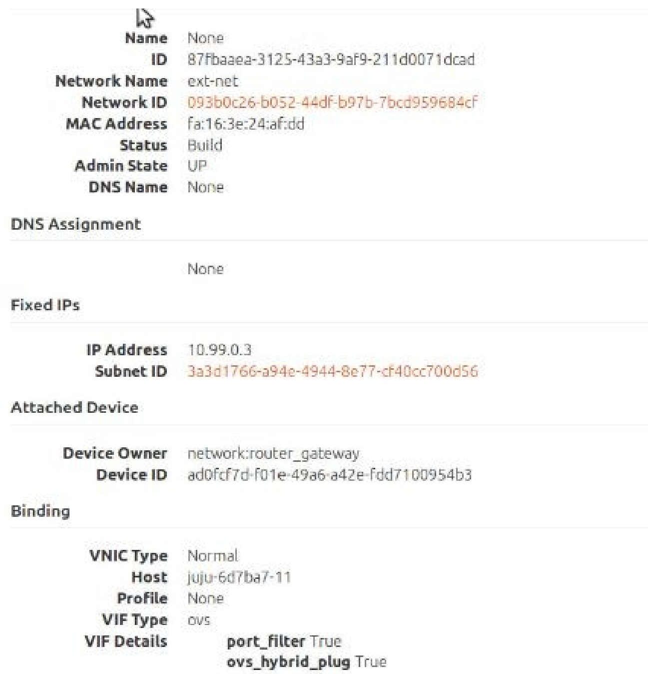

Fig. 18 External Gateway Information

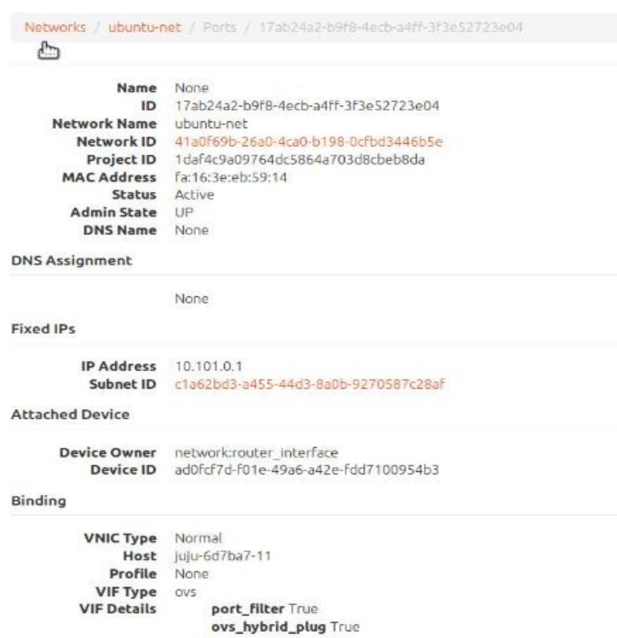

Fig. 19 Network Router Interface

V. CONCLUSION

Cloud architecture has become a corner stone of the world modern technology infrastructure. As the cloud becomes increasingly vital, services that once required dedicated physical appliance to function are now virtualized (*software defined*) due to easy of management and cost reduction the cloud ecosystem offers. The resulting virtualization of said services that once communicated via the use of physical networking infrastructure now uses software solution such as Neutron to maintain the networking requirements necessary for said services to function. In the case of this project, Linux containers and juju (Canonical version of Kubernetes) are the means in which OpenStack is successfully deployed to evaluate the Neutron portion of the stack. However, this particular deployment is mostly limited due to the resources limitation of containers. This is expected considering conjure-up was designed to be turnkey streamline deployment of OpenStack.

ACKNOWLEDGMENT

This research project has been supported by the grant: DOE Title V P031S200139

REFERENCES

[1] OpenStack Architecture Design, https://docs.openstack.org/arch-design
[2] Welcome to OpenStack Documentation, https://docs.openstack.org/2024.2/
[3] *OpenStack.* Wikipedia, https://en.wikipedia.org/wiki/OpenStack
[4] Cameron Hashemi-Pour, Stephen J. Bigelow. *Infrastructure as a Service(IaaS).* TechTarget. January 2025. https://www.techtarget.com/searchcloudcomputing/definition/Infrastructure-as-a-Service-IaaS
[5] Wesley Chai, Kate Brush, Stephen J. Bigelow. *Platform as a Service (PaaS).* TechTarget. January 2025. https://www.techtarget.com/searchcloudcomputing/definition/Platform-as-a-Service-PaaS
[6] Kinsa Yasar, Wesley Chai, Kathleen Casey. *Software as a Service (SaaS).* TechTarget. January 2025. https://www.techtarget.com/searchcloudcomputing/definition/Software-as-a-Service
[7] Zitzman, Sharone. *OpenStack Wiki in Short - A Quick Guide to Open Cloud.* Cloudify. July 18, 2014. http://getcloudify.org/2014/07/18/openstack-wiki-open-cloud.html.
[8] *Network Architecture.* February 2025. https://docs.openstack.org/security-guide/networking/architecture.html
[9] Ubuntu. *Get OpenStack Autopilot.* Ubuntu. 2025. https://www.ubuntu.com/download/cloud.
[10] *How to Build OpenStack Block Storage on ZFS.* https://docs.openstack.org/cinder/queens/configuration/block-storage/drivers/zfssa-iscsi-driver.html.
[11] *What's LXD.* https://documentation.ubuntu.com/lxd/en/stable-5.21/
[12] *The Canonical Distribution of Kubernetes.* 2025. https://charmhub.io/
[13] McCracken, Adam Stokes and Mike. *conjure-up User Manual.* Oct 2016. http://conjure-up.io/docs/en/users/#getting-started
[14] Padala, Pradeep. *NCURSES Programming HOWTO.* 2001. https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/intro.html.
[15] *Installing and configuring Nova LXD.* https://askubuntu.com/questions/858209/how-to-install-and-configure-nova-lxd-on-existing-openstack