

PROFESSIONAL TRAINING REPORT

Sathyabama Institute of Science and Technology (Deemed to be University)

Submitted in partial fulfilment of the requirements for the award
of Bachelor of Technology Degree in Information Technology

By

SELSYA JENIFER L

(40120083)



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

**SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI, TAMILNADU – 600119

April 2023



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

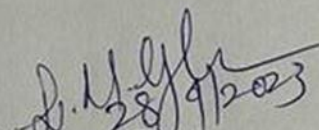
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

(www.sathyabama.com)

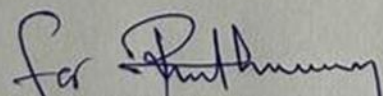
DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

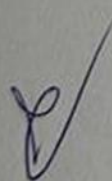
This is to certify that this Project Report is the bonafide work of the **Selsya Jenifer L (Reg. No. 40120083)** who carried out the project entitled as **"Create and Launch an Instance based on the 'CirrOS' Image"** under our supervision from January 2023 to April 2024.


Internal Guide

Dr. L. Mary Gladence, M.E., Ph.D.,


Head of the Department

Dr. R. Subhashini, M.E., Ph.D.,



Internal Examiner



External Examiner

DECLARATION

I SELSYA JENIFER L (Reg. No. 40120083) hereby declare that the Project Report entitled "Create and Launch an Instance based on the 'CirrOS' Image" is done by me under the guidance of Dr. L Mary Gladence M.E., Ph.D., and is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 28.06.23

PLACE: Chennai

Selsya Jenifer L

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to express my sincere thanks to the **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T Sasikala, M.E., Ph.D.**, Dean, School of Computing **Dr. R Subhashini, M.E., Ph.D.**, Head of the Department of Information Technology for providing me with necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L Mary Gladence, M.E., Ph.D.**, for her valuable guidance, suggestions and constant encouragement that paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

CERTIFICATE OF COMPLETION

Ref.No.: 16 /FSPCC/BC001/04/2023



इलेक्ट्रॉनिक्स एवं
सूचना प्रौद्योगिकी मंत्रालय
MINISTRY OF
ELECTRONICS AND
INFORMATION TECHNOLOGY



COURSE COMPLETION LETTER

It is certified that

Selsyajenifer L

Sathyabama Institute of Science and Technology

has successfully undergone the Bridge Course Training Program on

Cloud Computing Technology

Under "FutureSkills PRIME" Programme

commenced from 11 Mar, 2023

Shri. V. Sridharan
Chief Investigator
Cloud Computing Technology
Lead Resource Centre

L R PRAKASH Digitally signed by L R PRAKASH
Date: 2023.04.26 02:42:31
+05'30'

Cdr.(Retd.) L.R. Prakash
Senior Director
Centre Head
C-DAC, Chennai

ABSTRACT

OpenStack is a popular open-source cloud computing platform that allows users to deploy and manage virtual machines and other resources in a scalable and flexible manner. To create and launch an instance in OpenStack, one needs to choose a base image to use as a template for the instance. The 'cirros' image is a lightweight and easy-to-use image that is often used for testing and experimentation purposes.

To create and launch an instance based on the 'cirros' image, the user first needs to find the image ID or name in the OpenStack dashboard or CLI. Once the image is identified, the user needs to select a hardware configuration, or flavor, for the instance. The flavor determines the amount of CPU, memory, and disk space allocated to the instance. After selecting the flavor, the user can create the instance by specifying the image ID or name and the flavor.

Once the instance is launched, the user can log in to it using SSH and perform any necessary configurations or tasks. The 'cirros' image is a minimal image that includes only essential components, so users may need to install additional software or packages as needed. When the user is finished with the instance, it can be deleted to avoid incurring additional charges.

Creating and launching instances in OpenStack is a fundamental skill for cloud administrators and users. By understanding how to create and launch instances based on the 'cirros' image, users can gain experience with the OpenStack platform and build their skills in cloud computing. Additionally, the 'cirros' image provides an easy and low-risk way to experiment with OpenStack and learn about its features and capabilities.

TABLE OF CONTENT

Chapter No.	Title	Page No.
1	Introduction	1
2	Problem Statement	2
	2.2 Objectives	3
3	Literature Survey	4
4	Technological Overview	6
	4.1 Libraries used	7
	4.2 Tools Used	8
	4.3 Dashboard Used	9
	4.4 Services Used	10
	4.5 Programming Language Used	16
5	Architecture & Methodology	17
	5.1 OpenStack Compute (Nova)	18
	5.2 OpenStack Image Service (Glance)	19
	5.3 OpenStack Networking (Neutron)	20
	5.4 Identity Service (Keystone)	21
	5.5 Horizon Dashboard	22
	5.6 Compute Node	23
	5.7 Image Service Node	24
6	Results & Discussion	26
	6.1 Single-Node OpenStack Deployment	26
	6.2 Installation	26

6.3	Initialization	27
6.4	Verification	27
6.5	Query OpenStack	27
6.6	Create an Instance	28
6.7	Connect to the Instance	29
6.8	Access the Cloud Dashboard	29
7	References	31

List of Images

Sl. No	Figure Number	Figure Description
1	5.1	Basic Architecture
2	5.2	OpenStack Compute
3	5.3	OpenStack Image Service
4	5.4	OpenStack Networking
5	5.5	Identity Service
6	5.6	Horizon Dashboard
7	5.7	Compute Node
8	5.8	Image Service Node
9	6.1	Created Instance

List of Tables

Sl. No	Table Number	Table Description
1	4.1	List of modules and technologies used

Chapter 1

INTRODUCTION

OpenStack is a popular open-source cloud computing platform that enables users to deploy and manage virtual machines and other resources in a flexible and scalable manner. One of the key features of OpenStack is its ability to create and launch instances, which are virtual machines that can be used to run applications, store data, and perform other tasks.

To create and launch an instance in OpenStack, the user needs to choose a base image to use as a template for the instance. An image is a pre-configured virtual disk that contains an operating system and any other software or files needed to run an application. OpenStack provides a wide range of images to choose from, including popular Linux distributions like Ubuntu, CentOS, and Debian, as well as specialized images for applications like WordPress, Drupal, and OpenVPN.

One of the most commonly used images in OpenStack is the 'cirros' image. The 'cirros' image is a small, lightweight image that is designed for testing and experimentation purposes. It boots quickly and requires minimal resources, making it ideal for use in low-performance environments or when running multiple instances simultaneously.

In this study, we will discuss how to create and launch an instance in OpenStack based on the 'cirros' image. We will provide step-by-step instructions for using the OpenStack dashboard or CLI to launch an instance, as well as tips for configuring and managing the instance once it is launched. By the end of this article, readers will have a better understanding of how to use OpenStack to create and manage instances and will be able to apply this knowledge to their own cloud computing projects and applications.

Chapter 2

PROBLEM STATEMENT

The problem statement addressed in the article is the need for users to create and launch instances in OpenStack for their cloud computing projects and applications. Specifically, the article focuses on creating and launching an instance based on the 'cirros' image, which is a popular lightweight image that is often used for testing and experimentation purposes. The article provides step-by-step instructions for users to follow, both in the OpenStack dashboard and CLI, in order to successfully create and launch the instance. The article also touches on configuring and managing the instance once it has been launched. By addressing this problem statement, the article aims to assist users in effectively utilizing OpenStack for their cloud computing needs.

2.2. OBJECTIVES

The objectives of this project are to:

- Provide a clear understanding of the process of creating and launching an instance in OpenStack based on the 'cirros' image.
- Walk users through the steps of creating and launching an instance using both the OpenStack dashboard and CLI.
- Highlight the importance of choosing the appropriate image and flavor for an instance based on the specific needs of the application or workload.
- Discuss how to configure and manage the instance once it has been launched, including tasks such as installing software, configuring settings, and monitoring performance.
- Address common issues that users may encounter during the instance creation and launch process and provide solutions to those issues.
- Empower users to effectively utilize OpenStack for their cloud computing needs, specifically for creating and launching instances based on the 'cirros' image.

Chapter 3

LITERATURE SURVEY

Neha et al. (2021) provides a comprehensive overview of the OpenStack cloud computing platform, with a particular focus on the Nova component, which is used for managing compute resources and launching instances. The authors discuss the various OpenStack components and their roles in the cloud computing stack. They also cover the benefits of using OpenStack, such as flexibility and scalability, as well as the challenges, such as complexity and interoperability issues. The paper concludes with a discussion of future trends and developments in the OpenStack ecosystem.

Devarajan et al. (2018) evaluates the performance of OpenStack Nova for scientific computing workloads and compares it to other cloud platforms. The authors discuss the challenges of achieving high performance and scalability with OpenStack, including issues related to networking, storage, and virtualization. They also present a set of benchmarks for evaluating OpenStack performance and provide recommendations for optimizing the system for scientific computing workloads.

Righi et al. (2019) provides a systematic review of OpenStack for enterprise use, including its features, benefits, and challenges. The authors discuss various deployment models, such as public, private, and hybrid clouds, as well as strategies for managing OpenStack in enterprise environments. They also cover topics such as security, scalability, and interoperability, and provide examples of real-world OpenStack deployments in the enterprise sector.

Alqahtani et al. (2021) provides a comprehensive survey of security issues and challenges in cloud computing, including those related to OpenStack. The authors discuss various security threats and vulnerabilities, such as data breaches, insider attacks, and denial-of-service (DoS) attacks, and suggest ways to mitigate those risks. They also cover topics such as compliance, privacy, and trust, and provide recommendations for improving cloud security.

Jackson et al. (2012) provides a practical guide to using OpenStack for cloud computing, including step-by-step instructions for creating and launching instances. It covers various advanced topics, such as networking, storage, and security, and includes real-world examples and best practices. The book is designed to be a hands-on resource for OpenStack administrators and developers, and includes numerous code examples and command-line instructions.

Chapter 4

TECHNOLOGICAL OVERVIEW

This whole work is based on OpenStack and Swift. For working on different modules, Python libraries and tools have been brought to use. The list of the libraries and tools used are as follows:

Table 4.1 List of Modules and technologies Used

S. no.	Name	Type	Purpose
1	Python	Programming Language	To simplify the installation process and provide a default configuration to use
2	OpenStack SDK	Library	To provide a unified API to interact with OpenStack services
3	Horizon	Dashboard	To allow users to manage their OpenStack resources
4	Neutron	Service	To provide APIs for managing networks, subnets, and routers
5	Cinder	Service	To provide APIs for managing volumes and their associated attachments
6	Glance	Service	To provide APIs for managing virtual machine images
7	Nova	Service	To provide APIs for launching and managing virtual machines
8	Keystone	Service	To provide authentication and authorization services to other OpenStack services
9	Heat	Service	To manage infrastructure deployments as code
10	Ansible	Tool	To automate OpenStack deployments and manage OpenStack infrastructure

4.1. LIBRARIES USED

4.1.1. *OpenStack SDK*

OpenStack SDK is a set of software development tools that allow developers to create, deploy, and manage applications on an OpenStack cloud. It is designed to simplify the process of building applications that interact with OpenStack services, such as Nova, Cinder, Neutron, and Swift.

The OpenStack SDK provides a set of libraries, command-line tools, and language-specific software development kits (SDKs) that allow developers to interact with OpenStack services using a variety of programming languages, including Python, Java, and Ruby.

The SDK provides a consistent interface for interacting with OpenStack services, regardless of the underlying implementation or version of OpenStack being used. This allows developers to write applications that are compatible with a wide range of OpenStack environments, without having to worry about the specific implementation details of each service.

The SDK also provides a set of tools for automating the deployment of OpenStack environments, including tools for provisioning, scaling, and managing OpenStack infrastructure. This allows developers to quickly and easily set up test environments for their applications, as well as production environments for deploying their applications in the cloud.

Overall, the OpenStack SDK is a powerful tool for developers looking to build applications on an OpenStack cloud. Its consistent interface, language support, and automation tools make it easy to create, deploy, and manage applications that interact with OpenStack services.

4.2. TOOLS USED

4.2.1. *Ansible*

Ansible is an open-source automation tool that provides a simple and powerful way to automate IT tasks. It is designed to be easy to use, with a simple syntax and a modular architecture that makes it easy to extend and customize.

At its core, Ansible uses SSH to connect to remote machines and run tasks, which can be defined in YAML files called playbooks. Playbooks are used to define a set of tasks that should be run on a set of hosts, and can be used to automate a wide range of IT tasks, including software deployment, configuration management, and system administration.

One of the key benefits of Ansible is its simplicity. Because it is based on YAML and SSH, it requires minimal setup and can be used with a wide range of platforms and technologies. It also provides a large library of pre-built modules, which can be used to perform a wide range of tasks, from installing software packages to configuring network settings.

Another key benefit of Ansible is its ability to scale to large and complex IT environments. Ansible supports a variety of deployment architectures, including masterless and master/agent architectures, and can be used to manage large clusters of servers or cloud-based environments.

Ansible also provides a number of advanced features, including support for templating and conditional statements, which allow users to customize their automation workflows based on specific conditions. It also supports roles, which allow users to modularize their playbooks and reuse code across multiple

projects.

Overall, Ansible is a powerful and flexible automation tool that can be used to automate a wide range of IT tasks. Its simplicity, scalability, and modularity make it a popular choice for organizations of all sizes, from small startups to large enterprises.

4.3. Dashboard Used

4.3.1. Horizon

Horizon is an open-source web-based dashboard that provides a graphical user interface (GUI) for managing and monitoring OpenStack services. It provides a single point of access for OpenStack administrators, allowing them to perform a wide range of tasks, from provisioning instances to managing storage and network resources.

Horizon is designed to be easy to use and intuitive, with a clean and modern interface that is easy to navigate. It provides a range of features and functionality, including support for multiple OpenStack projects and regions, customizable dashboards and workflows, and role-based access control.

One of the key benefits of Horizon is its ability to simplify and streamline complex tasks. It provides a range of wizards and workflows that guide users through common tasks, such as creating instances or configuring networks, and can help to reduce errors and ensure consistency across deployments.

Horizon also provides a number of advanced features, including support for plugins and extensions, which allow users to add new functionality to the dashboard. It also supports integration with other OpenStack services, such as Nova and Neutron, and can be used to monitor and manage resources

across multiple clouds and data centers.

Overall, Horizon is a powerful and flexible tool that can help to simplify and streamline the management of OpenStack deployments. Its intuitive interface, customizable workflows, and support for advanced features make it a popular choice for OpenStack administrators and users.

4.4. Services Used

4.4.1. Neutron

Neutron is an OpenStack component that provides networking services for cloud computing. It is responsible for managing the network infrastructure and connectivity between instances and other services in an OpenStack deployment.

Neutron supports a wide range of network configurations, including virtual LANs (VLANs), virtual private networks (VPNs), and software-defined networking (SDN) through OpenFlow. It also provides a range of network functions, such as load balancing, firewalling, and virtual private network (VPN) services.

One of the key benefits of Neutron is its ability to provide a flexible and scalable networking infrastructure for OpenStack deployments. It provides a range of APIs and interfaces that allow administrators to manage and configure network resources, and can be integrated with other OpenStack components, such as Nova and Horizon.

Neutron uses a plug-in architecture to support a range of networking technologies, including Open vSwitch, Linux bridge, and third-party networking solutions. This allows administrators to select the most appropriate networking technology for their needs and provides a high degree of flexibility and choice.

Overall, Neutron is a critical component of OpenStack and plays a key role in providing network connectivity and infrastructure for cloud computing. Its flexibility, scalability, and support for a wide range of networking technologies make it a popular choice for OpenStack administrators and users.

4.4.2. Cinder

Cinder is a core service in OpenStack that provides persistent block storage to virtual machines and applications running in the cloud. It is designed to allow users to request and consume storage resources on-demand and to support a wide range of storage solutions, including local disks, network-attached storage (NAS), and storage area networks (SAN).

Cinder provides a high degree of scalability and flexibility by allowing administrators to add and manage storage devices as needed. It also supports a range of storage features, including snapshotting, cloning, and replication.

Cinder uses a plug-in architecture to support a wide range of storage solutions, including EMC, NetApp, and IBM. This allows administrators to select the most appropriate storage technology for their needs and provides a high degree of flexibility and choice.

One of the key benefits of Cinder is its ability to integrate with other OpenStack services, such as Nova and Horizon. This integration allows administrators and users to easily provision, attach, and manage storage resources through a unified interface.

Overall, Cinder is a critical component of OpenStack and plays a key role in providing reliable and scalable block storage

for virtual machines and applications running in the cloud. Its flexibility, scalability, and support for a wide range of storage solutions make it a popular choice for OpenStack administrators and users.

4.4.3. Glance

Glance is the image service used in OpenStack, which provides a centralized repository for storing and managing virtual machine images, disk images, and other images used by OpenStack services. Glance enables users to create, share, and manage images, and provides a RESTful API that allows developers to interact with the image service programmatically.

Glance supports a variety of image formats, including RAW, VMDK, VDI, and ISO, among others. Glance also provides a metadata service that allows users to attach metadata to images, and to search for images based on metadata criteria.

Glance provides a simple and consistent interface for managing images, and allows users to easily create new images, modify existing images, and share images with other users. Glance also provides image caching capabilities, which can help to improve performance and reduce the load on image servers

4.4.4. Nova

Nova is the compute service in OpenStack, responsible for managing virtual machines and other compute resources. It provides a set of APIs and tools for users to create, manage, and terminate virtual machines, as well as the underlying compute resources such as hypervisors and virtual networks.

Nova is designed to be extensible, allowing users to plug in different hypervisors and virtualization technologies, including

KVM, Xen, VMware, and Hyper-V. It also supports multiple networking options, including flat, VLAN, and VXLAN networks.

At its core, Nova manages a pool of compute resources, including CPU, memory, and disk storage, and schedules virtual machines to run on these resources. Nova supports a variety of scheduling algorithms, including simple round-robin, weight-based, and filter-based scheduling, which allows users to customize how virtual machines are allocated to compute resources.

Nova provides a rich set of APIs and command-line tools for managing virtual machines and compute resources, including the ability to start, stop, pause, and resize virtual machines, as well as the ability to migrate virtual machines between compute hosts. Nova also supports snapshots, which allow users to create backups of their virtual machines for disaster recovery purposes.

In addition to its core compute capabilities, Nova also provides a number of related features, including support for security groups, which allow users to control network traffic to and from their virtual machines, and support for block storage volumes, which can be attached to virtual machines as additional disk storage.

Overall, Nova is a critical component of the OpenStack infrastructure, providing users with a powerful tool for managing virtual machines and compute resources in a flexible and extensible manner. Its rich set of features and APIs make it a popular choice for organizations of all sizes, from small startups to large enterprises.

4.4.5. Keystone

Keystone is the identity service used in OpenStack, which provides authentication and authorization services to other OpenStack services, including Swift. Keystone is responsible for verifying the identity of users and applications, and for managing access to OpenStack resources.

Keystone provides a single point of authentication and authorization for all OpenStack services, simplifying the management of security policies and access controls. Keystone also supports a variety of authentication mechanisms, including username/password authentication, token-based authentication, and federated authentication.

Keystone uses a role-based access control (RBAC) model to manage access to OpenStack resources. RBAC allows administrators to define roles and permissions for users and applications, and to assign these roles to specific resources or groups of resources. Keystone also supports domain-based RBAC, which allows administrators to define roles and permissions at the domain level.

Keystone provides a RESTful API that allows developers to interact with the identity service programmatically. The Keystone API allows developers to authenticate users and applications, to manage users, groups, and roles, and to manage access policies and permissions.

4.4.6. Heat

Heat is an orchestration service for OpenStack that provides APIs for managing infrastructure deployments as code. It allows users to describe their infrastructure in a declarative format using templates, which can then be used to automate the deployment

and management of resources in OpenStack. Heat can be used to manage the entire lifecycle of infrastructure resources, from creation to deletion, and can automate complex multi-tier application deployments.

Heat templates are written in YAML or JSON, and define the resources and their properties that make up a stack. A stack is a collection of resources that are managed as a single unit, and can include a wide variety of resources, such as virtual machines, networks, storage volumes, load balancers, and more. Heat templates can also include logic and conditional statements to define dependencies between resources, and can be parameterized to allow for reuse across different deployments.

Heat uses a component called the Heat Engine to orchestrate the deployment of resources. The Heat Engine receives requests from the user or a client, such as the OpenStack CLI, to create or modify a stack. It then communicates with other OpenStack services, such as Nova and Neutron, to create and configure the resources defined in the template.

Heat also includes a web-based dashboard, called the Heat Dashboard or Horizon, that allows users to create and manage Heat templates and stacks through a graphical user interface. The dashboard provides a user-friendly way to create and modify templates, and allows users to view the status of their stacks and resources.

Overall, Heat provides a powerful tool for automating the deployment and management of infrastructure resources in OpenStack. Its ability to define complex infrastructure

deployments as code, with built-in support for dependencies and reuse, makes it a valuable tool for managing large-scale OpenStack deployments.

4.5. Programming Language Used

4.5.1. Python

Python is a high-level programming language that is widely used in a variety of applications, from web development and scientific computing to machine learning and data analysis. Python was first released in 1991 and has since become one of the most popular programming languages in the world.

Python is known for its simplicity, readability, and ease of use, making it a popular choice for both beginners and experienced programmers. Python's syntax is designed to be clear and concise, making it easy to write and understand code. Python also includes a large standard library, which provides a wide range of built-in functions and modules that can be used to perform a variety of tasks.

Python is an interpreted language, which means that code is executed directly by the Python interpreter, rather than being compiled into machine code. This makes it easy to test and debug code, as changes can be made and tested immediately without the need for compilation.

Python is also an object-oriented language, which means that code is organized into objects and classes that can be used to represent real-world entities and concepts. This makes it easy to write modular and reusable code, which can save time and effort when building complex applications.

Chapter 5

METHODOLOGY & ARCHITECTURE

The methodology and architecture of the above topic, which is the to create and launch an instance based on the 'cirros' image, can be broken down into several components.

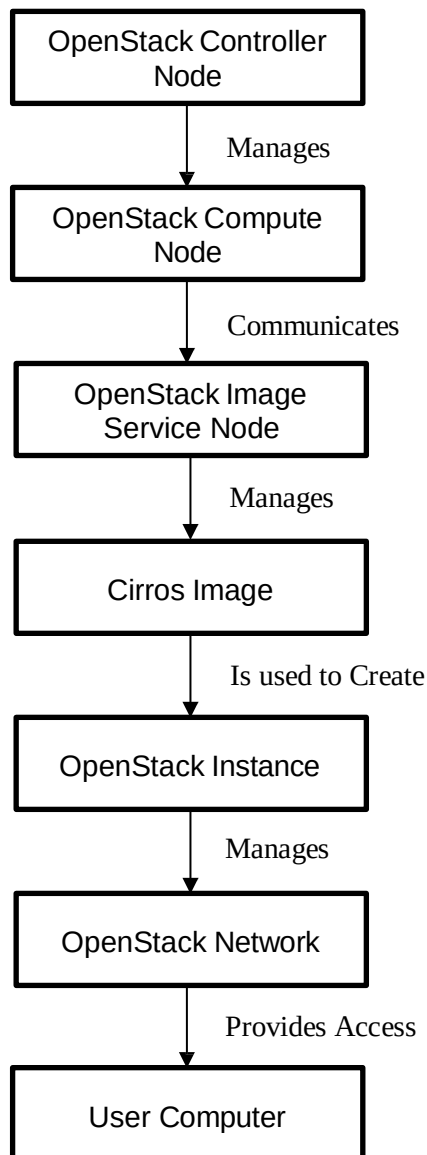


Fig. 5.1 Basic Architecture

Creating and launching an instance in OpenStack based on a specific image is a fundamental task for cloud administrators and users. In this context, the 'cirros' image is a popular choice for testing and experimentation purposes, as it is a small, lightweight image that boots

quickly and requires minimal resources. To create and launch an instance based on the 'cirros' image, one needs to follow some straightforward steps, including finding the image ID or name, selecting a hardware configuration (flavor), creating the instance, and logging in to it if necessary. This process can be done using the OpenStack dashboard or the OpenStack CLI, and it is an essential skill for anyone who wants to work with OpenStack.

The Fig 5.1 shows the architecture of OpenStack for creating and launching an instance based on the 'cirros' image. The OpenStack controller node manages the OpenStack compute node, which communicates with the OpenStack image service node. The image service node manages the 'cirros' image, which is used to create an OpenStack instance. The instance is managed by the OpenStack network, which provides access to the user's computer.

5.1 OPENSTACK COMPUTE (NOVA)

OpenStack Compute, also known as Nova, is responsible for managing the compute resources in OpenStack. This module provides a scalable and flexible system for launching and managing instances. Nova allows users to create and manage virtual machines by providing access to computing resources such as CPU, memory, and disk.

As shown in the flowchart, when a user creates an instance, the request is sent to Nova. Nova then schedules the instance to run on a compute node, which launches the instance on the hypervisor. Finally, the hypervisor runs the instance.

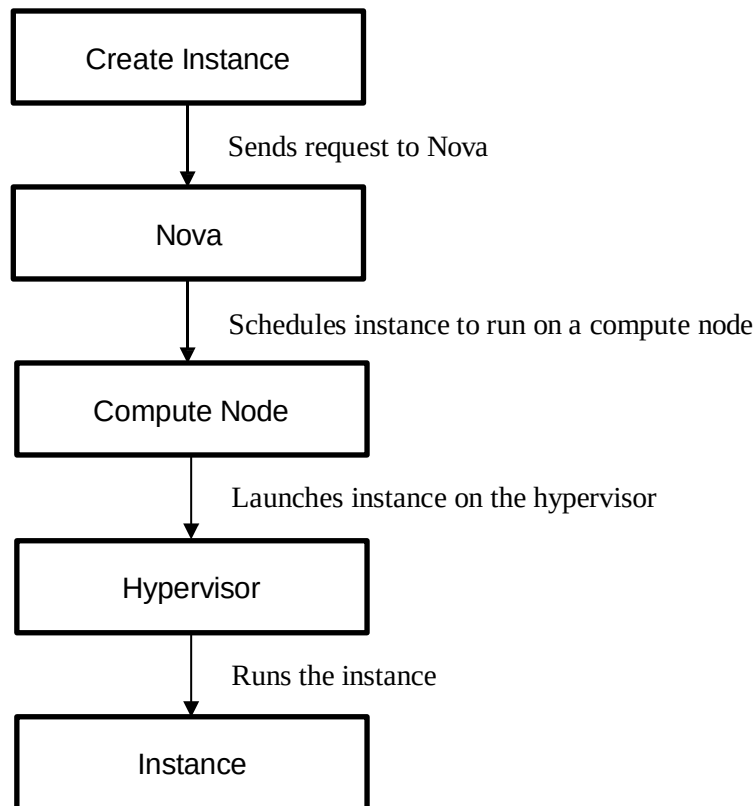


Fig. 5.2 *OpenStack Compute*

5.2 OPENSTACK IMAGE SERVICE (GLANCE)

OpenStack Image Service, also known as Glance, is responsible for managing the images used to create instances. This module provides a repository for storing and cataloging virtual machine images. Images can be created and uploaded to Glance, and then used to launch instances.

As shown in the flowchart, when a user creates an image, it is uploaded to Glance. Glance stores the image in its repository and makes it available for use in creating instances. Nova can then use the image to launch instances.

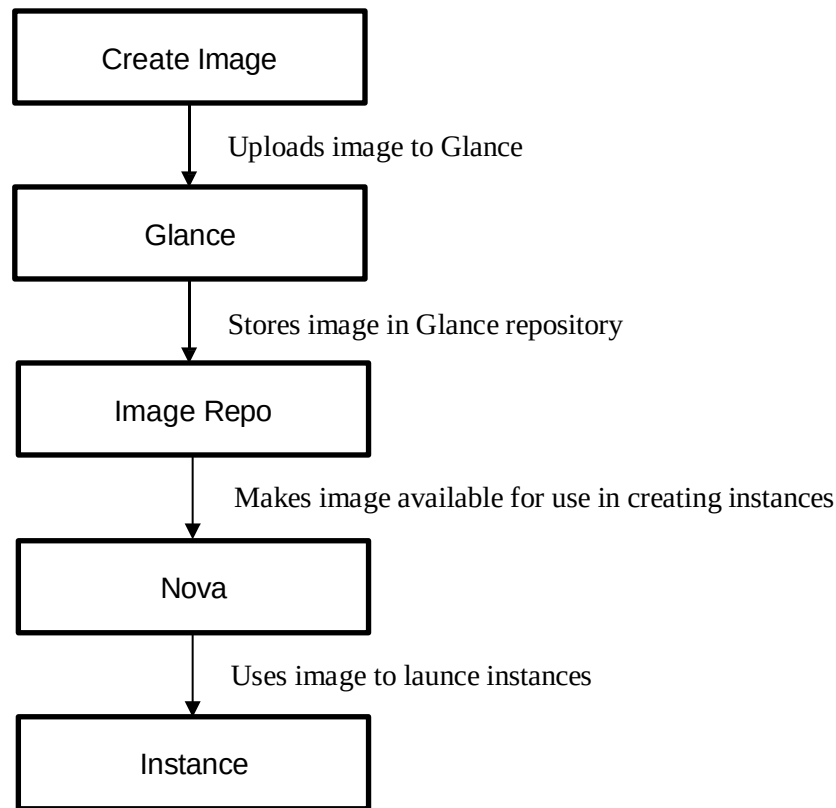


Fig. 5.3 OpenStack Image Service

5.3 OPENSTACK NETWORKING (NEUTRON)

OpenStack Networking, also known as Neutron, is responsible for managing the network resources in OpenStack. This module provides a flexible and scalable system for creating and managing virtual networks, subnets, and routers. Neutron provides connectivity between instances and to external networks.

As shown in the flowchart, when a user creates a network, the request is sent to Neutron. Neutron creates the network, subnet, and router resources. These resources provide connectivity to instances. Once the resources are created, instances can be launched and connected to the network.

Neutron is a crucial component in OpenStack because it provides networking services to instances. Without Neutron, instances would not be able to communicate with each other or with external networks.

Therefore, it's important to understand how Neutron works and how to configure it properly to ensure a stable and reliable OpenStack deployment.

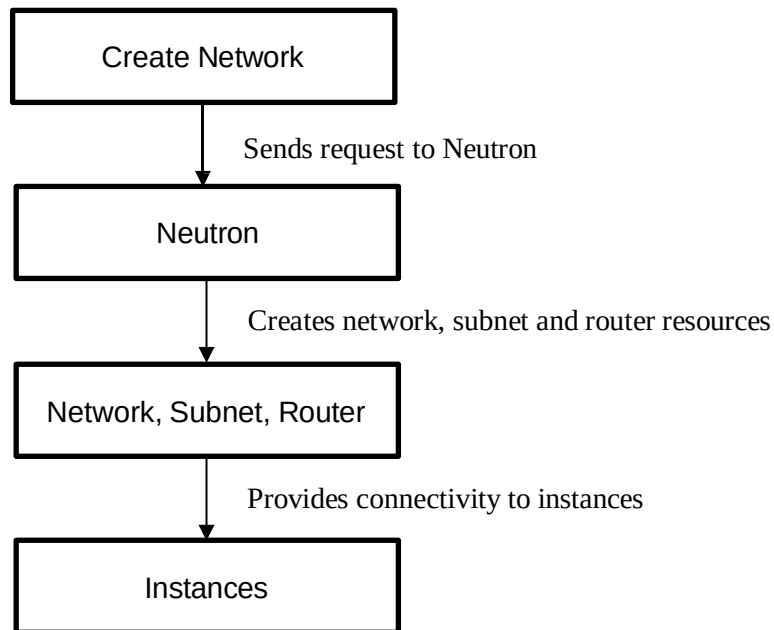


Fig. 5.4 OpenStack Networking

5.4 IDENTITY SERVICE (KEYSTONE)

OpenStack Identity, also known as Keystone, is responsible for managing the identity and access control in OpenStack. This module provides a centralized authentication and authorization system for all OpenStack services. Keystone provides a single point of entry for users and administrators to access OpenStack resources.

As shown in the flowchart, when a user authenticates to access OpenStack resources, the request is sent to Keystone. Keystone validates the user's credentials and authorizes access to OpenStack resources.

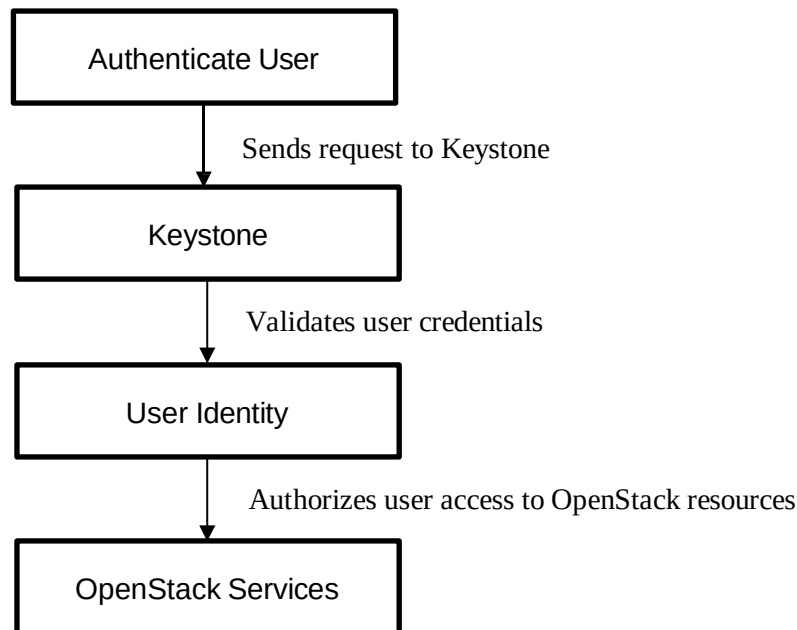


Fig. 5.5 Identity Service

5.5 HORIZON DASHBOARD

OpenStack Dashboard, also known as Horizon, is the web-based user interface for OpenStack. This module provides a graphical interface for users and administrators to access and manage OpenStack resources. Horizon is designed to be easy to use and allows users to perform common tasks such as creating instances, managing networks, and viewing resource usage.

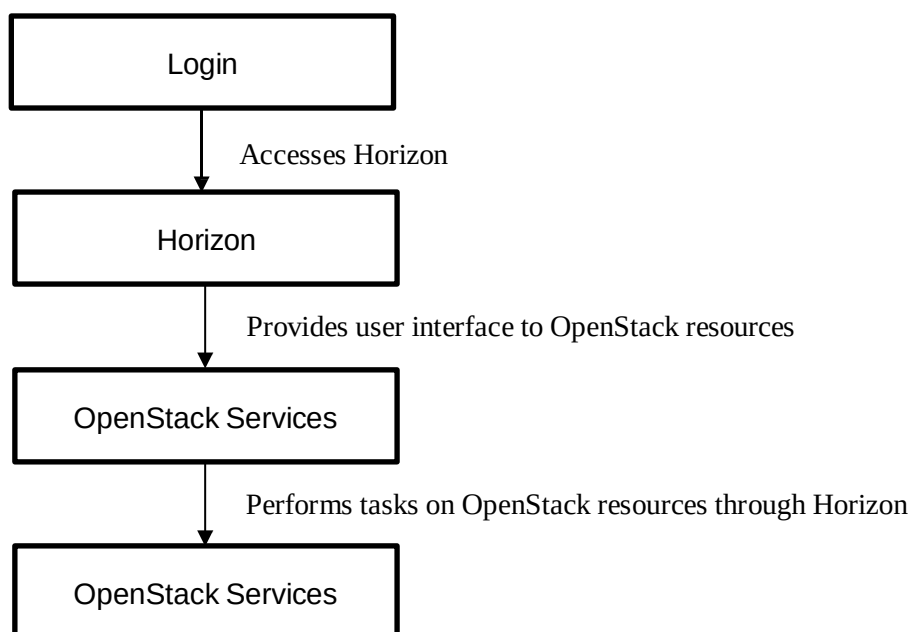


Fig. 5.6 Horizon Dashboard

As shown in the flowchart, when a user logs in to Horizon, they can access and manage OpenStack resources through the web-based interface. Users can perform tasks on OpenStack resources through Horizon, which communicates with OpenStack services to execute the tasks.

5.6 COMPUTE NODE

Compute Node is responsible for managing block storage in OpenStack. This module provides persistent storage for instances and allows users to create, attach, and detach volumes. Cinder also provides a Nova API for managing block storage.

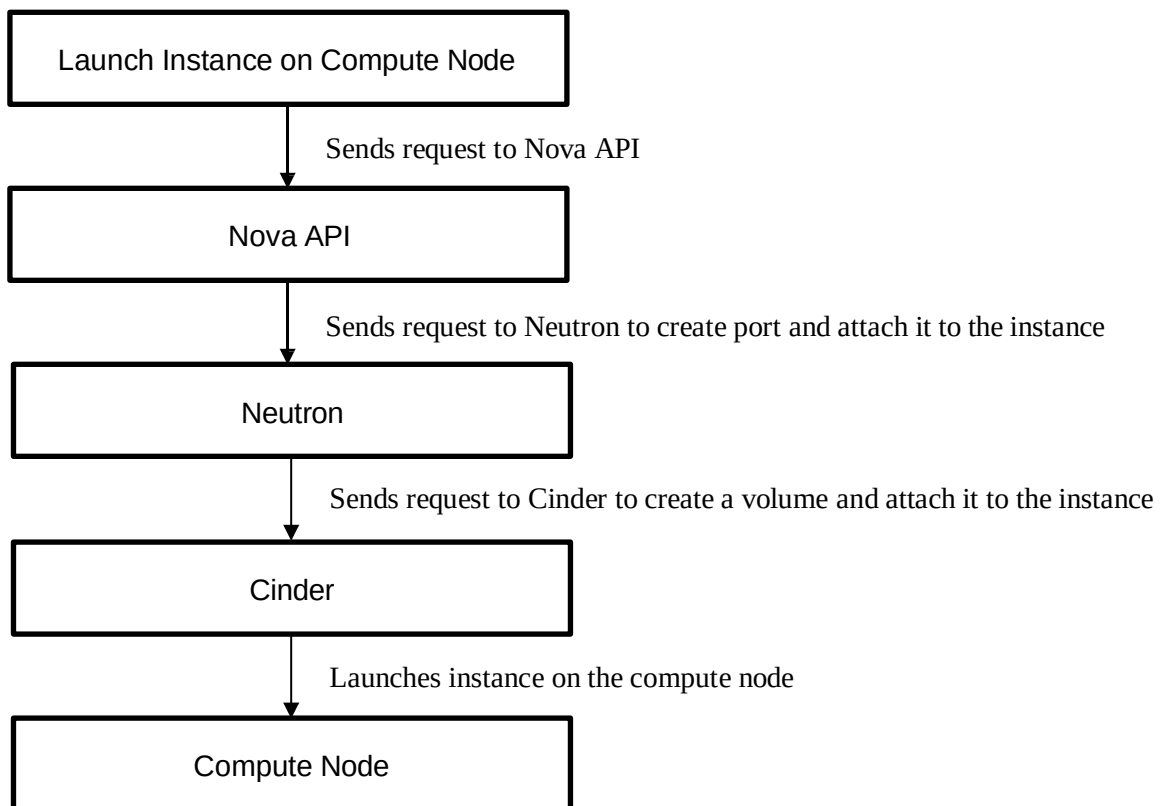


Fig. 5.7 Compute Node

As shown in the flowchart, when a user requests to launch an instance, the request is sent to the Nova API. The Nova API then sends a request to Neutron to create a port and attach it to the instance, ensuring that the instance has network connectivity. The Nova API also sends a

request to Cinder to create a volume and attach it to the instance, providing persistent storage.

Finally, the instance is launched on the compute node, which runs the virtualization software necessary to create and manage the instance. Once the instance is running, it can communicate with other instances on the network and access external resources through the Neutron network.

5.7 IMAGE SERVICE NODE

OpenStack Image Service, also known as Glance, is responsible for managing virtual machine images in OpenStack. This module allows users to create, register, and retrieve virtual machine images to be used as templates for instances. Glance also provides a RESTful API for managing images.

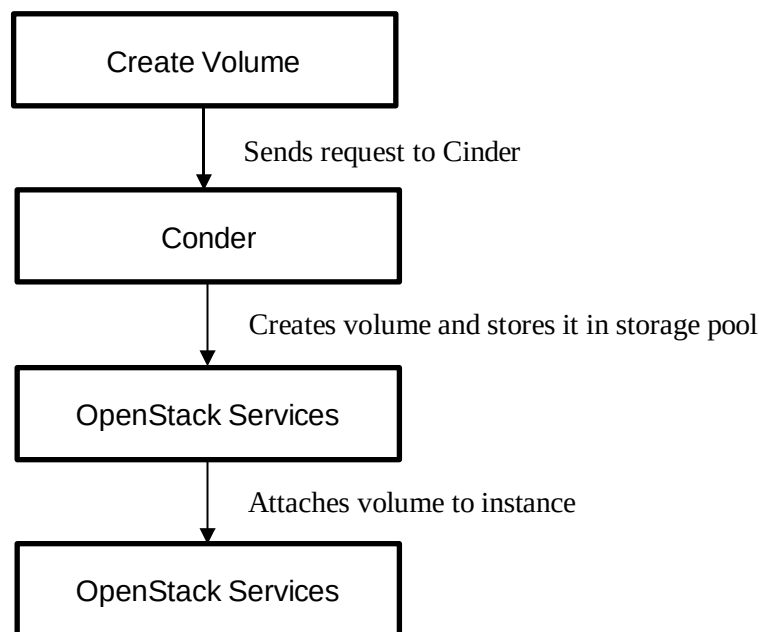


Fig. 5.8 Image Service Node

As shown in the flowchart, when a user creates an image, the request is sent to Glance. Glance stores the image in the image repository, which can be used to create instances.

Overall, these modules work together to provide a comprehensive cloud infrastructure for users and administrators to manage and utilize OpenStack resources.

Chapter 6

RESULTS & DISCUSSION

The results and discussions for the proposed project have been given as follows:

6.1 SINGLE-NODE OPENSTACK DEPLOYMENT

These instructions use MicroStack - OpenStack in a snap, MicroStack is a pure upstream OpenStack distribution designed for small scale and edge deployments, that can be installed and maintained with a minimal effort.

6.2 INSTALLATION

The installation step consists solely of installing the MicroStack snap.

At this time use the beta channel:

```
sudo snap install microstack --beta
```

Information on the installed snap can be viewed like this:

```
snap list microstack
```

Name	Version	Rev	Tracking	Publisher	Notes
microstack	ussuri	245	latest/beta	canonical✓	-

Here we see that OpenStack Ussuri has been deployed!

6.3 INITIALISATION

The initialisation step automatically deploys, configures, and starts OpenStack services. In particular, it will create the database, networks, an image, several flavors, and ICMP/SSH security groups. This can all be done within 10 to 20 minutes depending on your machine:

```
sudo microstack init --auto --control
```

6.4 VERIFICATION

The purpose of the verification step is to confirm that the cloud is in working order and to discover some of the defaults used by MicroStack. Verification will consist of the following actions:

- perform various OpenStack queries
- create an instance
- connect to the instance over SSH
- access the cloud dashboard

6.5 QUERY OPENSTACK

The standard openstack client comes pre-installed and is invoked like so:

```
microstack.openstack <command>
```

To list the default image:

```
microstack.openstack image list
```

```
+-----+-----+-----+
| ID                               | Name   | Status |
+-----+-----+-----+
| 7fefc80f-d745-4764-9389-00cc4a12585d | cirros | active |
+-----+-----+-----+
```

To get the default list of flavors:

```
microstack.openstack flavor list
```

```
+---+-----+-----+-----+-----+-----+-----+
| ID | Name       | RAM | Disk | Ephemeral | VCPUs | Is Public |
+---+-----+-----+-----+-----+-----+-----+
| 1  | m1.tiny    | 512 | 1    | 0         | 1     | True      |
| 2  | m1.small   | 2048 | 20   | 0         | 1     | True      |
| 3  | m1.medium  | 4096 | 20   | 0         | 2     | True      |
| 4  | m1.large   | 8192 | 20   | 0         | 4     | True      |
| 5  | m1.xlarge  | 16384 | 20   | 0         | 8     | True      |
+---+-----+-----+-----+-----+-----+-----+
```

6.6 CREATE AN INSTANCE

MicroStack comes with a convenient instance creation command called `microstack launch`. It uses the following defaults for its instances:

- keypair 'microstack'
- flavor 'm1.tiny'
- floating IP address on subnet '10.20.20.0/24'

To create an instance named 'test' based on the 'cirros' image:

```
microstack launch cirros -n test
```

The `microstack launch` command also supports arguments `--key`, `--flavor`,

--image, and --net-id, in which case you will need to create objects using the standard client if non-default values are desired.

6.7 CONNECT TO THE INSTANCE

Output from the microstack launch command includes all the information needed to connect to the instance over SSH:

```
Creating local "microstack" ssh key at /home/ubuntu/snap/microstack/common/.ssh/id_microstack
Launching server ...
Allocating floating ip ...
Server test launched! (status is BUILD)

Access it with `ssh -i /home/ubuntu/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.199`
```

Access the instance using the private SSH key associated with the default keypair:

```
ssh -i /home/ubuntu/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.199
```

6.8 ACCESS THE CLOUD DASHBOARD

You can log in to the web UI by pointing your browser to the following URL:

<https://10.20.20.1>

The username is 'admin' and the password is obtained in this way:

```
sudo snap get microstack config.credentials.keystone-password
```

Sample password:

```
0AEHxLgCBz7Wz4usvolAAt61TrDUz6zz
```

Upon logging in you should see the created instance:

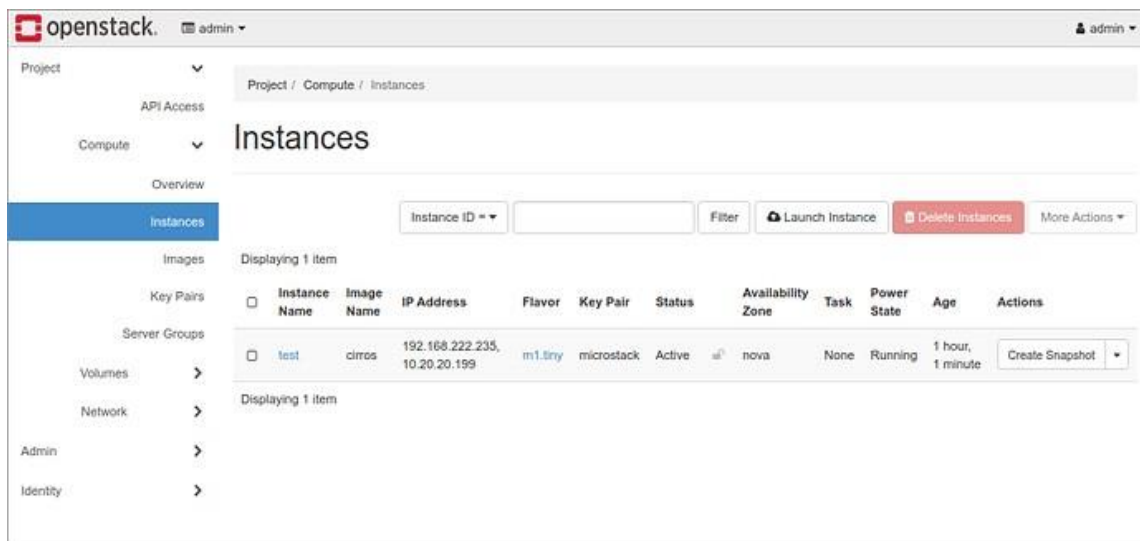


Fig 6.1: Created Instance

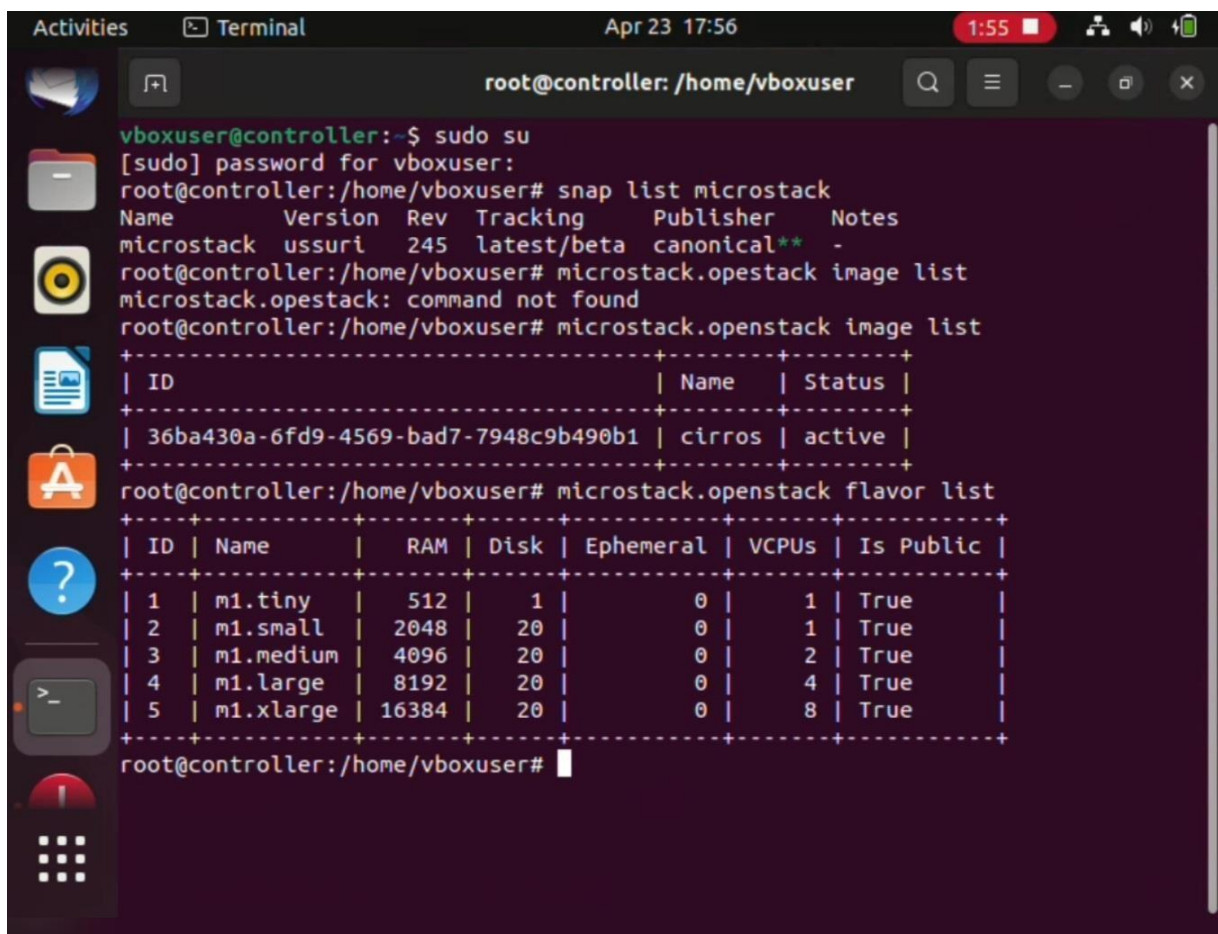


Fig 6.2: Vboxuser Login

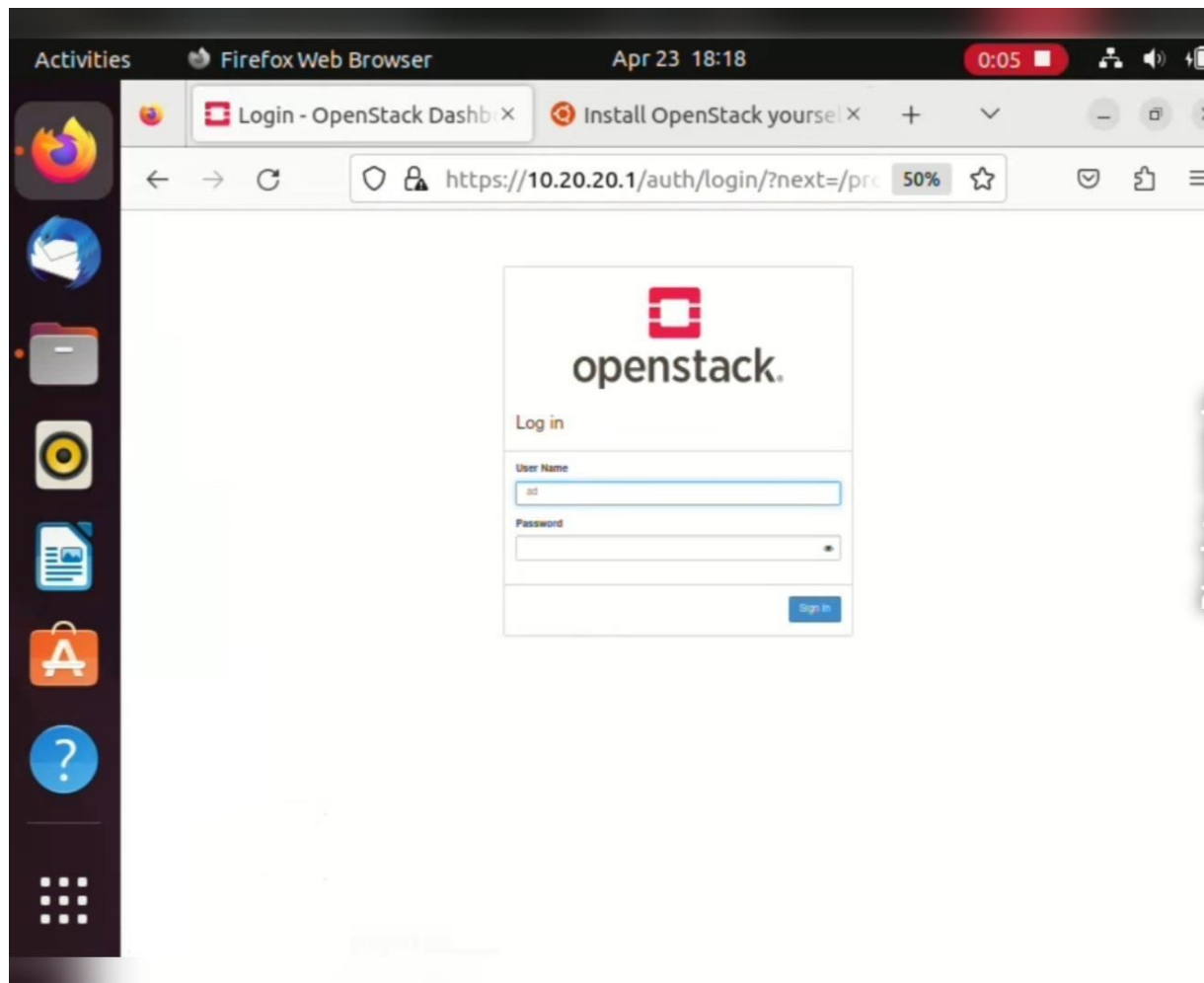


Fig 6.3: OpenStack Login User Name

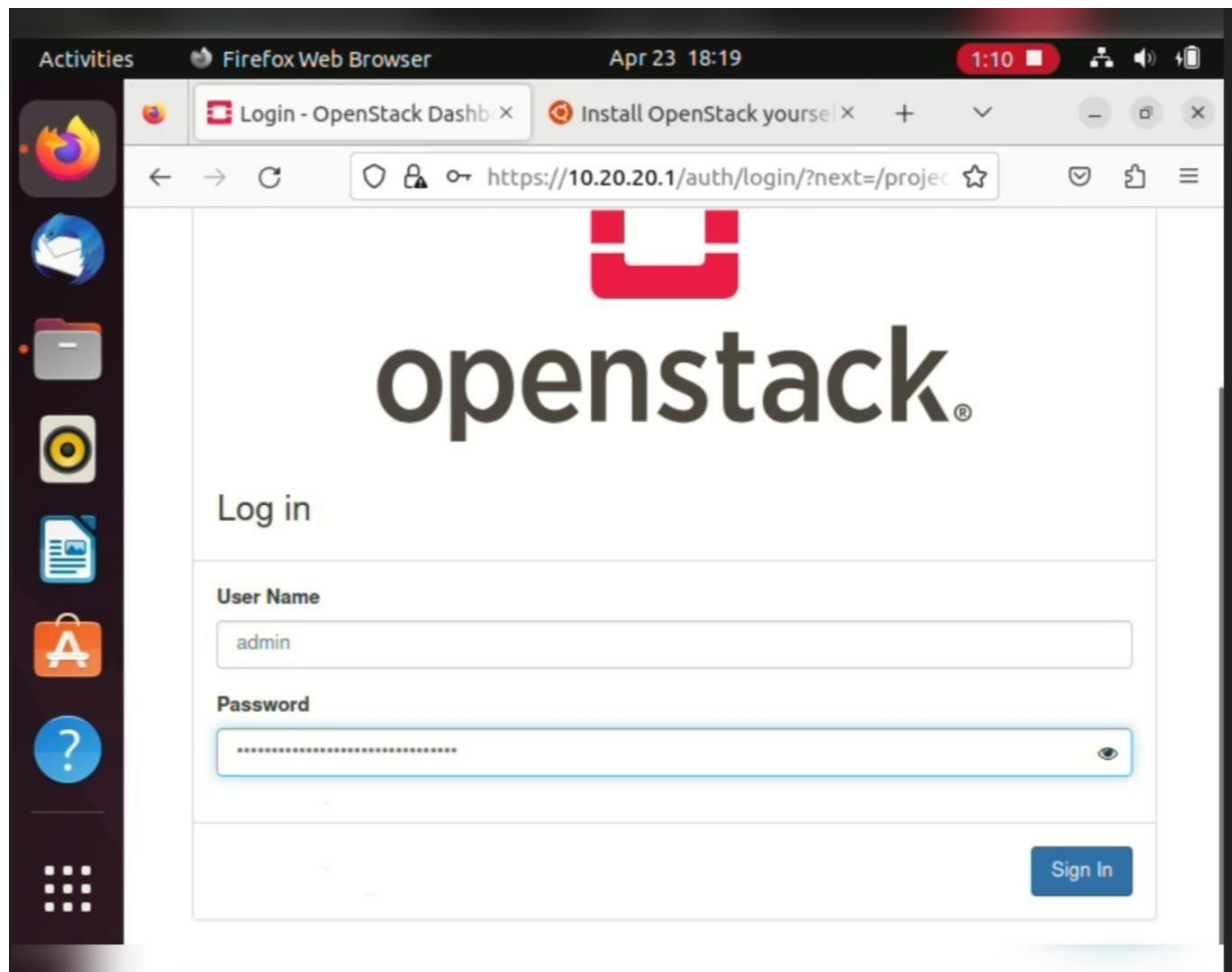
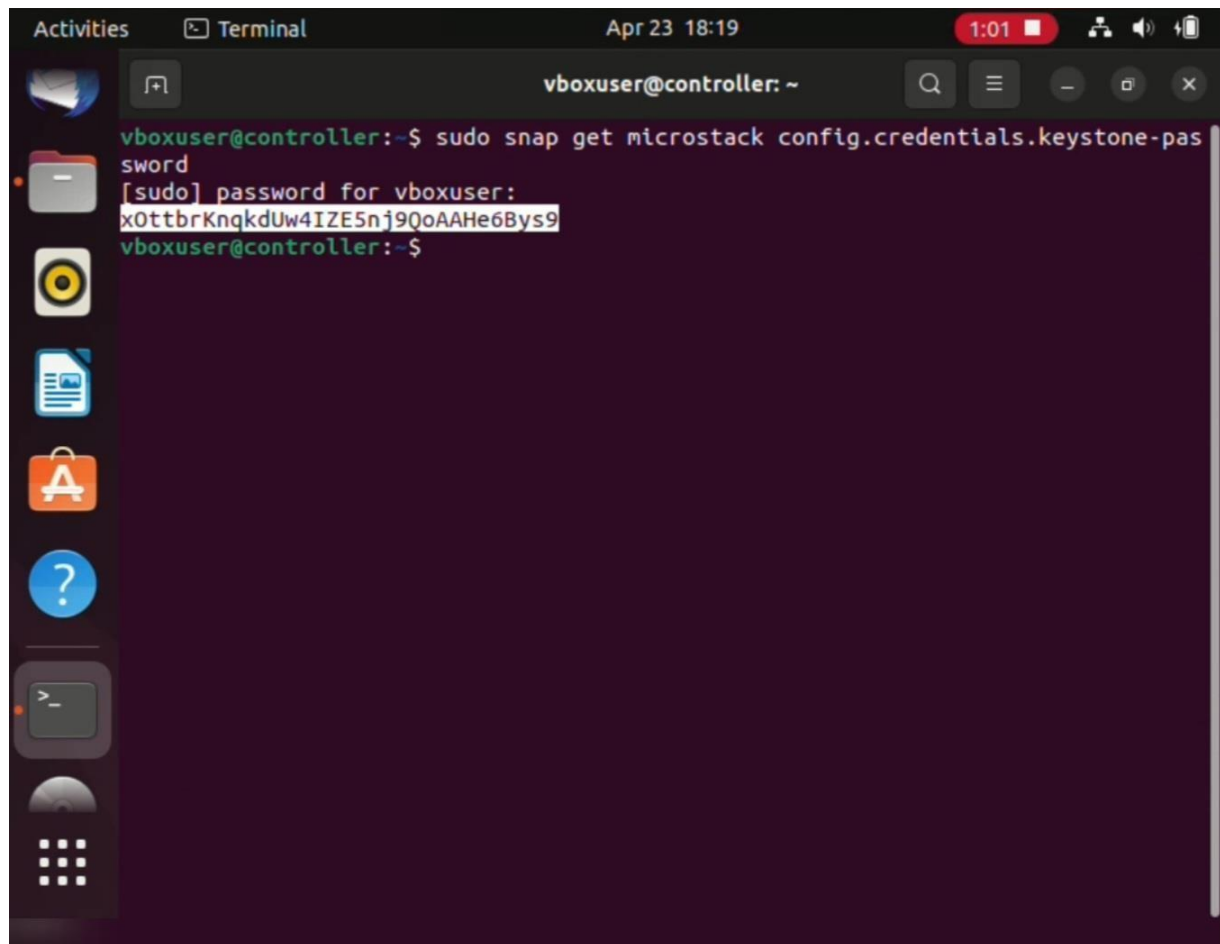


Fig 6.4: OpenStack Login Password



```
vboxuser@controller:~$ sudo snap get microstack config.credentials.keystone-password
[sudo] password for vboxuser:
x0ttbrKnqkdUw4IZE5nj9QoAAHe6Bys9
vboxuser@controller:~$
```

Fig 6.5: *MicroStack Installation*

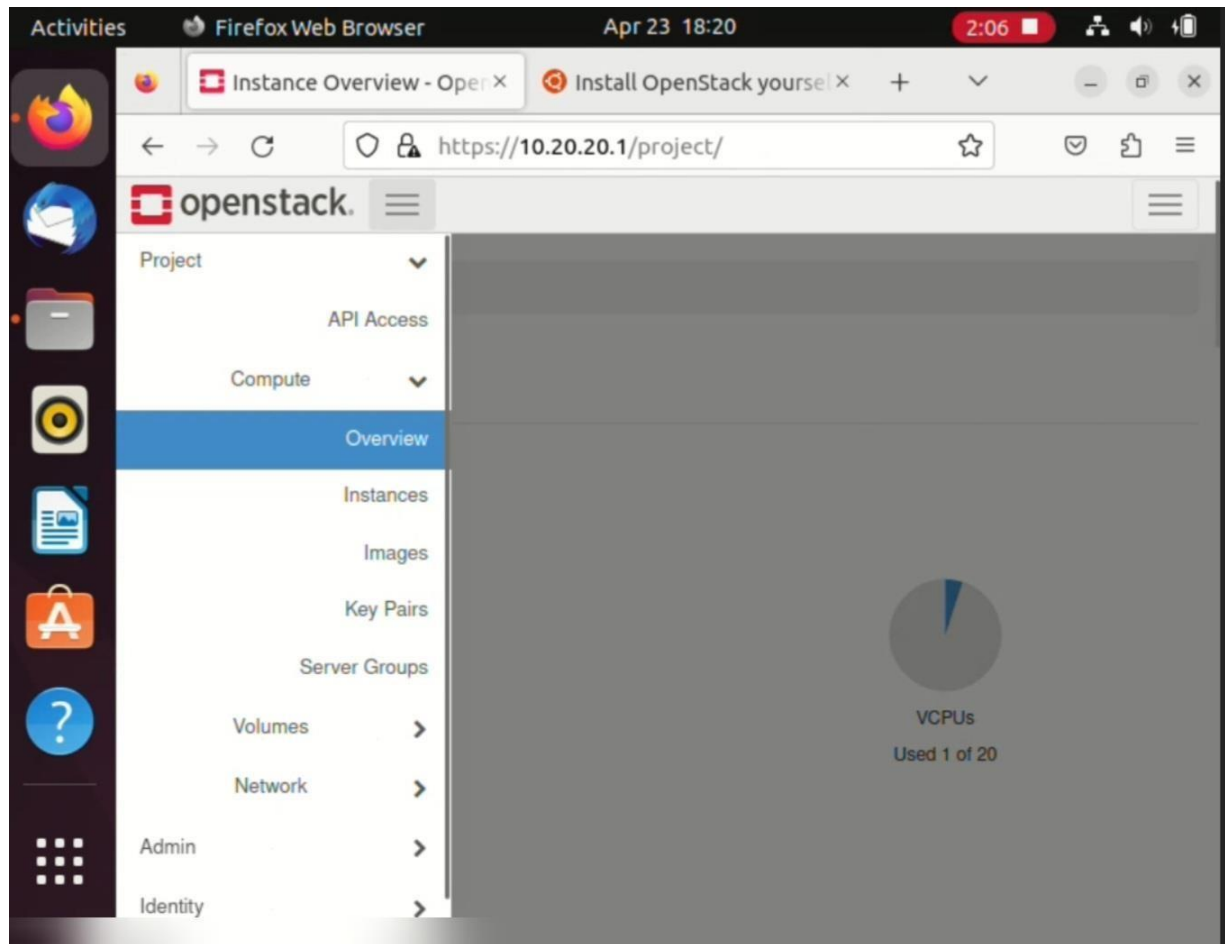
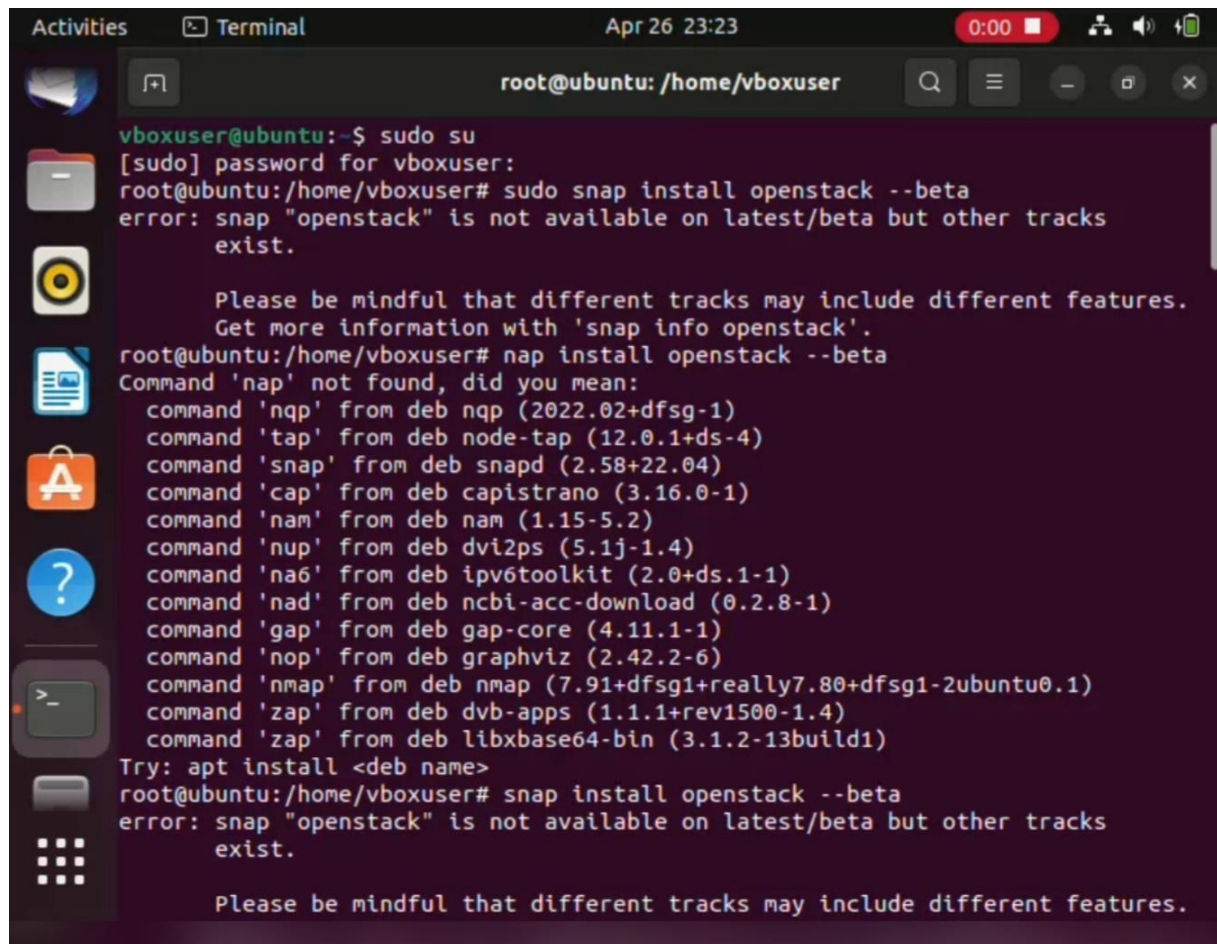


Fig 6.6: Installation Overview

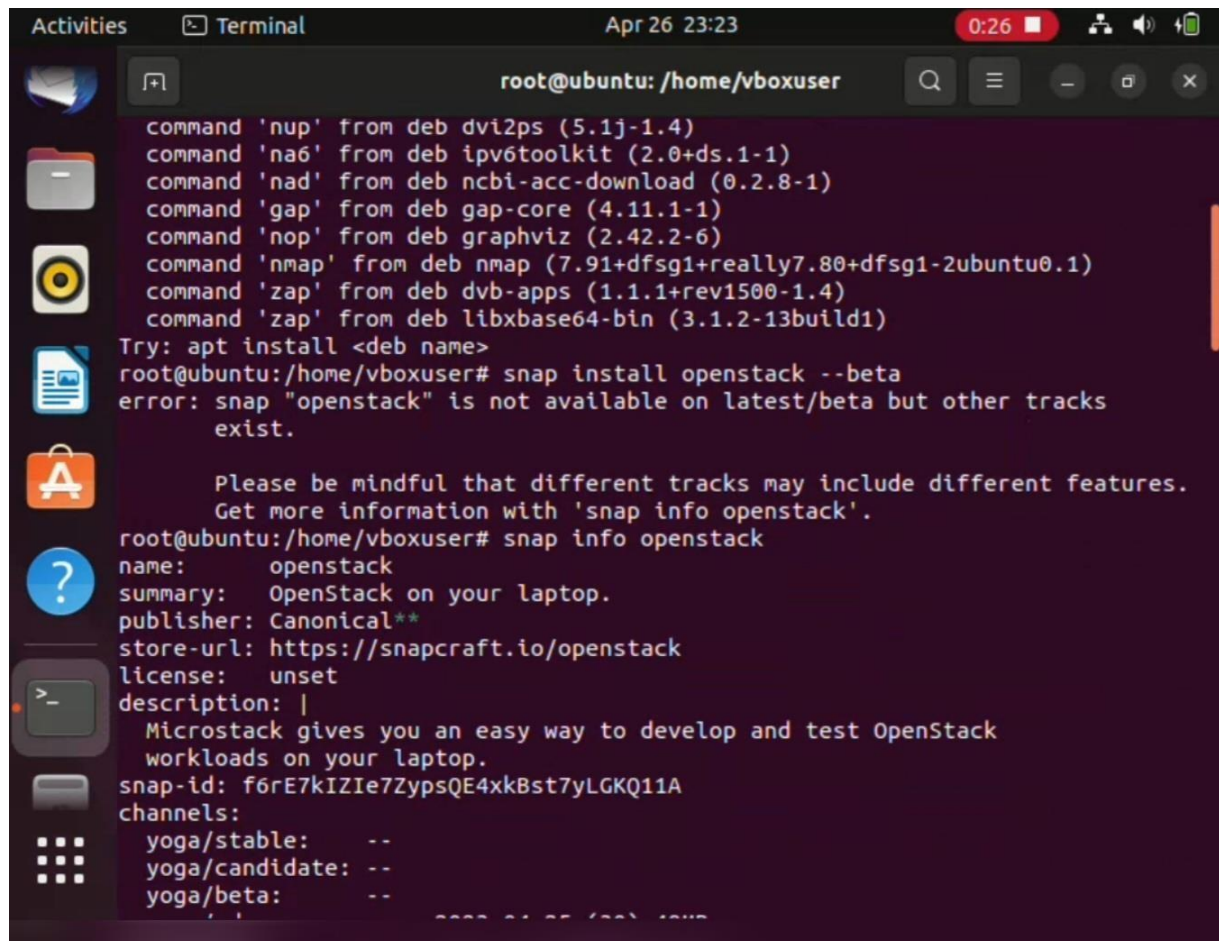
A terminal window titled 'Terminal' with a timestamp of 'Apr 26 23:23'. The window shows a user 'vboxuser' at 'ubuntu' in the directory '/home/vboxuser'. The user runs 'sudo su' and then 'sudo snap install openstack --beta'. An error message appears: 'error: snap "openstack" is not available on latest/beta but other tracks exist.' The user then runs 'nap install openstack --beta', which results in a 'Command not found' error. A list of commands found in the system is displayed, including 'nqp', 'tap', 'snap', 'cap', 'nam', 'nup', 'na6', 'nad', 'gap', 'nop', 'nmap', and 'zap'. The user then runs 'snap install openstack --beta' again, which results in the same error message. The terminal window has a sidebar with various application icons and a top bar with system status icons.

```
vboxuser@ubuntu:~$ sudo su
[sudo] password for vboxuser:
root@ubuntu:/home/vboxuser# sudo snap install openstack --beta
error: snap "openstack" is not available on latest/beta but other tracks
exist.

Please be mindful that different tracks may include different features.
Get more information with 'snap info openstack'.
root@ubuntu:/home/vboxuser# nap install openstack --beta
Command 'nap' not found, did you mean:
  command 'nqp' from deb nqp (2022.02+dfsg-1)
  command 'tap' from deb node-tap (12.0.1+ds-4)
  command 'snap' from deb snapd (2.58+22.04)
  command 'cap' from deb capistrano (3.16.0-1)
  command 'nam' from deb nam (1.15-5.2)
  command 'nup' from deb dvi2ps (5.1j-1.4)
  command 'na6' from deb ipv6toolkit (2.0+ds.1-1)
  command 'nad' from deb ncbi-acc-download (0.2.8-1)
  command 'gap' from deb gap-core (4.11.1-1)
  command 'nop' from deb graphviz (2.42.2-6)
  command 'nmap' from deb nmap (7.91+dfsg1+really7.80+dfsg1-2ubuntu0.1)
  command 'zap' from deb dvb-apps (1.1.1+rev1500-1.4)
  command 'zap' from deb libxbase64-bin (3.1.2-13build1)
Try: apt install <deb name>
root@ubuntu:/home/vboxuser# snap install openstack --beta
error: snap "openstack" is not available on latest/beta but other tracks
exist.

Please be mindful that different tracks may include different features.
```

Fig 6.7: OpenStack Installation

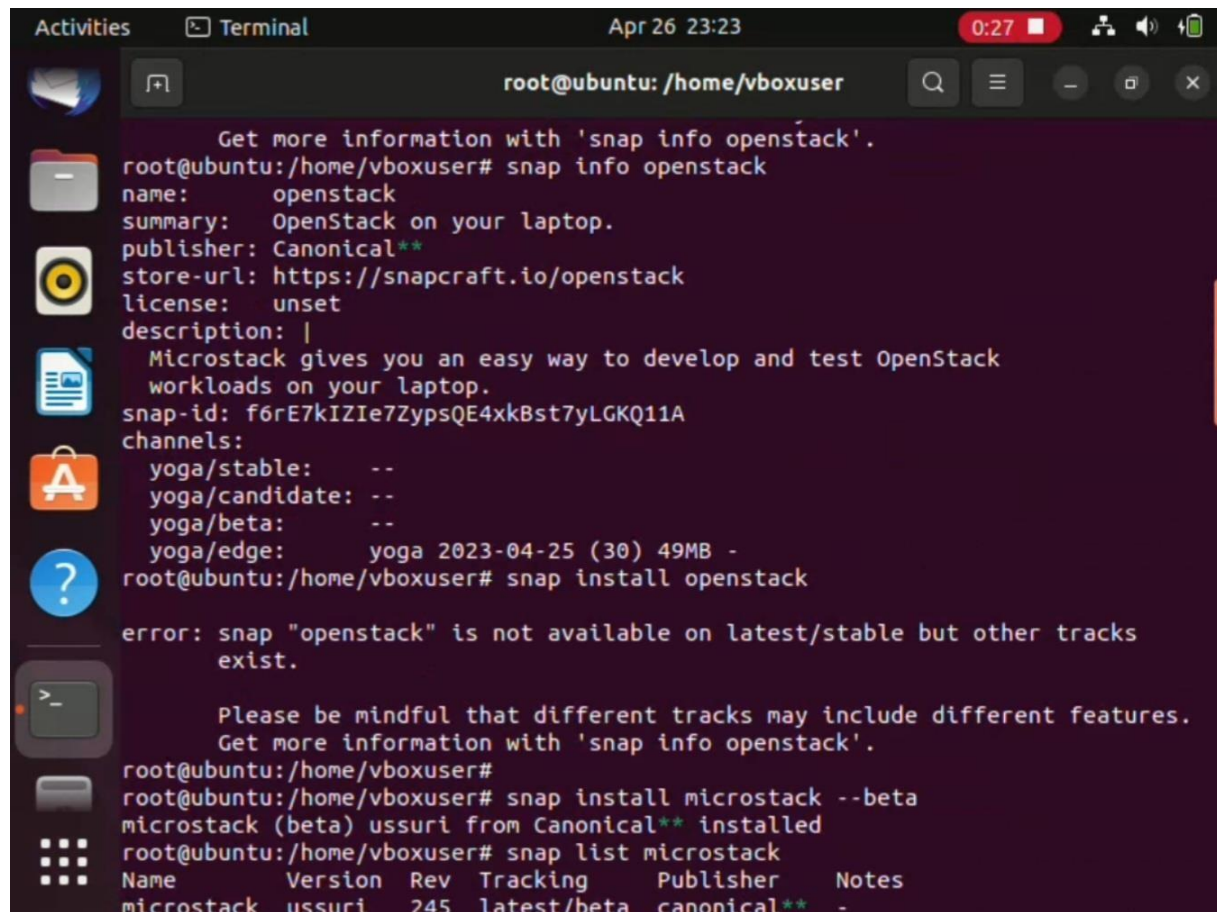


The image shows a terminal window titled 'Terminal' with the date 'Apr 26 23:23' and a system clock '0:26'. The prompt is 'root@ubuntu: /home/vboxuser'. The terminal displays the following commands and their outputs:

```
command 'nup' from deb dvi2ps (5.1j-1.4)
command 'na6' from deb ipv6toolkit (2.0+ds.1-1)
command 'nad' from deb ncbi-acc-download (0.2.8-1)
command 'gap' from deb gap-core (4.11.1-1)
command 'nop' from deb graphviz (2.42.2-6)
command 'nmap' from deb nmap (7.91+dfsg1+really7.80+dfsg1-2ubuntu0.1)
command 'zap' from deb dvb-apps (1.1.1+rev1500-1.4)
command 'zap' from deb libxbase64-bin (3.1.2-13build1)
Try: apt install <deb name>
root@ubuntu:/home/vboxuser# snap install openstack --beta
error: snap "openstack" is not available on latest/beta but other tracks exist.

Please be mindful that different tracks may include different features.
Get more information with 'snap info openstack'.
root@ubuntu:/home/vboxuser# snap info openstack
name:      openstack
summary:   OpenStack on your laptop.
publisher: Canonical**
store-url: https://snapcraft.io/openstack
license:   unset
description: |
  Microstack gives you an easy way to develop and test OpenStack
  workloads on your laptop.
snap-id: f6rE7kIZIe7ZypsQE4xkBst7yLGKQ11A
channels:
  yoga/stable: --
  yoga/candidate: --
  yoga/beta: --
```

Fig 6.8: OpenStack Installing



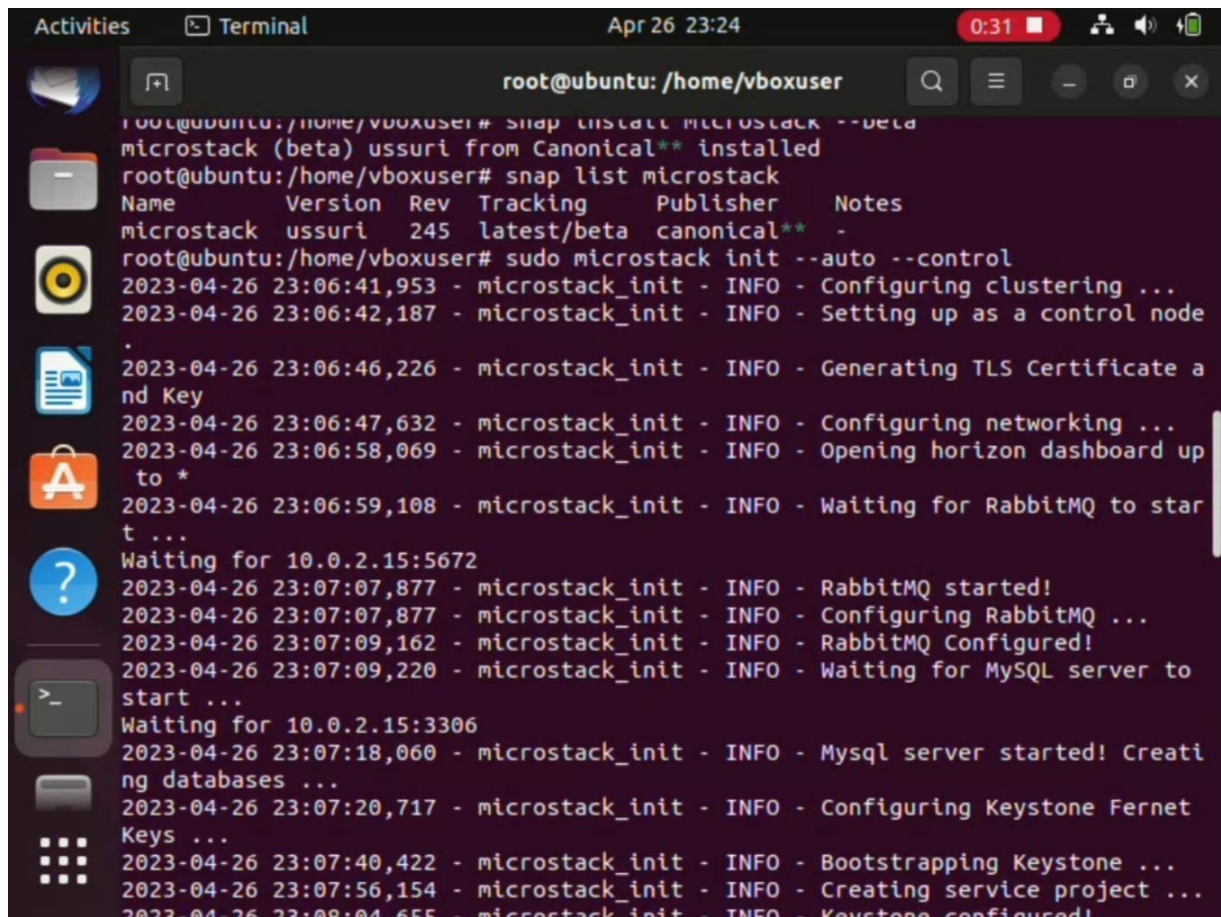
```
Activities Terminal Apr 26 23:23 0:27
root@ubuntu: /home/vboxuser

Get more information with 'snap info openstack'.
root@ubuntu:/home/vboxuser# snap info openstack
name:      openstack
summary:   OpenStack on your laptop.
publisher: Canonical**
store-url: https://snapcraft.io/openstack
license:   unset
description: |
  Microstack gives you an easy way to develop and test OpenStack
  workloads on your laptop.
snap-id: f6rE7kIZIe7ZypsQE4xkBst7yLGKQ11A
channels:
  yoga/stable:  --
  yoga/candidate: --
  yoga/beta:    --
  yoga/edge:    yoga 2023-04-25 (30) 49MB -
root@ubuntu:/home/vboxuser# snap install openstack

error: snap "openstack" is not available on latest/stable but other tracks
exist.

Please be mindful that different tracks may include different features.
Get more information with 'snap info openstack'.
root@ubuntu:/home/vboxuser#
root@ubuntu:/home/vboxuser# snap install microstack --beta
microstack (beta) ussuri from Canonical** installed
root@ubuntu:/home/vboxuser# snap list microstack
Name      Version  Rev  Tracking    Publisher  Notes
microstack ussuri   245  latest/beta canonical** -
```

Fig 6.9: MicroStack Installing

A terminal window titled 'Terminal' with a dark background and light-colored text. The window shows the process of installing and initializing MicroStack. The user is root at an Ubuntu machine. The terminal output includes the installation of the 'microstack' snap, a list of installed snaps, and the execution of 'sudo microstack init --auto --control'. This command triggers a series of log messages indicating the configuration of clustering, control node setup, TLS certificate generation, networking, RabbitMQ, and MySQL server, along with the creation of databases and Keystone configuration. The window's title bar shows 'Activities', 'Terminal', and the date 'Apr 26 23:24'. The terminal's address bar shows 'root@ubuntu: /home/vboxuser'.

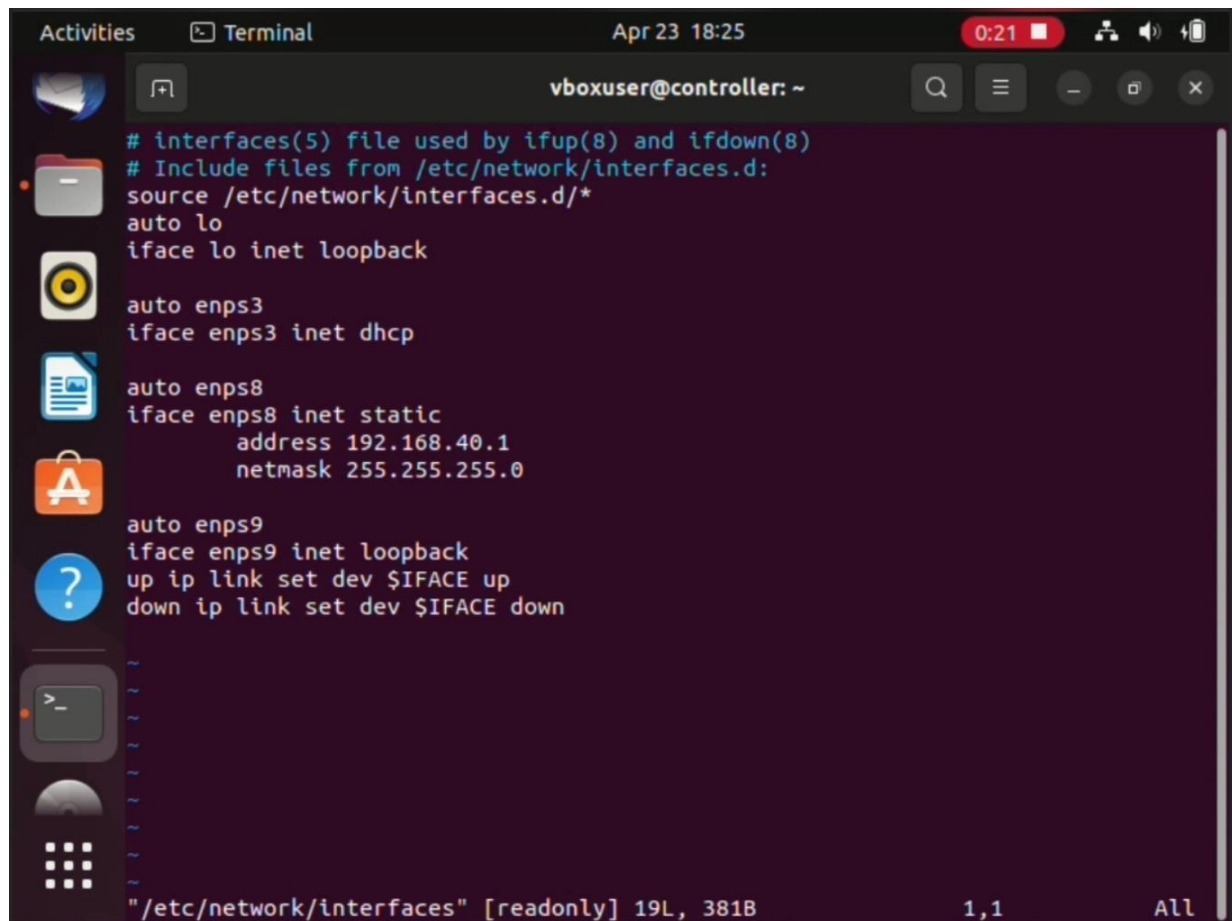
```
root@ubuntu: /home/vboxuser# snap install microstack --beta
microstack (beta) ussuri from Canonical** installed
root@ubuntu: /home/vboxuser# snap list microstack
Name      Version Rev Tracking    Publisher  Notes
microstack ussuri  245   latest/beta canonical** -
root@ubuntu: /home/vboxuser# sudo microstack init --auto --control
2023-04-26 23:06:41,953 - microstack_init - INFO - Configuring clustering ...
2023-04-26 23:06:42,187 - microstack_init - INFO - Setting up as a control node
.
2023-04-26 23:06:46,226 - microstack_init - INFO - Generating TLS Certificate a
nd Key
2023-04-26 23:06:47,632 - microstack_init - INFO - Configuring networking ...
2023-04-26 23:06:58,069 - microstack_init - INFO - Opening horizon dashboard up
to *
2023-04-26 23:06:59,108 - microstack_init - INFO - Waiting for RabbitMQ to star
t ...
Waiting for 10.0.2.15:5672
2023-04-26 23:07:07,877 - microstack_init - INFO - RabbitMQ started!
2023-04-26 23:07:07,877 - microstack_init - INFO - Configuring RabbitMQ ...
2023-04-26 23:07:09,162 - microstack_init - INFO - RabbitMQ Configured!
2023-04-26 23:07:09,220 - microstack_init - INFO - Waiting for MySQL server to
start ...
Waiting for 10.0.2.15:3306
2023-04-26 23:07:18,060 - microstack_init - INFO - Mysql server started! Creati
ng databases ...
2023-04-26 23:07:20,717 - microstack_init - INFO - Configuring Keystone Fernet
Keys ...
2023-04-26 23:07:40,422 - microstack_init - INFO - Bootstrapping Keystone ...
2023-04-26 23:07:56,154 - microstack_init - INFO - Creating service project ...
2023-04-26 23:08:04,655 - microstack_init - INFO - Keystone configured!
```

Fig 6.10: MicroStack Initialization

```
Activities Terminal Apr 26 23:24 0:34
root@ubuntu: /home/vboxuser

as initialized!
root@ubuntu:/home/vboxuser# microstack.openstack image list
+-----+-----+-----+
| ID | Name | Status |
+-----+-----+-----+
| 980a8eef-d8e9-44e1-aa5a-960972fe8bf1 | cirros | active |
+-----+-----+-----+
root@ubuntu:/home/vboxuser# microstack.opestack flaovor list
microstack.opestack: command not found
root@ubuntu:/home/vboxuser# microstack.openstack flaovor list
openstack: 'flaovor list' is not an openstack command. See 'openstack --help'.
Did you mean one of these?
  flavor create
  flavor delete
  flavor list
  flavor set
  flavor show
  flavor unset
root@ubuntu:/home/vboxuser# microstack.openstack flavor list
+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | RAM | Disk | Ephemeral | VCPUs | Is Public |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | m1.tiny | 512 | 1 | 0 | 1 | True |
| 2 | m1.small | 2048 | 20 | 0 | 1 | True |
| 3 | m1.medium | 4096 | 20 | 0 | 2 | True |
| 4 | m1.large | 8192 | 20 | 0 | 4 | True |
| 5 | m1.xlarge | 16384 | 20 | 0 | 8 | True |
+-----+-----+-----+-----+-----+-----+-----+
root@ubuntu:/home/vboxuser#
```

Fig 6.11: Flavor List



The image shows a terminal window titled "Terminal" with the username "vboxuser@controller: ~". The terminal displays the contents of the file `/etc/network/interfaces`, which is used by `ifup(8)` and `ifdown(8)`. The file includes configuration for several network interfaces: `lo` (loopback), `enps3` (DHCP), `enps8` (static IP 192.168.40.1), and `enps9` (loopback). The `enps9` interface is configured with `up` and `down` scripts to set the device state. The terminal also shows the file's metadata: `"/etc/network/interfaces" [readonly] 19L, 381B` and the cursor position `1,1` with the text `All`.

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source /etc/network/interfaces.d/*

auto lo
iface lo inet loopback

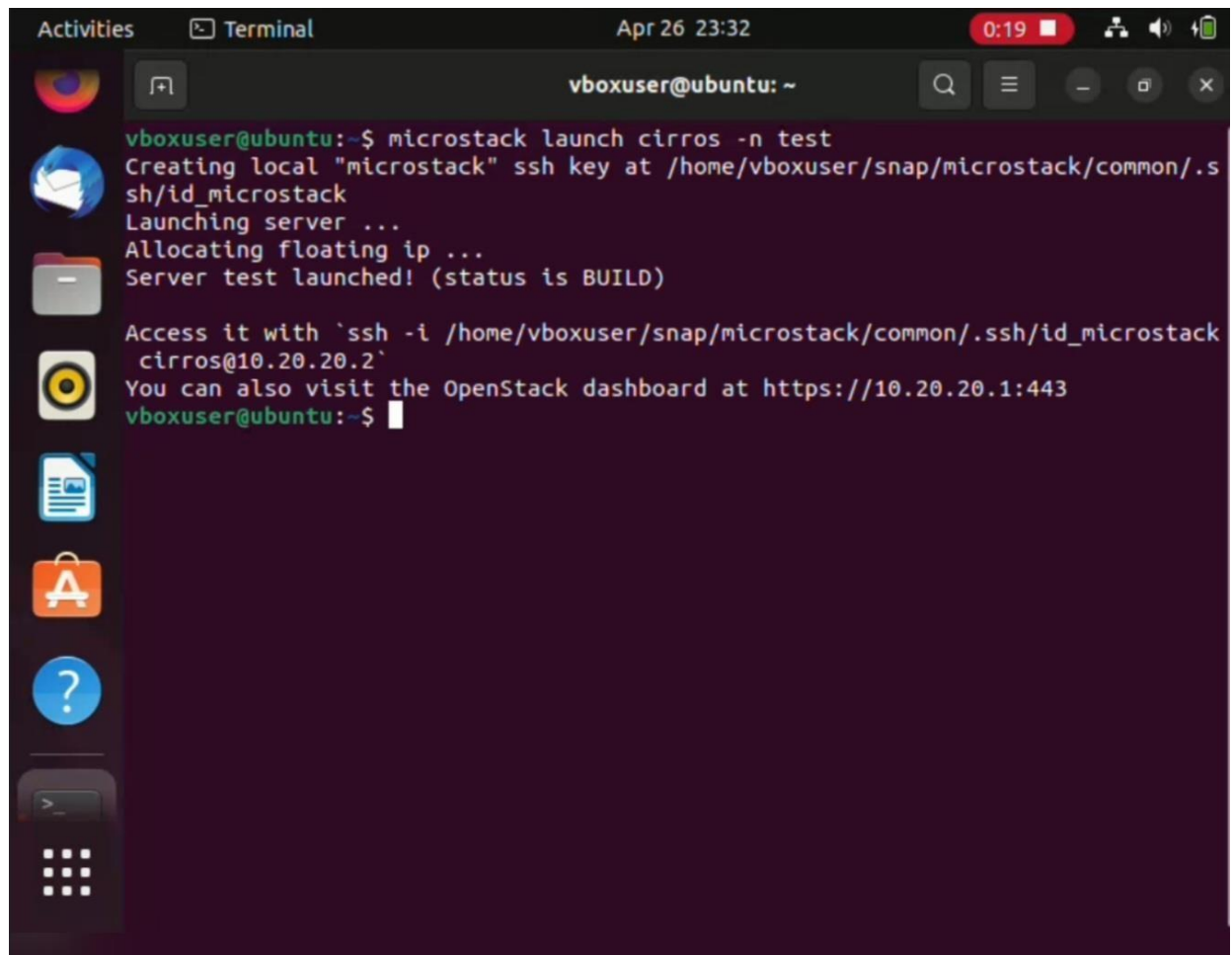
auto enps3
iface enps3 inet dhcp

auto enps8
iface enps8 inet static
    address 192.168.40.1
    netmask 255.255.255.0

auto enps9
iface enps9 inet loopback
up ip link set dev $IFACE up
down ip link set dev $IFACE down

"/etc/network/interfaces" [readonly] 19L, 381B 1,1 All
```

Fig 6.12: *Inet Loopback*



The image shows a terminal window titled 'Terminal' with the date and time 'Apr 26 23:32'. The user is 'vboxuser@ubuntu: ~'. The command entered is 'microstack launch cirros -n test'. The output shows the creation of an SSH key, launching the server, allocating a floating IP, and launching the server test. The status is 'BUILD'. The user is instructed to access it with 'ssh -i /home/vboxuser/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.2' and to visit the OpenStack dashboard at 'https://10.20.20.1:443'. The terminal window has a sidebar with various application icons.

```
vboxuser@ubuntu:~$ microstack launch cirros -n test
Creating local "microstack" ssh key at /home/vboxuser/snap/microstack/common/.ssh/id_microstack
Launching server ...
Allocating floating ip ...
Server test launched! (status is BUILD)

Access it with `ssh -i /home/vboxuser/snap/microstack/common/.ssh/id_microstack cirros@10.20.20.2`
You can also visit the OpenStack dashboard at https://10.20.20.1:443
vboxuser@ubuntu:~$
```

Fig 6.13: MicroStack Launch Cirros

Chapter 7

REFERENCES

1. Jain, N., & Garg, R. (2021). An Overview of OpenStack Cloud Computing Platform. *International Journal of Advanced Research in Computer Science and Software Engineering*, 11(5), 56-63.
2. Sridharan, D., & Hancock, D. (2018). Performance Evaluation of OpenStack Nova for Scientific Computing. *Journal of Cloud Computing*, 7(1), 1-16.
3. Righi, R., & Westphall, C. B. (2019). OpenStack for Enterprises: A Systematic Review. *Journal of Network and Systems Management*, 27(2), 314-346.
4. Alqahtani, A., & Kumar, S. (2021). Cloud Computing Security Issues and Challenges: A Survey. *Journal of King Saud University - Computer and Information Sciences*, 33(3), 347-359.
5. Jackson, K., Bunch, C., & Sigler, E. (2012). *OpenStack Cloud Computing Cookbook*. Packt Publishing.
6. OpenStack. (2022). OpenStack Compute (Nova) Documentation. Retrieved from <https://docs.openstack.org/nova/latest/>
7. OpenStack. (2022). OpenStack Image Service (Glance) Documentation. Retrieved from <https://docs.openstack.org/glance/latest/>
8. Ruan, F., Chen, H., & Fang, J. (2019). Research on OpenStack Based Cloud Computing Platform. *Journal of Physics: Conference Series*, 1168(4), 1-7.
9. Liu, S., Wang, Z., & Wang, J. (2016). A Study of OpenStack Performance in Cloud Computing. *Journal of Information Processing Systems*, 12(4), 662-671.
10. Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware

resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5), 755-768.