A project report on

BUILDING A VIRTUALIZED CLOUD WITH OPENSTACK

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

by

SUDULAGUNTA BRAHMANYA ASRIT (20BCE7236)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

MAY, 2024

BUILDING A VIRTUALIZED CLOUD WITH OPENSTACK

Submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

by

SUDULAGUNTA BRAHMANYA ASRIT (20BCE7236)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

MAY, 2024

DECLARATION

I here by declare that the thesis entitled “Building a Virtualized Cloud with Openstack” submitted by me, for the award of the degree of Specify the name of the degree VIT is a record of bonafide work carried out by me under the supervision of Dr. K. Hemanth Kumar Reddy.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Amaravati

S·B·Asrit

Date: 31/5/24 Signature of the Candidate

भारत सरकार
अन्तरिक्ष विभाग
राष्ट्रीय सुदूर संवेदन केन्द्र
बालानगर, हैदराबाद-500 037, तेलंगाण, भारत
टेलिफोन : +040-23879572-76
+040-23879261-65
फैक्स : +040-23878648

सत्यमेव जयते

Government of India
Department of Space
National Remote Sensing Centre
Balanagar, Hyderabad - 500 037, Telangana, India
Telephone : +040-23879572-76
+040-23879261-65
Fax : +040-23878648

Certificate

Date: 24-04-2024

This is to certify that the project entitled "**Building a Virtualized Cloud with Openstack**" is a bonafide work carried out by **Sudulagunta Brahmanya Asrit**, at National Remote Sensing Centre, Hyderabad during the period from 25 January to 25 April under my guidance.

The student is pursuing Bachelor of Technology in Computer Science and Engineering from **Vellore Institute of Technology, AP**

He has completed the assigned task successfully.

Name: Sudulagunta Brahmanya Asrit

Roll no: 20BCE7236

Project Guide

SVSRK Kishore
GH SISG, SISG/MSA

एस.वी.एस.आर.के. किशोर / S.V.S.R.K. KISHORE
समूह प्रधान, सिस्टम एवं आईटी समाधान समूह
Group Head, Systems & IT Solutions Group
प्रबंधन प्रणाली क्षेत्र / Management Systems Area
राष्ट्रीय सुदूर संवेदन केंद्र / National Remote Sensing Centre
इसरो, अन्तरिक्ष विभाग, भारत सरकार
ISRO, Dept. of Space Govt. of India
बालानगर, हैदराबाद / Balanagar, Hyderabad-500037.T.S. India

भारत सरकार
अन्तरिक्ष विभाग
राष्ट्रीय सुदूर संवेदन केन्द्र
बालानगर, हैदराबाद -500 037, तेलंगाना, भारत
टेलिफोन : +040-23879572-76
+040-23879261-65
फैक्स :+040-23878648

Government of India
Department of Space
National Remote Sensing Centre
Balanagar, Hyderabad - 500 037, Telangana, India
Telephone : +040-23879572-76
+040-23879261-65
Fax : +040-23878648

Project Closure Certificate

Date:

This is to certify that the below mentioned student has completed the assigned work successfully at National Remote Sensing Centre, Hyderabad during the period: 25-01-2024 to 25-04-2024

STUDENT NAME: Sudulagunta Brahmanya Asrit

जया सक्सेना

डॉ. जया सक्सेना
Dr. Jaya Saxena
वैज्ञानिक 'एफ' एंव प्रमुख/Scientist 'F' & Head
छात्र परियोजना इंटरफेस प्रभाग, टीइओजी
Student Project Interface Division, TEOG
राष्ट्रीय सुदूर संवेदन केन्द्र/National Remote Sensing Centre
इसरो/डॉस, भारत सरकार/ISRO/DOS, Govt. of India
बालानगर, हैदराबाद/Balanagar, Hyderabad - 500 037

Dr. Jaya Saxena,
Scientist 'F' & Head,
Student Project Interface Division,
Training, Education & Outreach Group,
Management Systems Area (MSA)
NRSC (ISRO), HYDERABAD

CERTIFICATE

This is to certify that the Internship titled **"Building a Virtualized Cloud with Openstack"** that is being submitted by **SUDULAGUNTA BRAHMANYA ASRIT (20BCE7236)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

K Hemmant Reddy

Internal Guide

Dr. Hemanth Kumar Reddy

The thesis is satisfactory / unsatisfactory

Internal Examiner1

Dr. Aravapalli Rama Satish

Internal Examiner2

Dr. Koduru Hajarathaiah

Approved by

Dr.G.Muneeswari . HoD

School of Computer Science and Engineering

ABSTRACT

National Remote Sensing Centre (NRSC) is one of the primary centres of Indian Space Research Organisation (ISRO), Department of Space (DOS). NRSC has the mandate for establishment of ground stations for receiving satellite data, generation of data products, dissemination to the users, development of techniques for remote sensing applications. NRSC is the nodal centre for hosting Satellite Data Products from more than 13 IRS satellites. The evolution of the remote sensing technologies drives innovation in IT, including data storage, processing, and visualization techniques.

Remote sensing generates massive amounts of data, which require advanced IT infrastructure and algorithms for processing and analysis. Cloud technology provide flexibility, scalability, and efficiency required to effectively manage and analyze remote sensing data. In NRSC, cloud technologies are already in place to enhance the remote sensing and geospatial capabilities while efficiently managing resources and costs.

The development of a Virtualized cloud portal represents a transformative approach to cloud computing provisioning and management. The virtualized cloud portal serves as a centralized platform for users to autonomously provision, manage, and monitor cloud resources. Through an intuitive user interface, users can easily request and deploy virtual machines, storage, networking, and other cloud services. Additionally, the portal offers Virtualized capabilities for scaling resources up or down dynamically, optimizing cost efficiency and resource utilization.

Furthermore, the portal incorporates robust monitoring and reporting functionalities, providing users with real-time insights into resource usage, performance metrics, and cost analytics.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. K.Hemanth Kumar Reddy, Assistant Professor, SCOPE, VIT-AP, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Cloud Computing.

I would like to express my gratitude to Dr. G. Viswanathan, Dr. G. V. Selvam, Mr. Sankar Viswanathan, Dr. Sekar Viswanathan, Dr. S. V. Kota Reddy and Dr. CH. Pradeep Reddy School of Computer Science and Engineering (SCOPE), for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr.G.Muneeswari . HoD / DDSE, all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Amaravati

Date: 31/5/2024 Sudulagunta Brahmanya Asrit

Table of Contents

Contents

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

NRSC is the nodal centre for hosting Satellite Data Products from more than 13 IRS satellites. The continuous evolution of remote sensing technologies propels innovation in IT, particularly in data storage, processing, and visualization techniques. Given the immense volume of data generated by remote sensing, sophisticated IT infrastructure and algorithms are essential for processing and analysis.

Recognizing this need, NRSC has already integrated cloud technologies into its operations. Cloud technology offers the flexibility, scalability, and efficiency required to manage and analyze remote sensing data effectively. With cloud technology in place, NRSC can enhance its remote sensing and geospatial capabilities while optimizing resource utilization and reducing costs. This integration underscores NRSC's commitment to leveraging cutting-edge IT solutions to fulfill its mission of providing valuable geospatial information for various applications.

1.2 CLOUD COMPUTING PLATFORM

Cloud computing platforms leverage virtualization technology to provide users with access to computing resources over the internet. These platforms utilize virtualization to abstract physical hardware into virtual instances, allowing users to deploy and manage virtual machines (VMs), storage, networking, and other resources remotely.

Virtualization enables cloud computing platforms to offer services such as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). In IaaS, users can access virtualized computing resources like VMs, storage, and networking infrastructure on-demand. PaaS provides a platform for developers to build, deploy, and manage applications without needing to worry about underlying infrastructure. SaaS delivers software applications over the internet, often running on virtualized infrastructure managed by the cloud provider.

Leading cloud computing platforms, such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), leverage virtualization extensively to offer scalable, flexible, and cost-effective solutions for businesses and individuals. Virtualization is thus a crucial component in the architecture of cloud computing platforms, enabling the efficient utilization of hardware resources and facilitating the delivery of a wide range of cloud services.

At NRSC, virtualization forms the core of the computing infrastructure, supporting services such as web hosting, FTP, and software development through virtual instances or VMs. NRSC already

uses high-quality commercial applications to manage these virtualized environments. Towards this, it was proposed to develop a virtualized computing platform with Self Service portal using open-source solutions, and OpenStack is one such freely available option.

1.3 OPENSTACK

Firstly, OpenStack is an open-source platform, reducing licensing costs and providing greater flexibility for customization to specific needs. Secondly, OpenStack boasts a vibrant and active community of developers and contributors, ensuring continuous improvement, updates, and support. This community-driven approach fosters innovation and ensures that the platform remains relevant and responsive to evolving requirements.

Another key benefit is the scalability and flexibility offered by OpenStack. It allows users to build and manage large-scale, distributed cloud environments, accommodating varying workloads and adapting to changing demands effortlessly. OpenStack's modular architecture enables users to choose and integrate only the components they need, tailoring the platform to suit specific use cases and avoiding unnecessary complexity.

Moreover, OpenStack provides comprehensive support for various virtualization technologies, including KVM, Xen, and VMware, offering compatibility with existing infrastructures and enabling seamless migration of workloads. Additionally, OpenStack supports multi-tenancy, enabling efficient resource sharing and isolation between different users or projects.

Chapter 2

LITERATURE SURVEY

2.1 KEY FEATURES OF OPENSTACK

Key features of OpenStack Cloud computing platform are listed below:

a. **Modular Architecture:** OpenStack is composed of numerous interoperable services (projects) that you can mix and match based on your specific cloud requirements. This modularity provides flexibility and scalability.

b. **API-Driven:** OpenStack is API-driven, allowing users to automate tasks, integrate with other tools and build custom cloud solutions. This simplifies management and customization.

c. **Virtualized Portal:** OpenStack offers a web-based dashboard (Horizon) for both administrators and cloud tenants. Administrators can configure and manage the cloud infrastructure, while tenants can provision, manage, and monitor their own cloud resources. This Virtualized approach streamlines resource allocation and usage.

d. **Open Source:** Being open-source, OpenStack gives you access to its source code, allowing for customization, community support and cost-effectiveness compared to proprietary solutions.

e. **Scalability:** OpenStack is designed to scale horizontally, allowing organizations to seamlessly expand their cloud infrastructure as demand grows. This scalability ensures that OpenStack can accommodate varying workloads and evolving business needs without compromising performance or stability. Organizations can easily add additional compute, storage, and networking resources to their OpenStack deployment, ensuring agility and efficiency in resource management.

2.2 HISTORY OF OPENSTACK

OpenStack, an open-source cloud computing platform, emerged from a collaboration between Rackspace Hosting and NASA in July 2010. The goal was to develop a cloud computing platform that would be both flexible and scalable, providing infrastructure-as-a-service (IaaS) solutions to users.

Here's a brief history:

a. **Inception (2010):** OpenStack was launched as a joint project between Rackspace and NASA, with Rackspace contributing its cloud storage platform, Cloud Files, and NASA contributing its open-source cloud platform, Nebula. The first release, code-named "Austin," laid the foundation for subsequent developments.

b. **Expansion and Growth (2011-2013):** OpenStack quickly gained traction in the tech community. The project attracted numerous contributors and expanded beyond its initial sponsors. Major companies like IBM, Red Hat, HP, and Intel joined the initiative, contributing resources and expertise to its development. The community released several updates, each enhancing the platform's capabilities and stability.

c. **Maturation (2014-2016):** During this period, OpenStack continued to mature as a platform. The project released several significant updates, adding features like support for containers and software-defined networking (SDN). The user base expanded beyond tech companies to include enterprises from various industries seeking scalable cloud solutions.

d. **Diversification (2017-2019):** OpenStack's ecosystem diversified, with developers focusing on making the platform more modular and interoperable. This period saw increased integration with emerging technologies like Kubernetes for container orchestration and edge computing for distributed environments. OpenStack also gained prominence in telecommunications and research sectors.

e. **Current Developments (2020-present):** OpenStack remains a vibrant open-source project, continuously evolving to meet the changing demands of cloud computing. Efforts are underway to improve scalability, enhance security features, and streamline operations. The community continues to expand, with contributions from developers, enterprises, and research institutions worldwide.

Throughout its history, OpenStack has remained committed to its core principles of openness, interoperability, and community-driven development. It has become one of the leading open-source cloud computing platforms, powering a wide range of private and public cloud deployments globally.

2.3 COMPONENTS IN OPENSTACK

Apart from various projects which constitute the OpenStack platform, there are nine major services namely Nova, Neutron, Swift, Cinder, Keystone, Horizon, Ceilometer, and Heat. Here is the basic definition of all the components which will give us a basic idea about these components.

a. **Nova (compute service):** It manages the compute resources like creating, deleting, and handling the scheduling. It can be seen as a program dedicated to the automation of resources that are responsible for the virtualization of services and high-performance computing.

Nova Components

1. ***nova-api:*** The nova-api service handles API requests, including creation, deletion, and management of instances. It supports both OpenStack API and EC2 API. It processes API calls, validates requests, and forwards them to other Nova components for processing. It also supports RESTful APIs for integration with external systems.

2. **nova-compute**: The nova-compute service is the core component responsible for creating and terminating virtual machine instances through hypervisors like KVM, Xen, VMware, or Hyper-V. It runs on the hypervisor host and communicates with the nova-scheduler to determine the placement of instances. It also manages instance lifecycle operations like snapshotting and resizing.

3. ***nova-scheduler:*** The nova-scheduler service determines which compute node should run a new instance based on defined policies and resource availability. It uses filters and weighs mechanisms to select the optimal host. Filters might include available memory, CPU, disk space, and other criteria.

4. ***nova-conductor:*** The nova-conductor service acts as a mediator between nova-compute and the database to handle database operations. It improves security by preventing direct database access from compute nodes. It also helps in scaling by offloading complex operations from the compute nodes.

5. ***nova-consoleauth:*** The nova-consoleauth service authorizes users for accessing instance consoles. It manages tokens for authenticating console access, typically for VNC or SPICE protocols.

6. ***nova-novncproxy:*** These services provide proxy access to virtual machine consoles using VNC or SPICE protocols. They enable remote management of instances through a web browser interface.

7. ***nova-cert:*** The nova-cert service manages X.509 certificate generation for instances. It is mainly used for the EC2 API but is not commonly deployed in newer OpenStack installations.

b. Neutron (networking service): It is responsible for connecting all the networks across OpenStack. It is an API driven service that manages all networks and IP addresses.

Neutron Components

1. ***neutron-server***: The neutron-server is the central component that provides the Neutron API and routes API requests to the appropriate Neutron plugin.It handles REST API calls, processes them, and interacts with the database and other Neutron components.

2. ***neutron Plugins:*** Plugins are responsible for implementing the actual networking functionality. Different plugins are used for different types of networking backends. Examples include the Open vSwitch (OVS) plugin, Linux Bridge plugin, and vendor-specific plugins like those from Cisco, Juniper, and VMware.

3. ***neutron-dhcp-agent:*** The DHCP agent provides DHCP services to tenant networks. It ensures that instances receive IP addresses and other network configuration parameters dynamically.

4. neutron-l3-agent: The L3 agent provides routing and NAT (Network Address Translation) services for tenant networks. It manages virtual routers and external network gateways, enabling instances to communicate with external networks, including the internet.

5. ***neutron-metadata-agent:*** The metadata agent provides a way for instances to access instance-specific metadata. It proxies requests from instances to the Nova metadata service, allowing instances to retrieve configuration information.

6. ***neutron-openvswitch-agent***: The Open vSwitch agent manages networking on compute nodes using Open vSwitch (OVS). It handles the creation of virtual network bridges, ports, and tunnels, ensuring that instances can communicate over the network.

7. ***neutron-linuxbridge-agent:*** The Linux Bridge agent manages networking on compute nodes using Linux bridges. It performs similar functions to the OVS agent but uses Linux bridge mechanisms instead.

8. ***neutron-sriov-nic-agent***: The SR-IOV NIC agent manages networking for Single Root I/O Virtualization (SR-IOV) enabled network interfaces. It provides high-performance networking by allowing virtual functions (VFs) of physical network interfaces to be directly assigned to instances.

c. **Swift (object storage):** It is an object storage service with high fault tolerance capabilities and it used to retrieve unstructured data objects with the help of Restful API. Being a distributed platform, it is also used to provide redundant storage within servers that are clustered together. It is able to successfully manage petabytes of data.

Swift Components

1. ***swift-proxy-server:*** The proxy server handles incoming API requests from clients. It routes requests to the appropriate storage nodes, handles authentication, and manages object metadata. It also implements the Swift REST API for object storage operations like PUT, GET, DELETE, and metadata operations.

2. ***swift-account-server:*** The account server manages account metadata.It keeps track of the storage usage and metadata for user accounts, such as the number of containers and objects within an account. Each account is stored in a database file that is replicated across multiple storage nodes.

3. ***swift-container-server:*** The container server manages container metadata.It tracks the objects stored within a container and maintains metadata about the container itself. Similar to the account server, each container's metadata is stored in a replicated database file.

4. ***swift-object-server:*** The object server is responsible for storing, retrieving, and deleting the actual data objects. Objects are stored on the filesystem of the storage nodes, and the object server manages these files. It uses a straightforward storage scheme where each object is stored as a file on disk, and the directory structure is used for object namespacing.

5. ***swift-replicator:*** The replicator services ensure that data is replicated across multiple nodes to achieve high availability and durability. There are separate replicators for accounts, containers, and objects, each responsible for synchronizing the respective data across different nodes based on the replication strategy.

6. ***swift-updater:*** The updater services are responsible for updating the account and container databases to reflect changes made to objects. They ensure eventual

consistency by propagating changes across the cluster, making sure that metadata and actual object data are consistent.

7. ***swift-auditor:*** The auditor services periodically check the integrity of data stored in the system. Separate auditors exist for accounts, containers, and objects, each verifying that data is not corrupted and matches its expected state.

d. Cinder (block storage): It is responsible for providing persistent block storage that is made accessible using an API (self- service). Consequently, it allows users to define and manage the amount of cloud storage required.

Cinder Components

1. ***cinder-api:*** The cinder-api service handles incoming REST API requests and routes them to the appropriate Cinder components. It provides endpoints for volume management, such as creating, listing, attaching, detaching, and deleting volumes. The API service also supports authentication and authorization via Keystone.

2. ***cinder-scheduler:*** The cinder-scheduler is responsible for selecting the appropriate storage backend and storage node for new volume requests. It uses various filters and weighers to determine the best placement for a volume based on factors such as available capacity, backend capabilities, and performance metrics.

3. ***cinder-volume:*** The cinder-volume service manages the lifecycle of volumes, including creation, deletion, attachment, and detachment. It interacts with the storage backends to perform volume operations. Each cinder-volume service instance can manage multiple backends, defined by configuration.

4. ***cinder-backup:*** The cinder-backup service provides functionality for backing up Cinder volumes to a backup storage backend. Supported backends include Swift, NFS, and other object storage systems. The backup service ensures that data can be recovered in case of volume failure or corruption.

5. ***cinder-snapshot***: The cinder-snapshot service manages the creation and deletion of snapshots for volumes. Snapshots capture the state of a volume at a specific point in time, allowing for data recovery or volume cloning.

6. ***cinder-restore***: The cinder-restore service handles restoring volumes from backups. It works with the cinder-backup service to ensure that volumes can be restored to their original state or to a new volume from backup data.

7. c***inder-manager:*** The cinder-manager service oversees the coordination and management of volume-related tasks across different Cinder services. It ensures that tasks are executed correctly and in order, handling retries and error management.

e. **Keystone (identity service provider):** It is responsible for all types of authentications and authorizations in the OpenStack services. It is a directory-based service that uses a central repository to map the correct services with the correct user.

Keystone Components

1. ***keystoneapi:*** The keystoneapi service handles incoming REST API requests for authentication and authorization. It provides endpoints for managing users, groups, projects, roles, and tokens. The API service ensures secure access to OpenStack services by validating tokens and checking user permissions.

2. ***keystoneauth:*** The keystoneauth service manages user authentication and token generation. It supports various authentication methods, including username/password, tokenbased authentication, and federation with external identity providers. Keystoneauth issues tokens that other OpenStack services use to verify user identity.

3. ***keystoneidentity:*** The keystoneidentity service handles identity management, including user, group, and domain management. It maintains the identity backend, which can be an SQL database, LDAP directory, or other identity stores. Keystoneidentity ensures that user credentials and attributes are securely stored and managed.

4. keystoneresource: The keystoneresource service manages resources such as projects, domains, and roles. It organizes users and groups into projects and domains, defining the scope of their permissions. Roles are assigned to users or groups within specific projects or domains to control access levels.

5. ***keystonepolicy***: The keystonepolicy service manages authorization policies that define what actions users can perform. Policies are defined in JSON format and specify permissions based on user roles, projects, and resources. Keystonepolicy enforces these policies across all OpenStack services.

6. ***keystonecatalog***: The keystonecatalog service maintains a service catalog that lists all available OpenStack services and their endpoints. When users authenticate, they receive a token and the service catalog, which includes the URLs for the various OpenStack services they can access. This allows seamless integration and discovery of services.

7. ***keystonetoken:*** The keystonetoken service issues and validates tokens used for authenticating API requests. Tokens can be temporary (UUID tokens) or more secure

(PKI or Fernet tokens). Keystonetoken ensures that tokens are valid and have not expired before allowing access to services.

f. **Glance (image service provider):** It is responsible for registering, storing, and retrieving virtual disk images from the complete network. These images are stored in a wide range of back-end systems.

Glance Components

1. ***glanceapi***: The glanceapi service handles incoming HTTP/HTTPS requests for image management. It supports operations like image upload, retrieval, deletion, and listing. The API service validates requests, interacts with the database, and manages image metadata.

2. ***glanceregistry:*** The glanceregistry service manages metadata for images stored in the Glance database. It stores information such as image size, type, checksum, and creation date. The registry service is essential for tracking and managing image metadata but has been deprecated in favor of using the glanceapi directly for metadata storage in newer releases.

3. ***glancescrubber:*** The glancescrubber service is responsible for cleaning up images marked for deletion. When an image is deleted, it is first marked for deletion. The scrubber service then periodically checks for these images and permanently removes them from the backend storage, freeing up space.

4. ***glancecache:*** The glancecache service manages a local cache of images to improve performance. It speeds up image retrieval operations by caching frequently accessed images locally on the compute nodes, reducing the need to fetch images from the backend storage repeatedly.

5. ***glanceimport:*** The glanceimport service facilitates the import of images from external sources. It supports various image import methods, such as direct upload, web download, and copying from other storage locations. This service ensures that images can be easily imported and used within the OpenStack environment.

Glance Image Formats and Storage Backends

1. ***Supported Image Formats***: Glance supports various image formats, including QCOW2, RAW, VHD, VMDK, ISO, and more. This flexibility allows users to work with images created by different virtualization platforms.

2. ***Local filesystem***: Storing images on the local disk of the Glance node.

3. ***Swift:*** Using OpenStack Swift object storage for scalable and redundant image storage.

4. ***Ceph RADOS***: Integrating with Ceph to use its object storage capabilities.

5. ***NFS:*** Using Network File System (NFS) for shared storage.

6. ***Other object stores:*** Integrating with various thirdparty object storage solutions like Amazon S3 or Google Cloud Storage.

g. Horizon (dashboard): It is responsible for providing a web-based interface for OpenStack services. It is used to manage, provision, and monitor cloud resources.

Horizon Components

1. ***horizondashboard:*** The main component that serves the web interface, offering access to various OpenStack services. It organizes the dashboard into different panels and tabs for easy navigation. Users can perform tasks such as launching instances, creating networks, managing images, and more through this interface.

2. ***horizonadmindashboard:*** A specialized section of the Horizon dashboard designed for administrative tasks. It includes additional capabilities for managing users, projects, roles, and quotas, providing administrators with the tools needed to oversee and control the OpenStack environment.

3. ***horizonapi:*** The backend service that handles communication between the Horizon web interface and other OpenStack services. It translates user actions into API calls to various OpenStack components like Nova, Neutron, Cinder, and Glance, enabling the dashboard to interact with the cloud infrastructure.
4. ***horizonsettings:*** Configuration settings that define the behavior and appearance of the Horizon dashboard. Administrators can customize aspects such as enabled panels, themes, and integrations with external services through configuration files.

h. Ceilometer (telemetry): It is responsible for metering and billing of services used. Also, it is used to generate alarms when a certain threshold is exceeded.

Ceilometer Components

1. ***ceilometer-api:*** The ceilometer-api service provides a REST API for querying metering data. It supports queries for resource usage, event data, and billing information, allowing users and administrators to retrieve telemetry data.

2. ***ceilometer-agent:*** Ceilometer agents collect and report metering data from various OpenStack services. Agents are deployed on compute nodes, network nodes, and other relevant infrastructure components to monitor resource utilization, instance usage, network traffic, and more.

3. ***ceilometer-collector:*** The ceilometer-collector service receives metering data from agents and stores it in the database. It aggregates, validates, and processes incoming data before storing it in the Ceilometer database for querying and analysis.

4. ***ceilometer-notification:*** Ceilometer notifications capture events and triggers based on predefined rules. Notifications are used for billing, alarming, and generating usage reports. They help track changes in resource state and detect anomalies.

5. ***ceilometer-alarm:*** The ceilometer-alarm service manages alarm configurations and triggers actions based on thresholds or conditions. Users can set up alarms to monitor resource metrics (e.g., CPU usage, network bandwidth) and receive notifications or perform automated actions when thresholds are exceeded.

i. **Heat (orchestration):** It is used for on-demand service provisioning with auto-scaling of cloud resources. It works in coordination with the ceilometer.

Heat Components

1. ***heat-api:*** The heat-api service handles incoming API requests for orchestration operations. It provides endpoints for creating and managing stacks (collections of resources), as well as handling template validation and policy enforcement.

2. ***heat-engine:*** The heat-engine service is responsible for executing orchestration tasks. It interprets templates, communicates with OpenStack services to create or modify resources, monitors stack progress, and handles stack events.

3. ***heat-cfn:*** The heat-cfn service provides compatibility with AWS CloudFormation API. It allows users familiar with CloudFormation to use the same templates and tools to manage OpenStack resources through the AWS-like API.

4. ***heat-dashboard:*** The Heat dashboard provides a graphical user interface for managing stacks and templates. It allows users to create, view, update, and delete stacks using a web-based interface, providing a visual representation of the orchestration process.

Chapter 3

METHODOLOGY

This project aims to design, deploy, and evaluate an OpenStack-based Virtualized cloud infrastructure to meet the growing demands of modern businesses.

3.1 PRIMARY OBJECTIVES

a. **Infrastructure Design:** Design a scalable and resilient cloud infrastructure using OpenStack, comprising compute, storage, and networking components.

b. **Deployment and Configuration:** Implement the designed infrastructure by deploying and configuring OpenStack components, ensuring seamless integration and optimal performance.

c. **User Interface Development:** Develop a user-friendly web interface leveraging OpenStack Horizon to provide Virtualized capabilities for provisioning and managing cloud resources.

d. **Integration and Compatibility:** Ensure seamless integration with a wide range of existing infrastructure and technologies, fostering compatibility with industry standards and enabling interoperability across different cloud environments.

e. **Security and Access Control:** Implement robust security measures using OpenStack Keystone to ensure secure authentication, authorization, and data protection within the cloud environment.

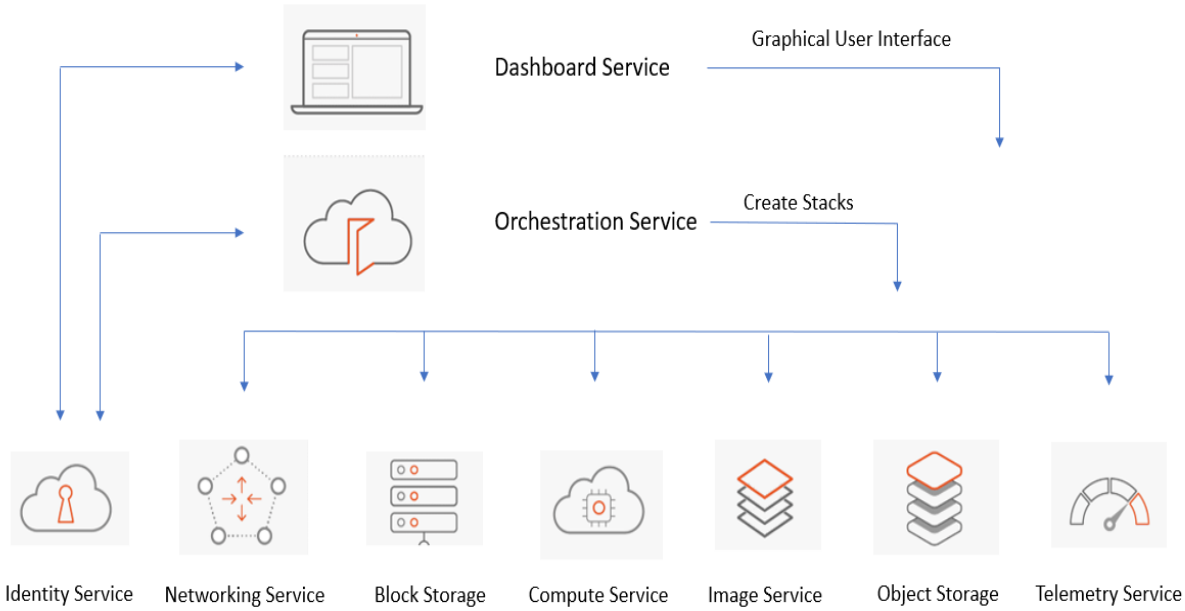
f. **Performance Monitoring and Optimization**: Implement monitoring tools and performance optimization techniques to continuously monitor and enhance the performance and efficiency of the OpenStack cloud infrastructure.

3.2 DEPLOYMENT ARCHITECTURE

The proposal is to build a virtualized computing platform using OpenStack. OpenStack is an open-source cloud platform designed to manage distributed compute, network and storage resources in the data center. In principle, OpenStack aggregates physical resources into one big pool and allocates virtual resources out of this pool to users who can request them on-demand through a Virtualized portal or application programming interfaces (APIs). But OpenStack itself does not handle virtualization. Instead, it leverages the existing virtualization technologies. Therefore,

Openstack is more like a wrapper around traditional virtualization tools, enabling cloud-native capabilities.

The architecture of OpenStack deployment is given below:



3.3 SYSTEM REQUIREMENTS FOR INSTALLATION

It was proposed to create test instances with the following configurations to begin with the setup.

Resource	**Minimum Requirement**	**Available Resources**	**Meets Requirement?**
Linux OS	Centos/Ubuntu	Centos 8 Stream	Yes (ample headroom)
RAM	16 GB	256 GB Server	Yes (ample headroom)

CPUs	4 CPUs (physical or virtual)	80 CPUs (physical)	Yes (ample headroom)
Internal Storage	100 GB	Disk 1: 512 GB, Disk 2: 1.9 TB	Yes (significant headroom)
Network	Any assigned IP	One assigned IP	Yes (assuming it's a usable IP for OpenStack)

3.4 SYSTEM DETAILS

OpenStack was installed on a VM configured on CentOS base system. The details of the base and VM are given below.

Feature	**Base Machine Configuration**	**VM Configuration**
Operating System	CentOS Stream 8	CentOS Stream 8
Hardware	RAM: 256 GB Disk 1: 512 GB Disk 2: 1.9 TB	RAM: 64 GB Volume1: 200GB vdb:300 GB (Cinder) vdc:300 GB (Cinder)
Software		- Specific OpenStack services (e.g., Nova, Cinder, Glance)
Resource Allocation	Physical Cores: 80 CPUS	Virtual Cores: 40 VCPUS
Purpose	Running Virtual Machine	Deploying OpenStack
Security	- SELinux can be disabled (not recommended) or configured for OpenStack compatibility.	- Security considerations apply to each VM running an OpenStack service.

3.5 INSTALLATION PROCESS

Install OpenStack Yoga on CentOS 8 Stream with Packstack

Packstack is command line utility that uses Puppet modules to deploy various parts of OpenStack on multiple pre-installed servers over SSH automatically. Currently it only supports deployment on CentOS, Red Hat Enterprise Linux (RHEL) and compatible derivatives of both are supported.

Step 1*:* Set hostname, DNS and Update System

i. *sudo hostnamectl set-hostname openstack.example.com*

ii. *$ sudo vi /etc/hosts*
172.x.x.x openstack.example.com

iii. *sudo dnf update -y & sudo reboot*

Step 2: Enable repositories, disable NetworkManager

i. *sudo dnf config-manager --enable powertools*

ii. *sudo dnf -y install network-scripts*
readlink $(readlink $(which ifup))
sudo touch /etc/sysconfig/disable-deprecation-warnings

iii. *sudo systemctl disable --now NetworkManager*
sudo systemctl enable network
sudo systemctl start network

iv. *sudo systemctl disable --now firewalld*

v. *sudo reboot*

Step 3: Add OpenStack Victoria repository

i. *sudo dnf search centos-release-openstack*

ii. *sudo dnf -y install centos-release-openstack-yoga*

iii. *sudo dnf update -y*

Step 4: Install Packstack and generate answers file

i. *sudo dnf install -y openstack-packstack*

ii. *sudo packstack --os-neutron-ml2-tenant-network-types=vxlan *
*--os-neutron-l2-agent=openvswitch *

*--os-neutron-ml2-type-drivers=vxlan, flat *
*--os-neutron-ml2-mechanism-drivers=openvswitch *
*--keystone-admin-passwd=<admin password> *
*--nova-libvirt-virt-type=kvm *
*--provision-demo=n *
*--cinder-volumes-create=y *
*--os-heat-install=y *
*--os-swift-storage-size=2G *
--gen-answer-file /root/answers.txt

Step 5: Install OpenStack Yoga on CentOS 8 With Packstack

i. *# With answers file you can install Openstack*
sudo packstack --answer-file /root/answers.txt --timeout=3000

ii. *sudo reboot*

iii. *source ~/keystonerc_admin (you can access openstack from terminal)*

Step 6: Accessing the OpenStack Dashboard

http://ip-address-given/dashboard

Credentials will be available at file – /root/keystonerc_admin

Step 7: Configuring Neutron Networking.

$ sudo vi /etc/sysconfig/network-scripts/ifcfg-eno1

DEVICE=eno1

ONBOOT=yes

TYPE=OVSPort

DEVICETYPE=ovs

OVS_BRIDGE=br-ex

$ sudo vi /etc/sysconfig/network-scripts/ifcfg-br-ex

DEVICE=br-ex

BOOTPROTO=none

ONBOOT=yes

TYPE=OVSBridge

DEVICETYPE=ovs

USERCTL=yes

PEERDNS=yes

IPV6INIT=no

IPADDR=172.x.x.x

NETMASK=255.x.x.x

GATEWAY=172.x.x.x

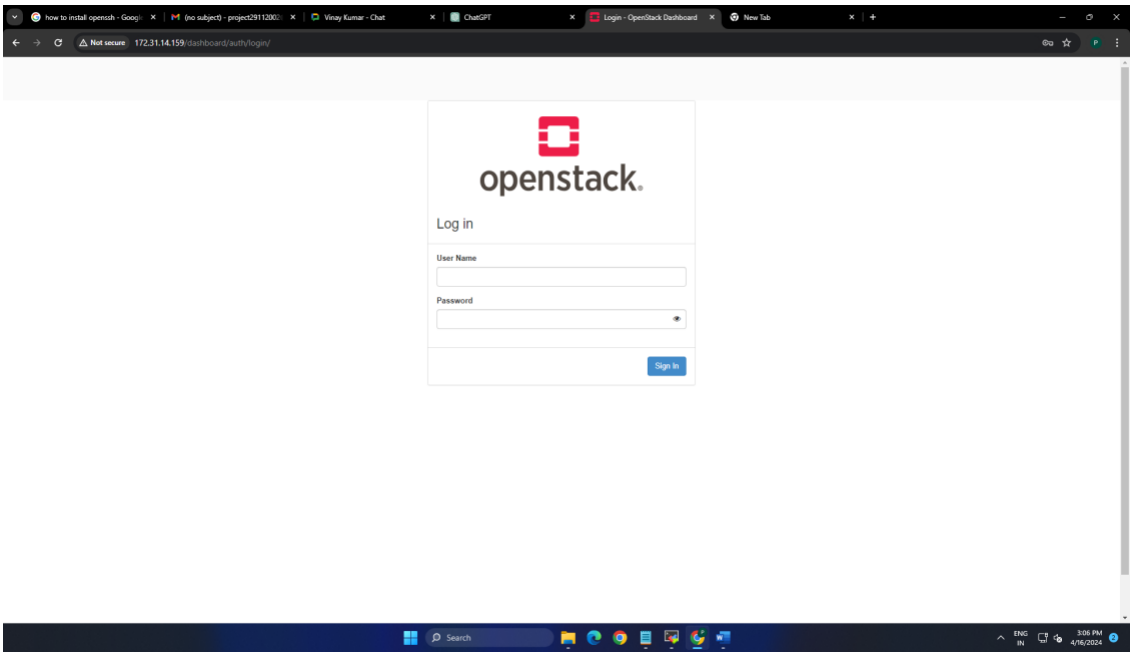
Step 8: Restart Network

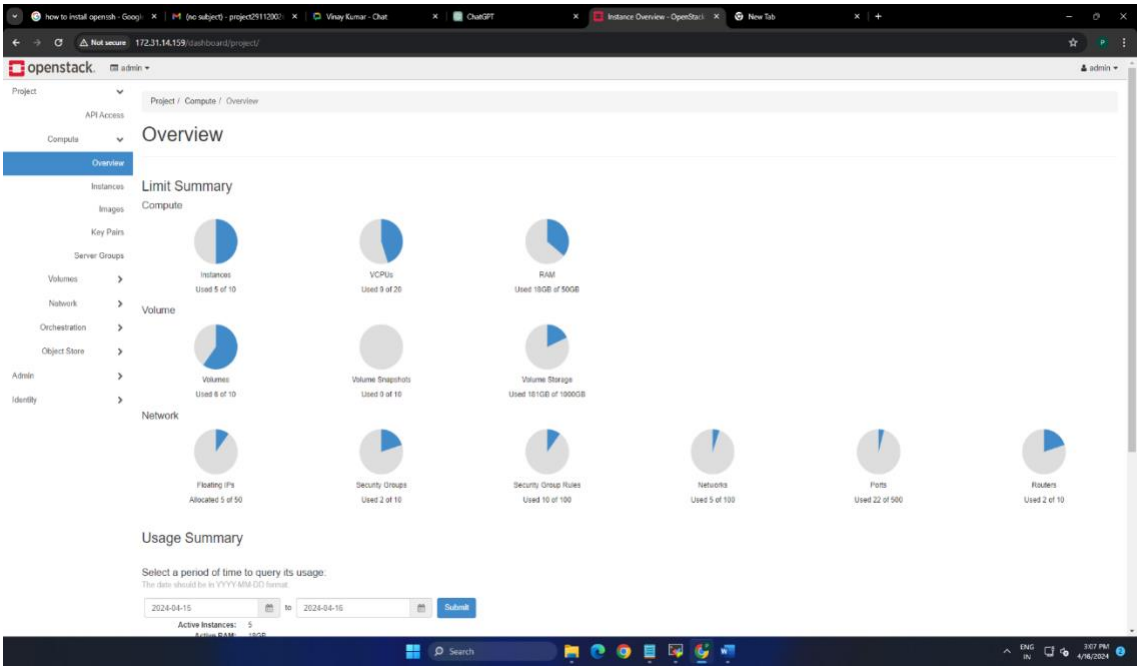
$ sudo systemctl restart network

Chapter 4

SERVICE OUTCOMES

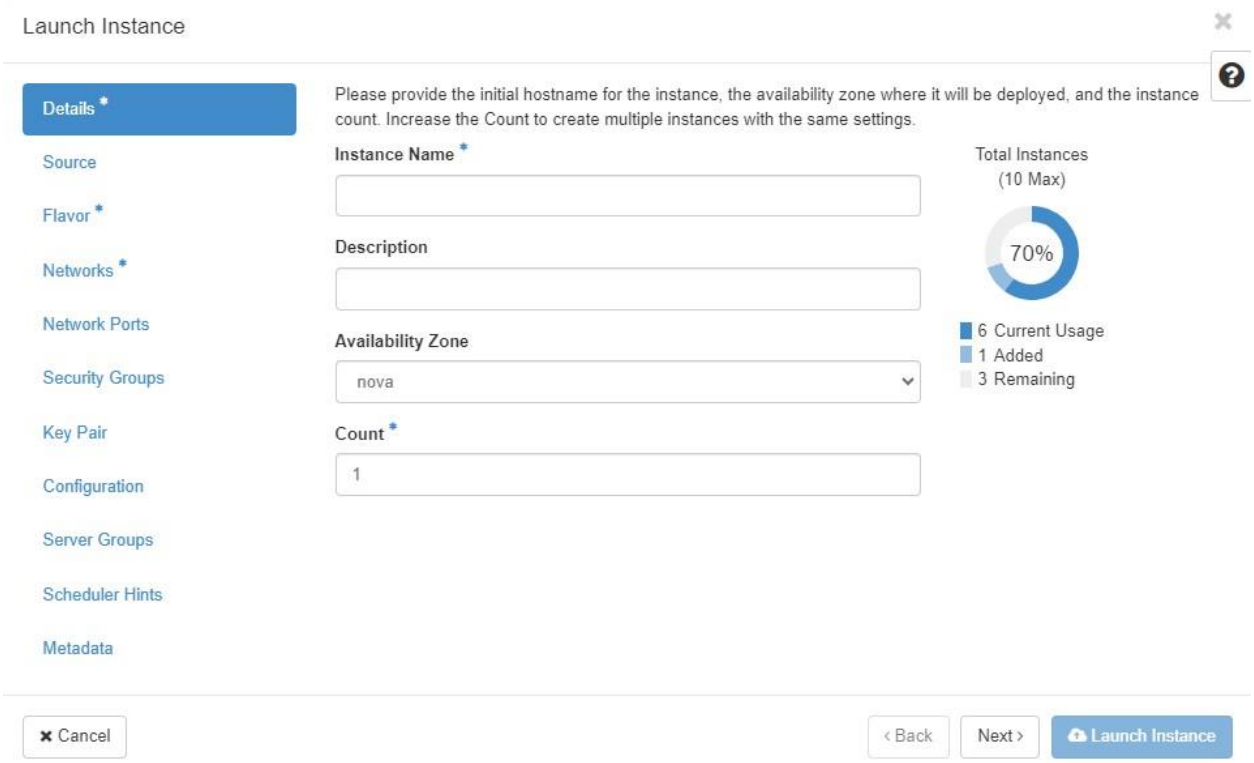
4.1 USER INTERFACE





4.2 CREATE AN INSTANCE

Step 1: Click on Create Instance and Tap Details



Fill the given details like Instance Name and Description is optional

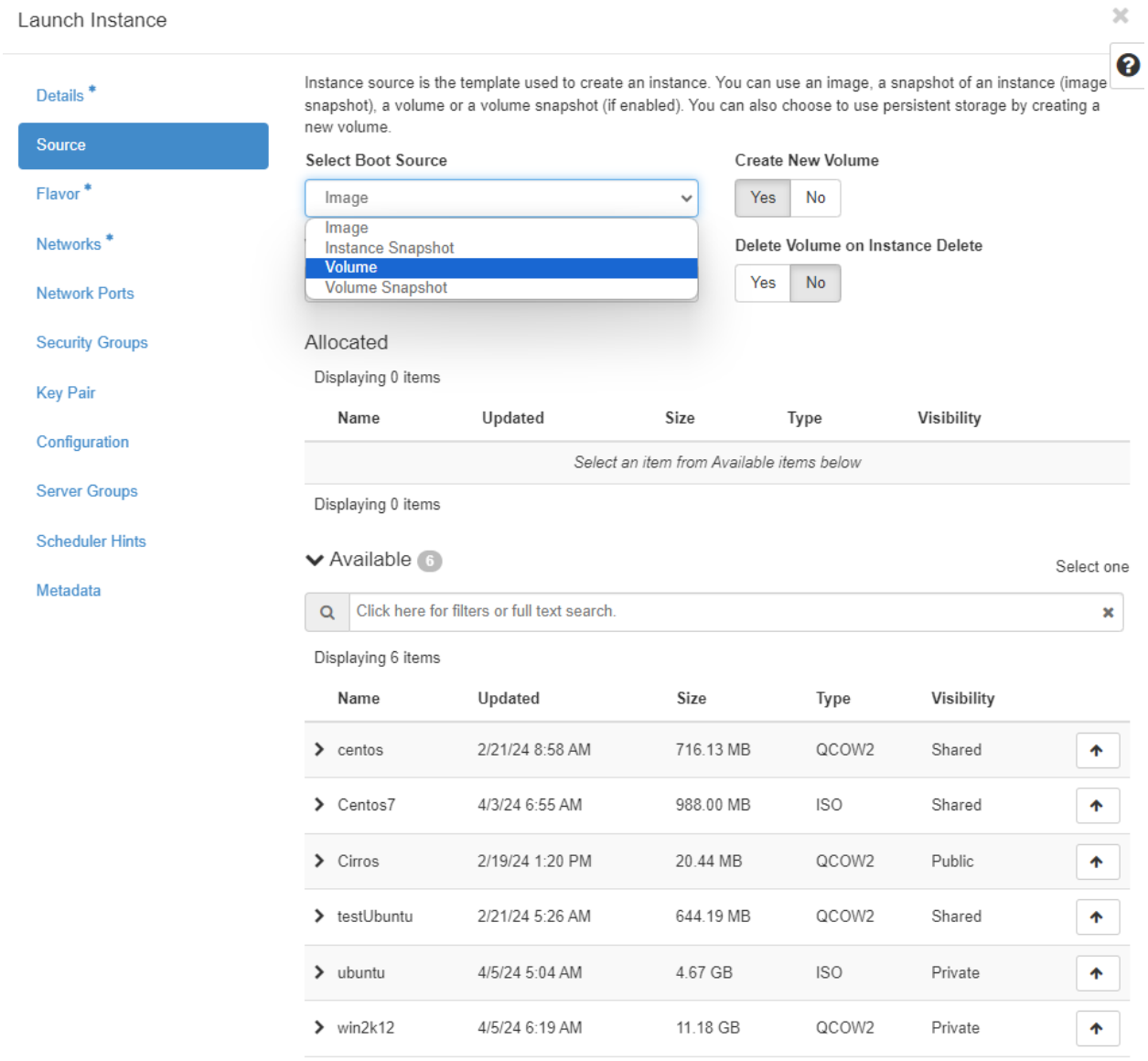
Step 2: Fill the Source

There are four types of Sources 1. Image

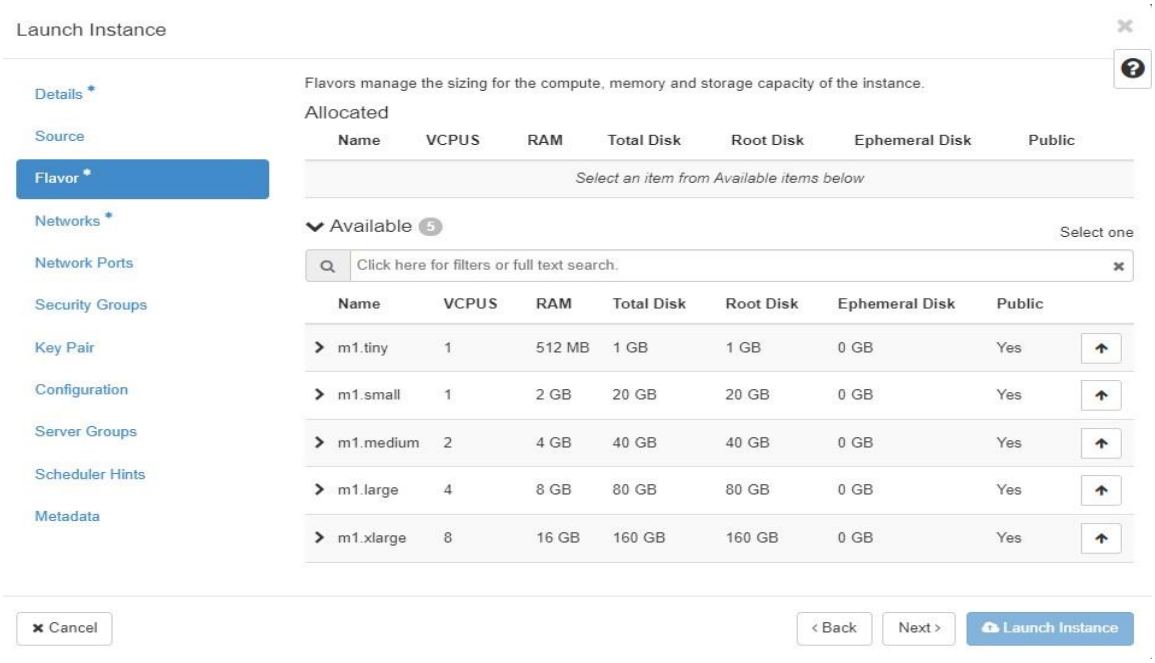
2. Image Snapshot

3. Volume

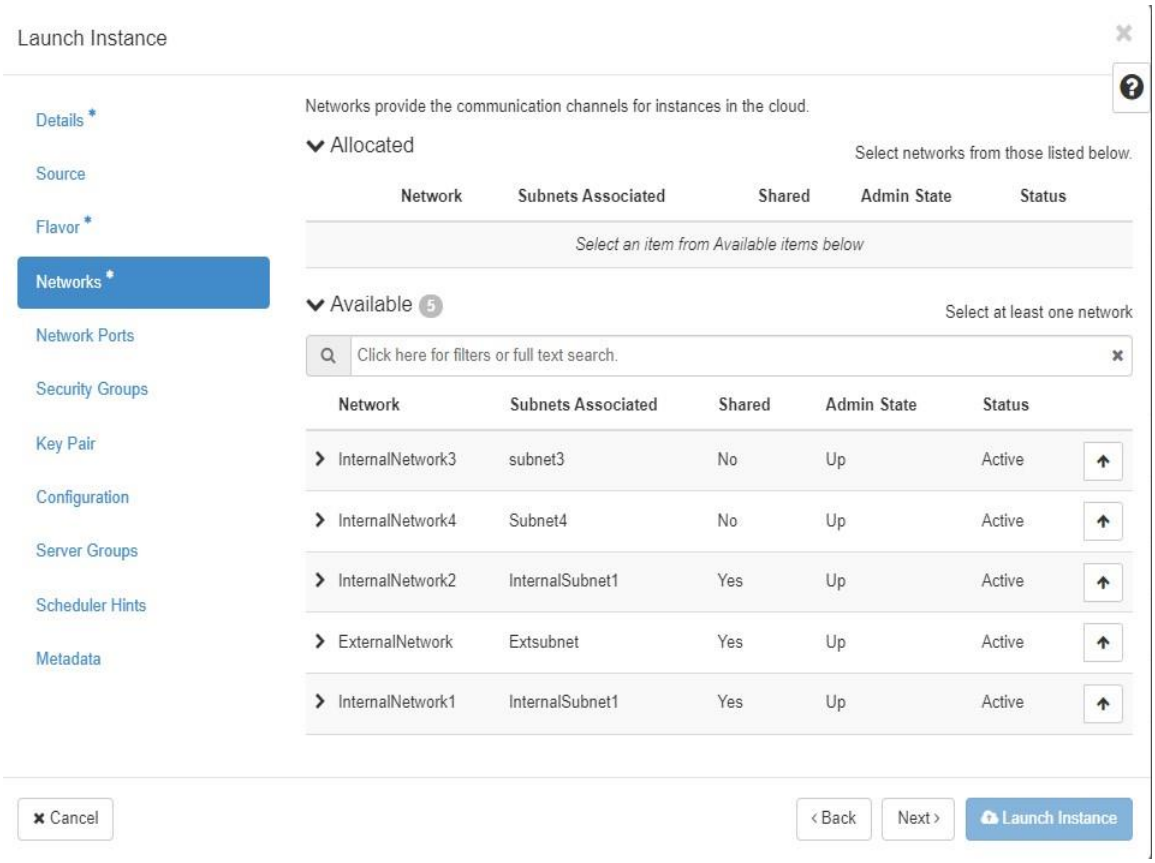
4. Volume Snapshot



Step 3: Select a Flavor

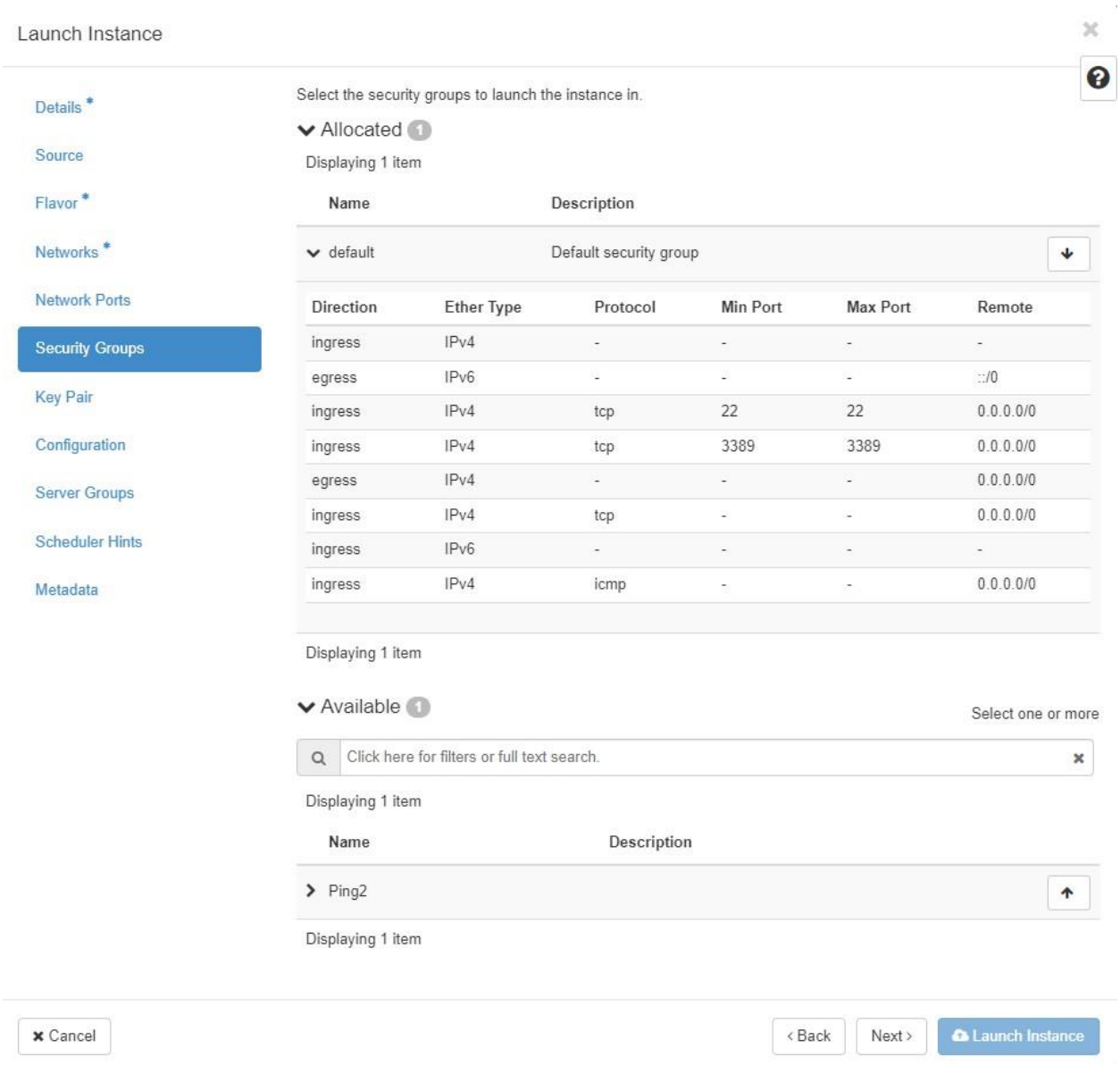


Step 4: Select a Private Network



Step 5: The Security Groups are very Important for ingress and egress of RDP, SSH, ICMP request.

Select the default group provided along with installation pack.



Step 6: Now click on Launch Instance. Your instance is ready whenever it shows *spawning.*

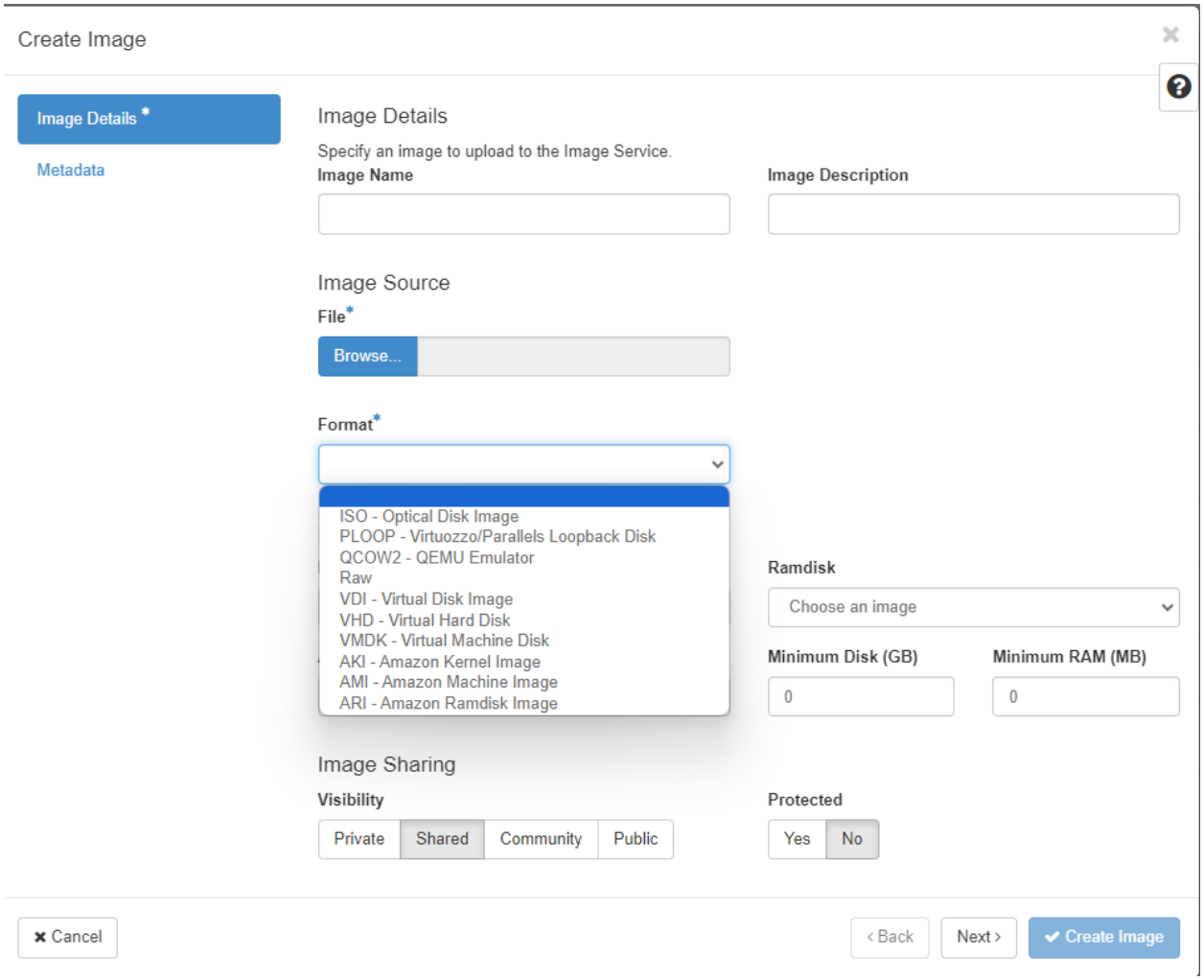
4.3 CREATE AN IMAGE

Method 1: Using User Interface (If the size is less than 1 GB)

Step 1: Click on Create Image and Fill the Required Details

Step 2: Upload an Image and Select Image Format

Step 3: This Service requires the Knowledge of different kinds of image formats.



Step 4: Now Click on Create Image.

Method 2: Using Openstack Command Line Interface (If image size is more than 1 GB)

Step 1: Head to terminal in Centos 8 Stream

Step 2: Type *source keystonerc_admin*

Step 3: Type the following command

Command:

openstack image create --disk-format iso --private --file /Windows_22H2_64bit.iso image_name --fit-width

Replaces

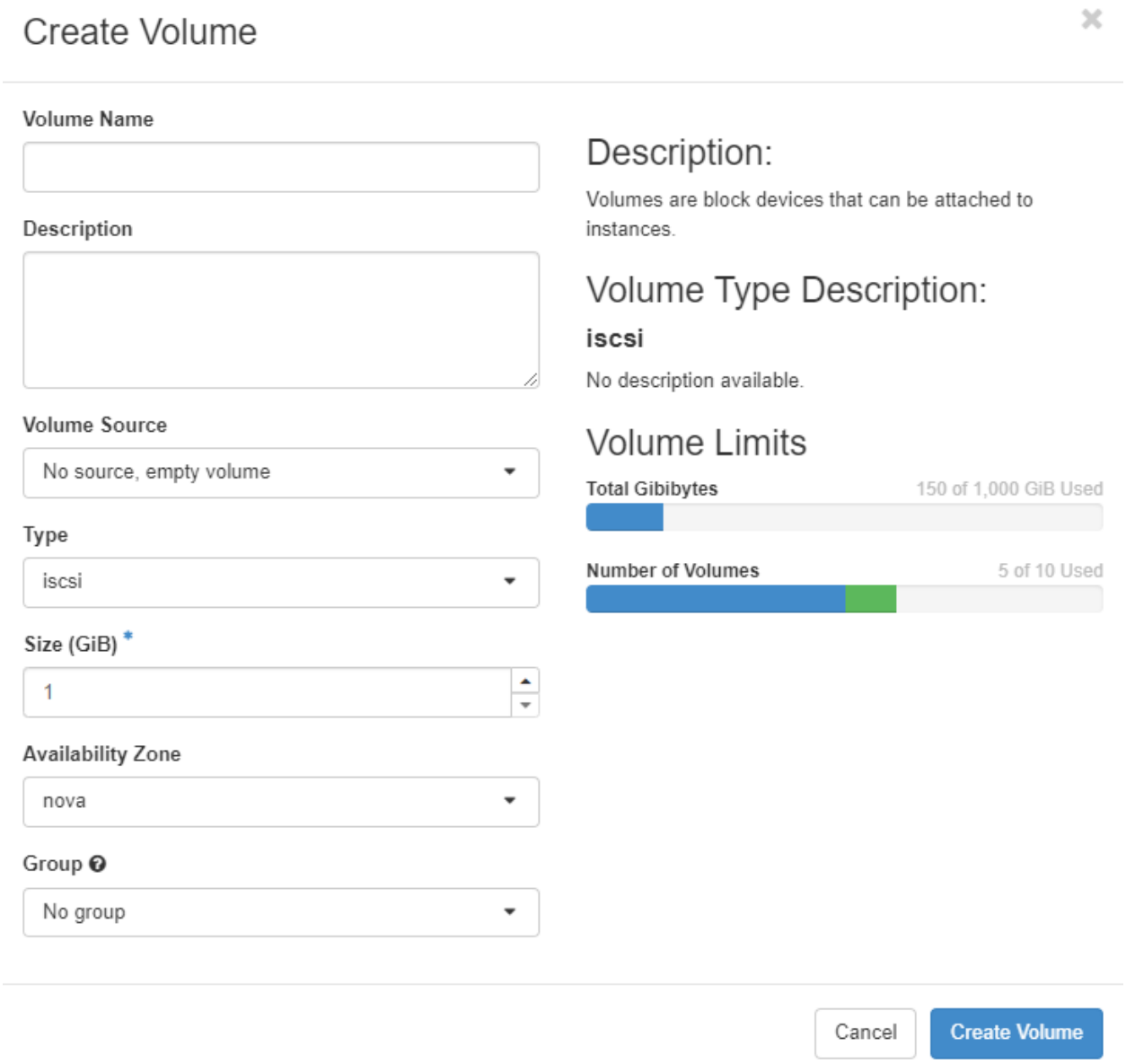
1. *iso with your image format (i.e., .qcow2, .img, raw, .vmdk)*
2. *Windows_22H2_64bit.iso with the name of your uploading image.*
3. *image_name with your own name.*

Step 4: Type openstack image list, now check whether it is created or not.

```
Last login: Mon Apr 15 11:03:54 2024
[root@openstack ~]# source keystonerc_admin
[root@openstack ~(keystone_admin)]# openstack image list
+--------------------------------------+------------+--------+
| ID                                   | Name       | Status |
+--------------------------------------+------------+--------+
| 285ea4cd-2b41-4393-9403-c52273215403 | Centos8    | active |
| 82023b11-aa87-4d14-b1e3-fb4529cd233c | Cirros     | active |
| e5b8314a-b62d-41d8-ab8c-cc2817c19903 | centos     | active |
| e73918a1-c9c0-448e-8d68-e4a509248cfa | testUbuntu | active |
| 45a523a4-9ada-4a39-8514-99ece0b0d93e | ubuntu     | active |
| 9790e459-5304-44d6-b679-d1d8d759901a | win2k12    | active |
+--------------------------------------+------------+--------+
[root@openstack ~(keystone_admin)]#
```

4.4 CREATE A VOLUME

Step 1: Fill the required details

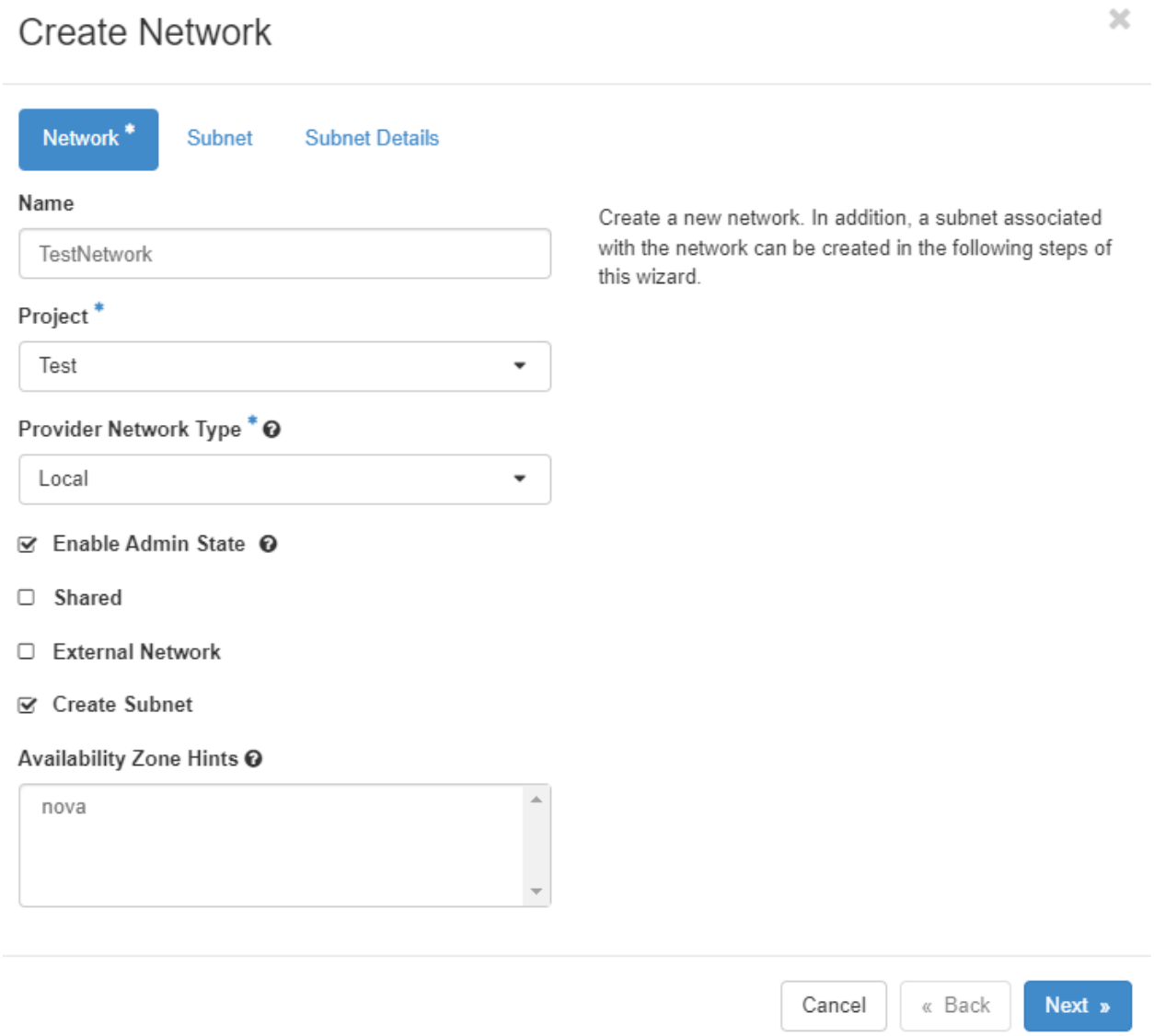


Step 2: Click on Create Volume

4.5 CREATE A NETWORK

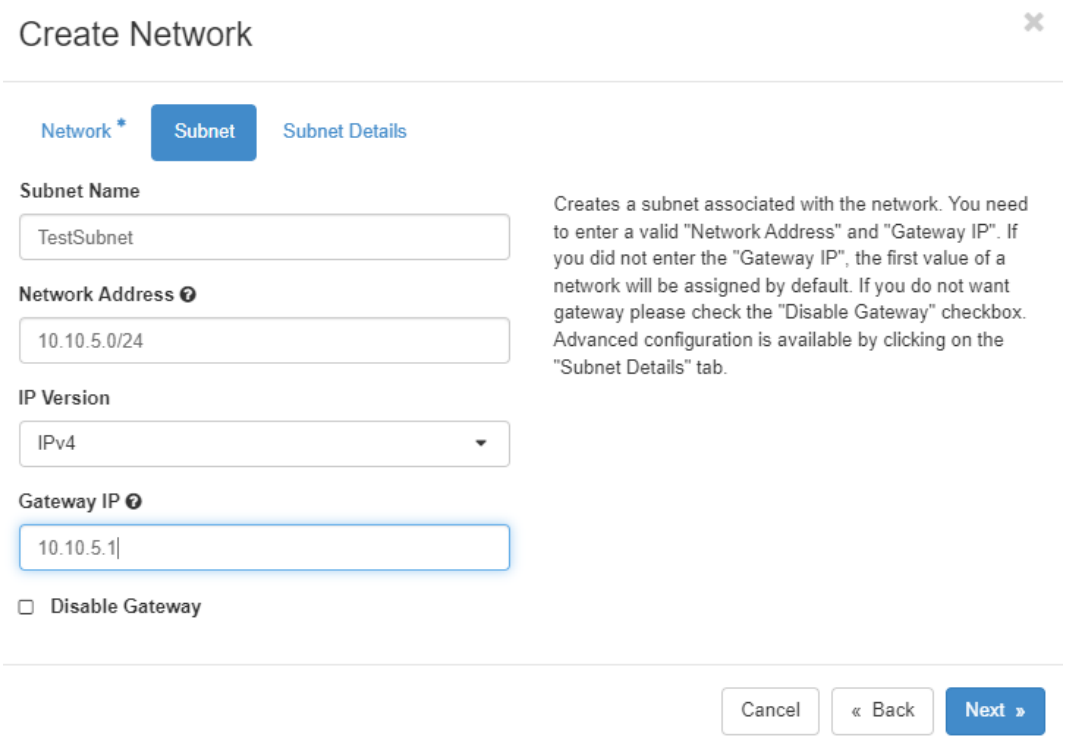
A. NETWORK

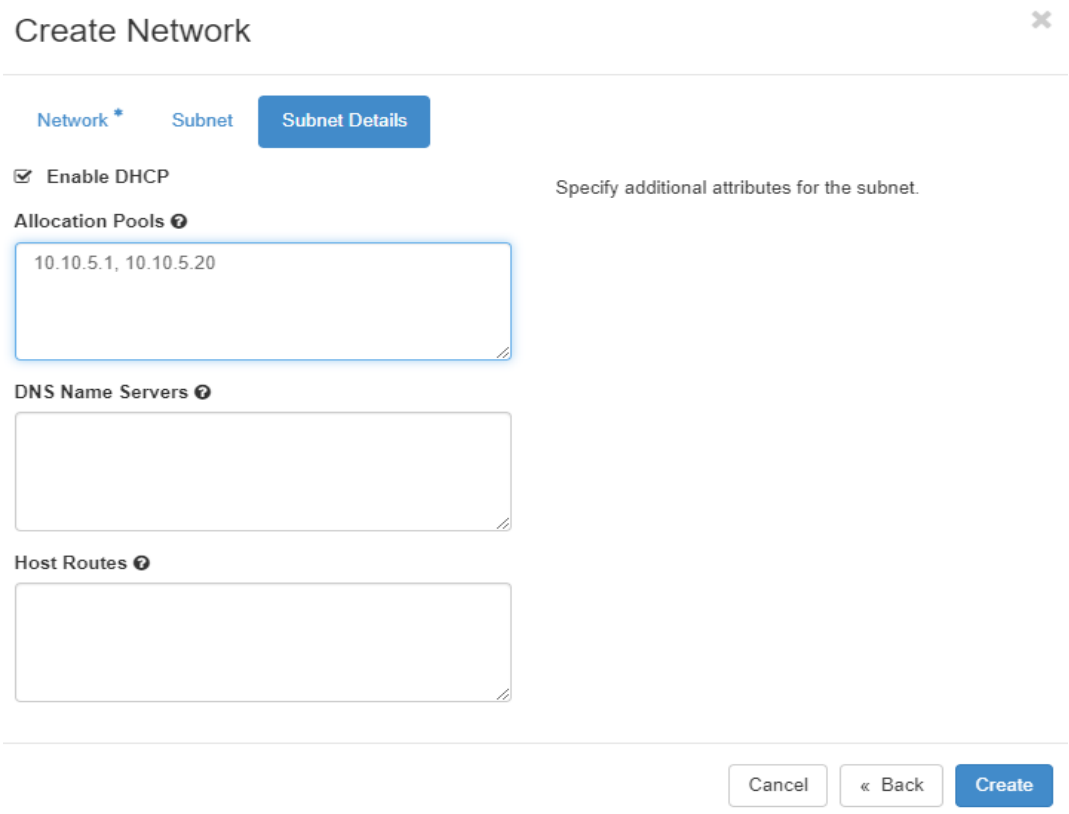
Step 1: Navigate to Network Tab and Click on +Create Network



Step 2: Click Next and Give the details of Subnet such as name and Network Address with Subnet mask representation as IP-Address/ 8, 16, 24

Give Gateway if requires. You can use DHCP as well.





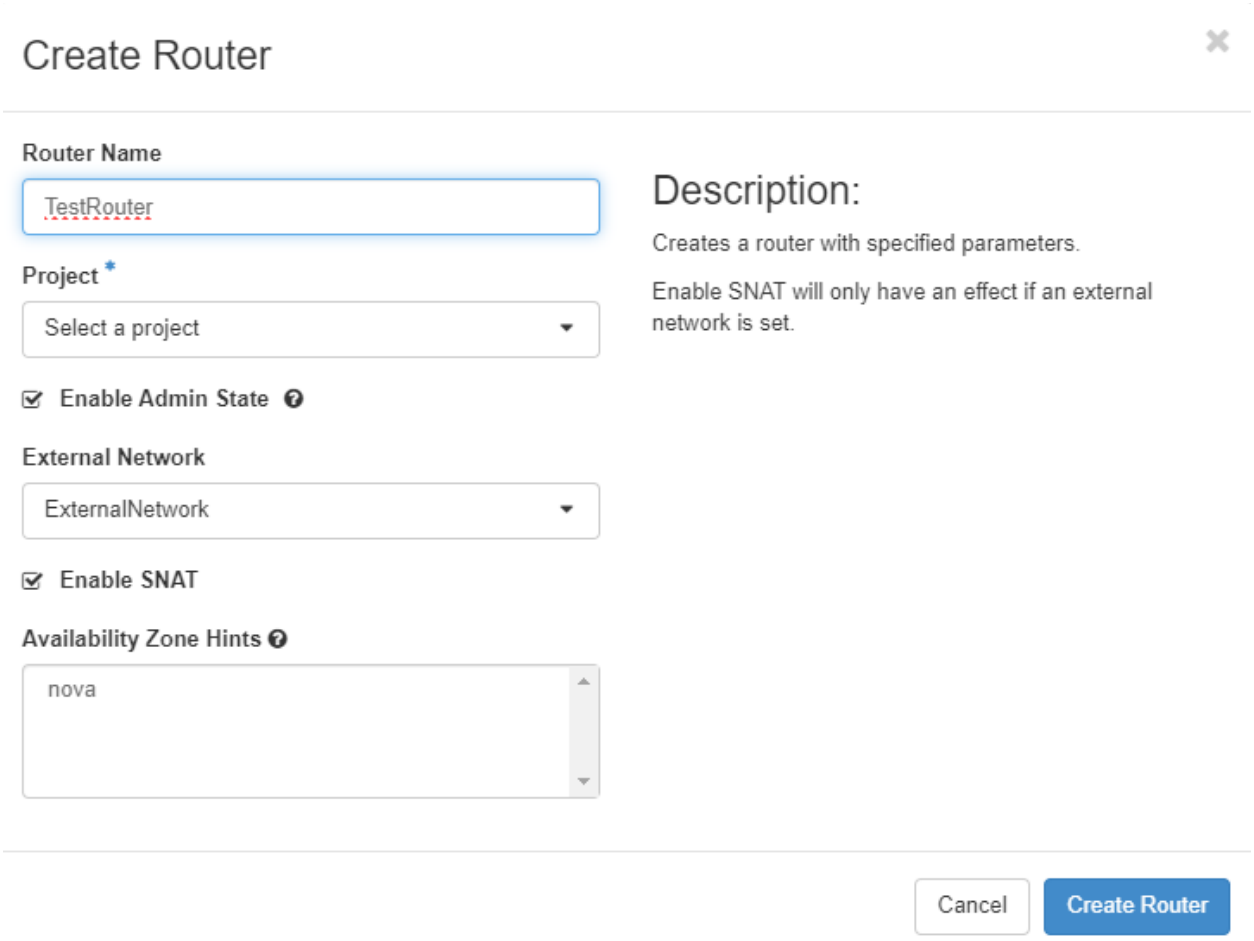
Step 3: Click on Create

B. CREATE A ROUTER

Step 1: This requires an External Network (Public Network).

Step 2: Click on Routers in Network Tab and Click +Create Router

Step 3: Select External Network



Step 4: Click on Create Router Button

Router is required to connect two networks such that they can communicate each other.

4.6 FLOATING IP

In short, a floating IP is a flexible IP address that can be quickly moved or reassigned between servers. It's commonly used in cloud computing and virtualized environments to provide high availability and fault tolerance for services by redirecting traffic in case of server failures.

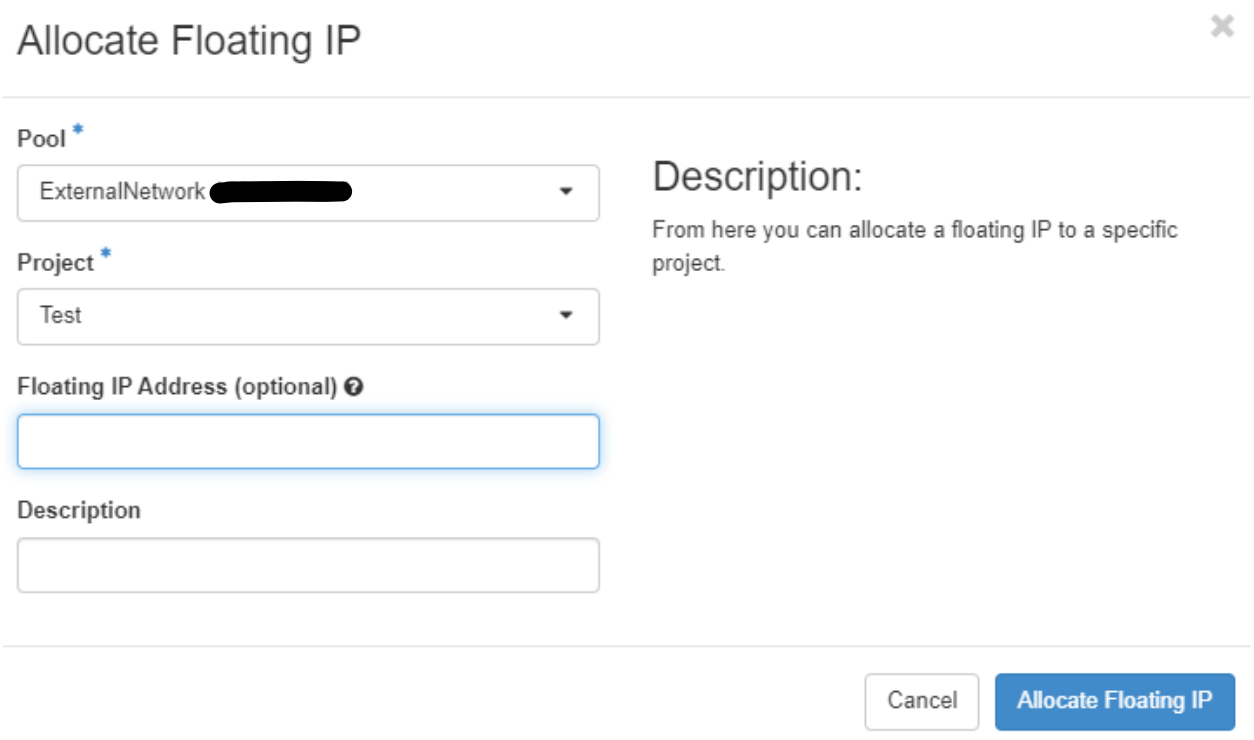
Note: In this project floating ip is used for accessing public network.

Step: 1. An External Network is used.

Step: 2. Pool of unused IP's Required from DHCP Allocation Pool.

Step: 3. Navigate to Network Tab and Click Floating IPs.

Step: 4. Click Allocation IP to Project button. Fill the details

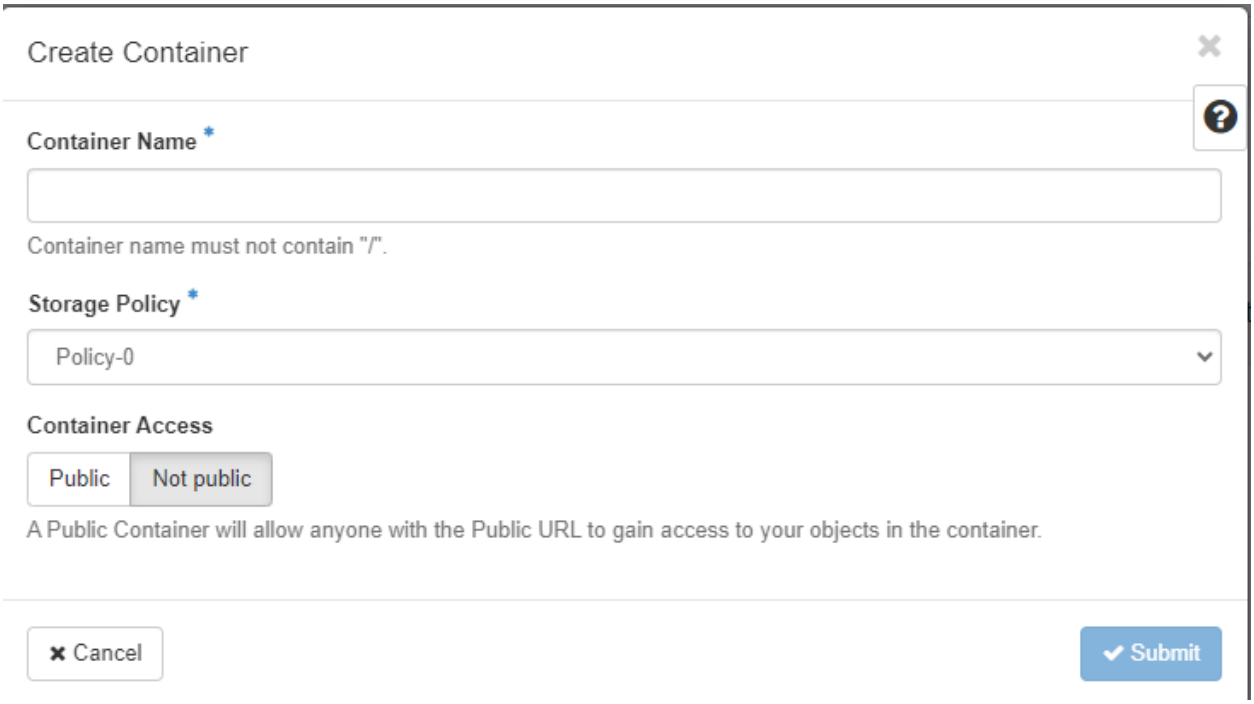


Step: 5. Click Allocate Floating IP.

4.7 CREATE CONTAINERS

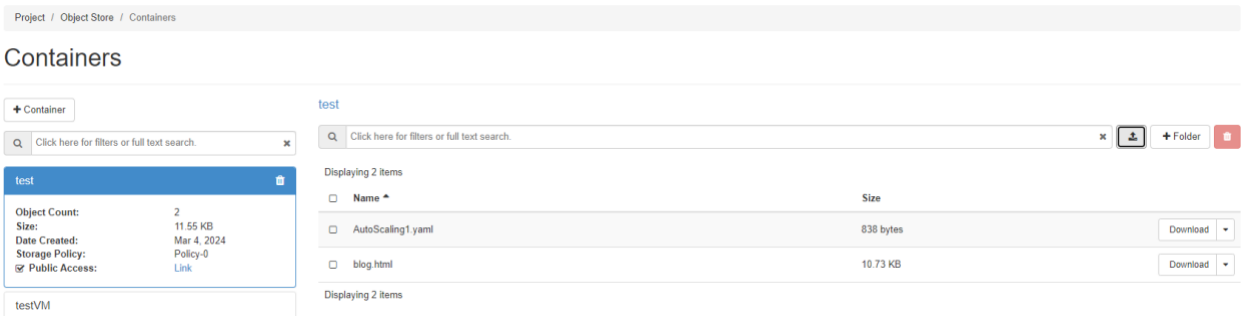
Step 1: Head to Object Store and select Containers. Click +Container button to create a Storage container.

Step 2: Click +Folder to upload your folder directly into the container or click upload button to store your files one by one.



Step 3: Click the Public Access check button and copy the Link.

Step 4: Paste the link in the new browser tab to access the file stored in the Container.

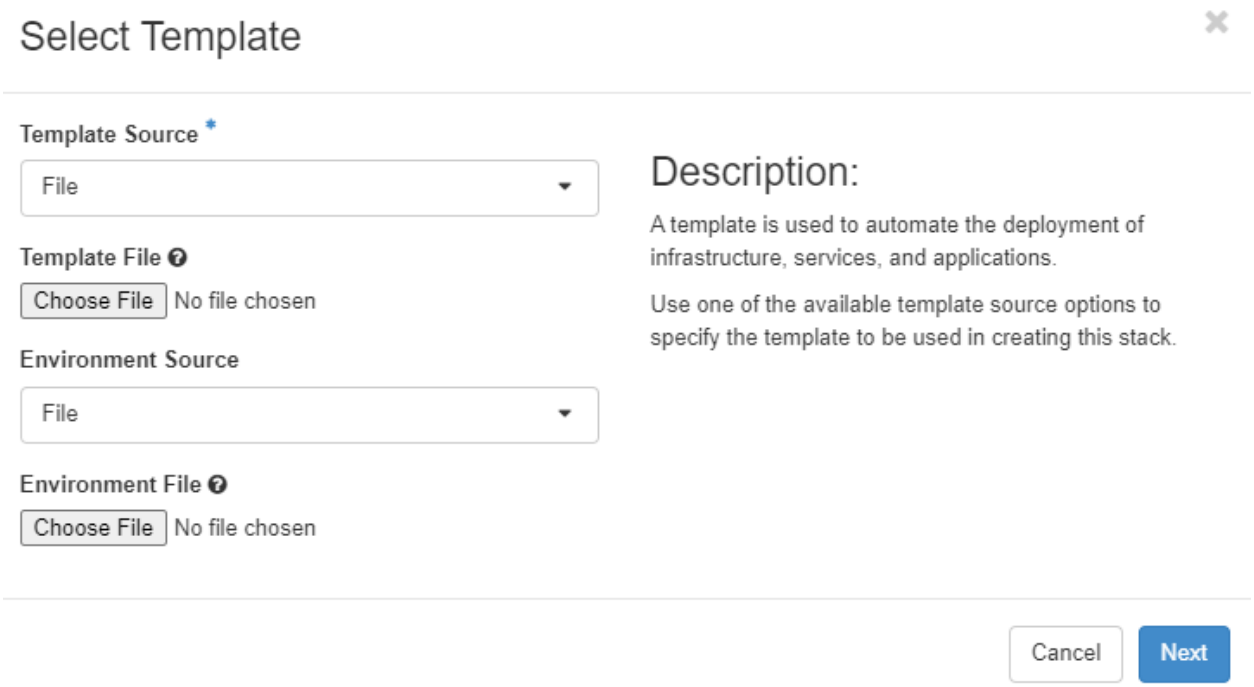


4.8 STACKS

In OpenStack, "stacks" refer to the orchestration service provided by the OpenStack Heat project. Heat allows users to define and manage infrastructure resources, such as virtual machines, networks, and storage, using templates written in YAML or JSON format. These templates describe the desired state of the infrastructure, including dependencies between resources.

Step 1: Head to Orchestration and select Stacks

Step 2: Click on +Launch Stack button and fill the template file



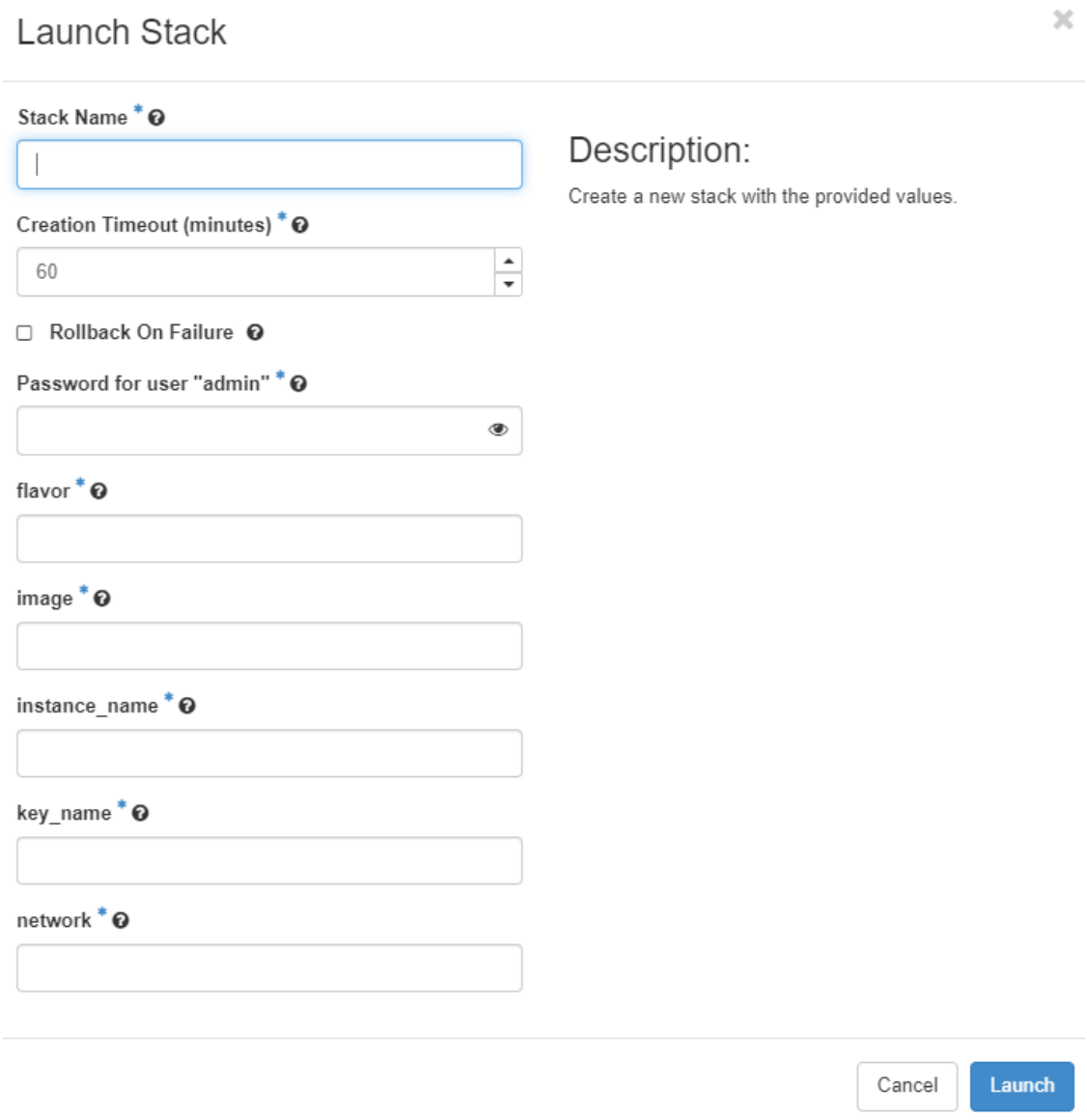
Step 3: Knowledge of YAML and JSON is required, then we have to create template and environment file.

Here we see how to create an instance using stacks.

Sample Template.yaml

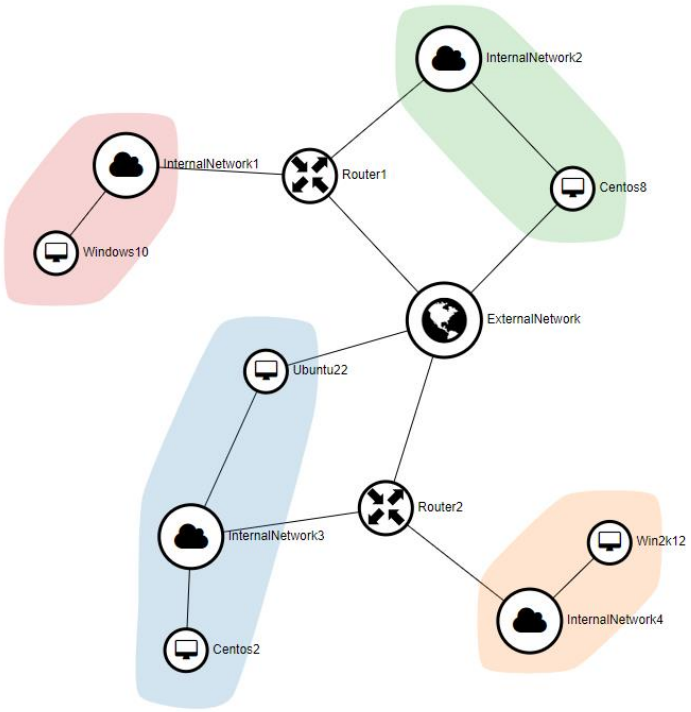
```
heat_template_version: 2018-05-31
parameters:
  flavor:
    type: string
    description: Flavor of the instance
  image:
    type: string
    description: Image ID or name for the instance
  network:
    type: string
    description: Network ID or name for the instance
  key_name:
    type: string
    description: Key pair name for SSH access to the instance
  instance_name:
    type: string
    description: Name of the instance
resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      name: { get_param: instance_name }
      flavor: { get_param: flavor }
      image: { get_param: image }
      key_name: { get_param: key_name }
      networks:
        - network: { get_param: network }
```

Step 4: Now upload the yaml file and fill the requirements, your instance is created.



4.9 GRAPH AND TOPOLOGY VIEW OF NETWORK

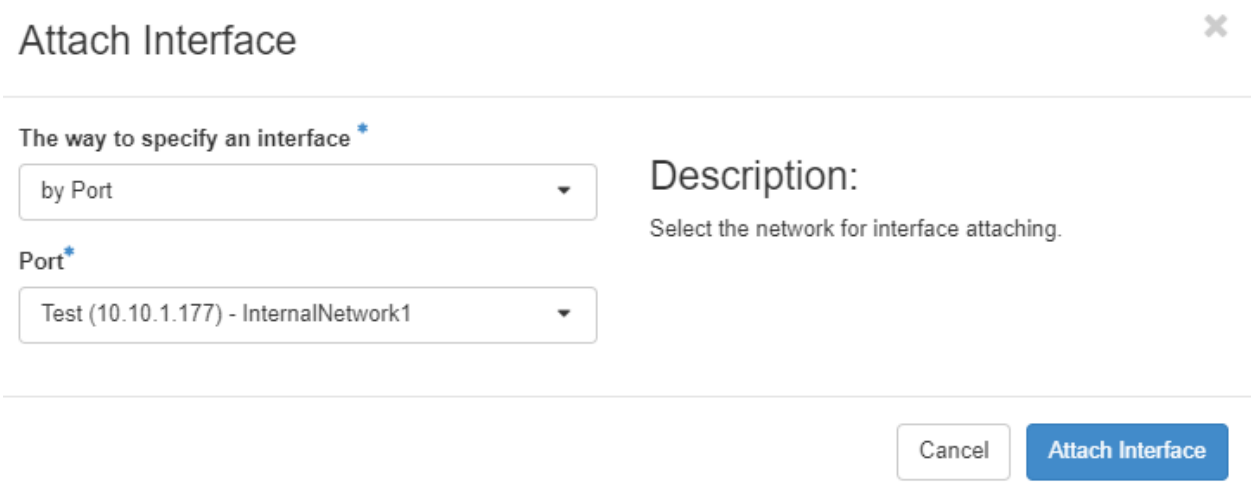
a.



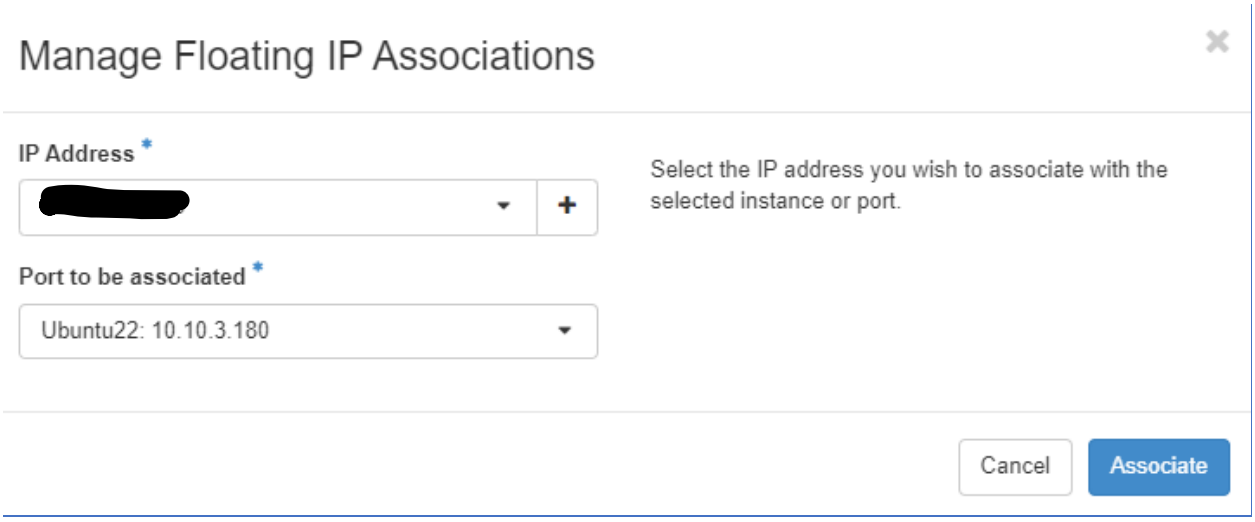
b.

4.10 ADD – ONS FOR CREATED INSTANCES

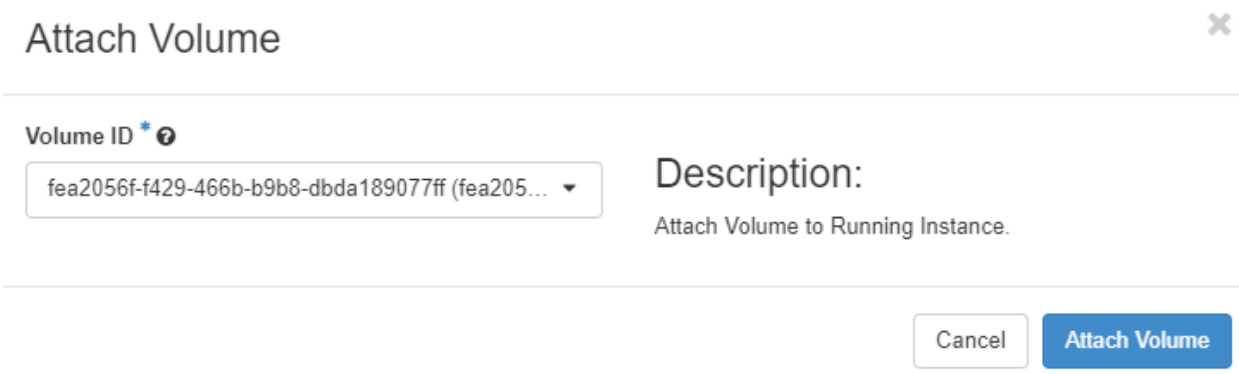
a. Attach Interface.



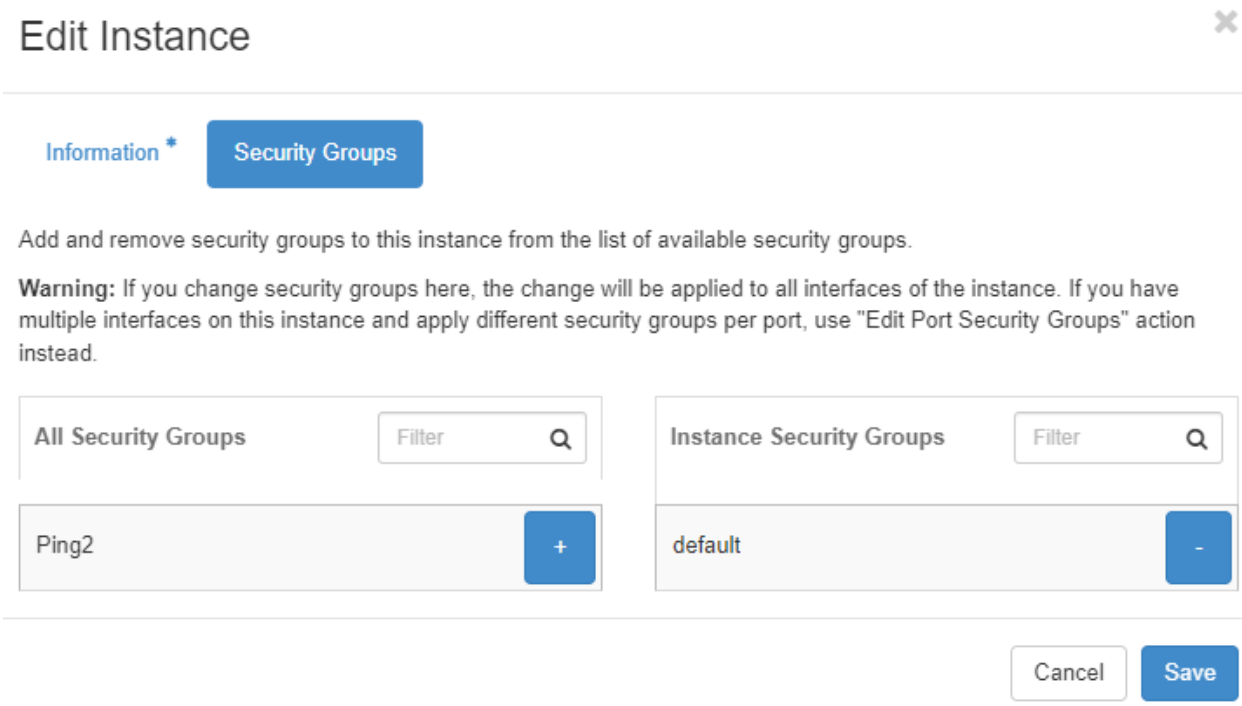
b. Associate Floating IP



c. Attach Volume



d. Edit Security Groups

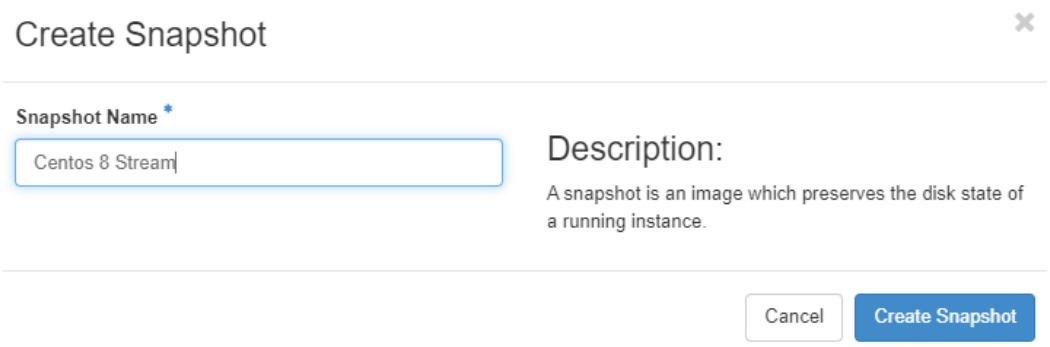


e. Create Instance Snapshot:

An instance snapshot in OpenStack is a feature that allows users to capture the current state of a virtual machine (VM) or instance at a specific point in time. This snapshot includes the VM's disk, memory, and configuration settings, essentially creating a backup or image of the instance.

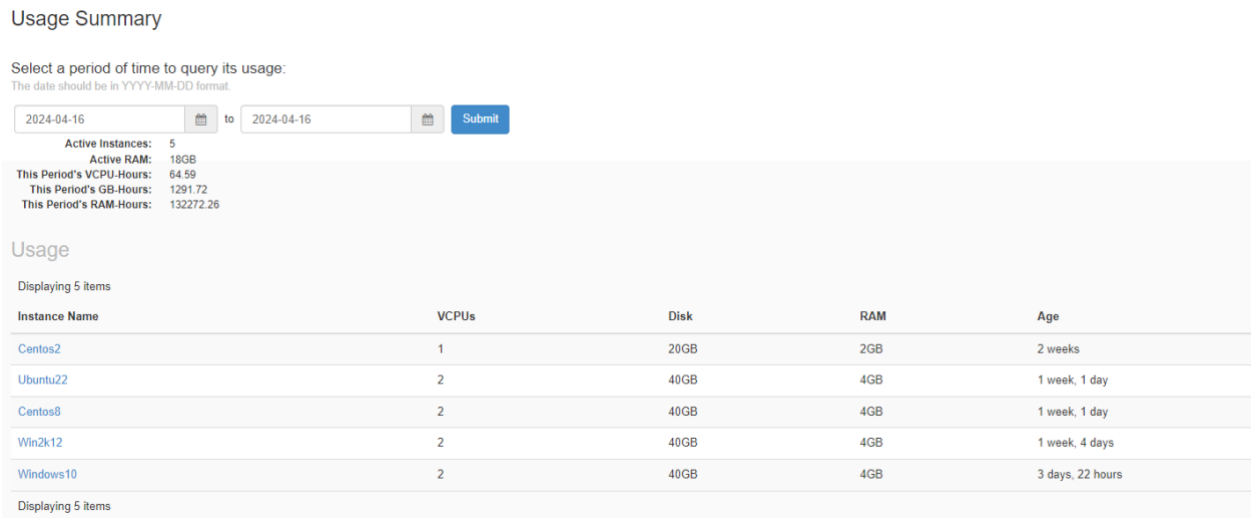
Snapshotting is a valuable tool for data protection, disaster recovery, and system maintenance. Users can create snapshots of their instances before making significant changes or updates, providing a point of rollback in case of errors or failures. Additionally, snapshots can be used to clone instances, replicate environments, or migrate workloads between different cloud regions or providers.

By enabling users to easily create and manage instance snapshots through the OpenStack dashboard or API, the platform promotes flexibility, efficiency, and reliability in managing cloud resources. This feature enhances data security and accessibility while facilitating agile and resilient cloud operations.



4.11 PERFORMANCE AND ACCESS CONTROL

a. Monitoring Resources



b. Usage Table

Active Instances:	5			
Total VCPU Usage (Hours):	63.87			
Total Active RAM (MB):	18432			
Total Memory Usage (Hours):	130808			
Total Disk Size (GB):	180			
Total Disk Usage (Hours):	1277.43			
Project Name	**VCPUs**	**RAM (MB)**	**Disk (GB)**	**Usage (Hours)**
admin	9	18432	180	63.87

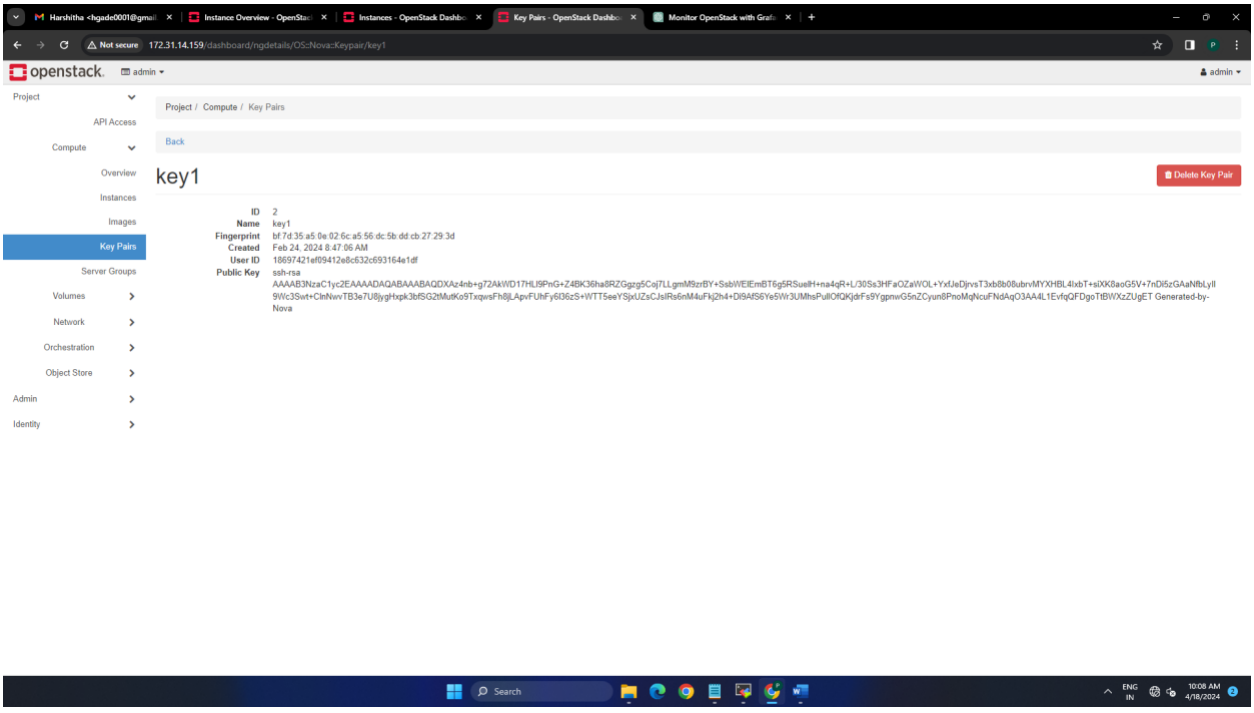
4.12 SECURITY ACCESS AND CONTROL

Key pairs are typically used for SSH (Secure Shell) access to instances within OpenStack or other cloud platforms. They consist of a public key and a private key. The public key is stored on the server (the instance in this case), while the private key is kept securely by the user. When connecting to the instance via SSH, the user presents their private key, and if it matches the public key stored on the server, access is granted.

a. Authentication: Key pairs provide a secure method of authenticating users who are attempting to access cloud instances. Without the corresponding private key, unauthorized users cannot gain access.

b. Authorization: By controlling access to the private key, administrators can control who has permission to access the instances. This helps enforce authorization policies and restrict access to only authorized users.

c. Security: Key pairs enhance security by eliminating the need to transmit passwords over the network when accessing instances via SSH. This reduces the risk of password interception and unauthorized access.



4.13 SERVICE MANAGEMENT

Root Cause Analysis: In the face of OpenStack service disruptions, conducting a root cause analysis is imperative. This investigative process delves deep into the infrastructure, examining every layer and component to identify the underlying reason for the malfunction. By meticulously dissecting system logs, monitoring metrics, and configuration settings, administrators can pinpoint the exact point of failure, be it a misconfiguration, resource bottleneck, software bug, or hardware issue. This methodical approach enables swift resolution, minimizing downtime and ensuring the reliability and stability of the OpenStack environment. Through proactive root cause analysis, organizations can bolster their troubleshooting capabilities and fortify their infrastructure against future disruptions, fostering a resilient and high-performing cloud ecosystem.

Whenever any service is not working, please see all the services list below

Try restarting the particular service and check the status of service

```
Last login: Thu Apr 18 09:59:37 2024 from 172.31.2.165
[root@openstack ~]# systemctl status openstack-
openstack-aodh-api.service                    openstack-nova-metadata-api.service
openstack-aodh-evaluator.service              openstack-nova-novncproxy.service
openstack-aodh-listener.service               openstack-nova-os-compute-api.service
openstack-aodh-notifier.service               openstack-nova-scheduler.service
openstack-ceilometer-ipmi.service             openstack-swift-account-auditor.service
openstack-ceilometer-notification.service     openstack-swift-account-reaper.service
openstack-ceilometer-polling.service          openstack-swift-account-replicator.service
openstack-cinder-api.service                  openstack-swift-account.service
openstack-cinder-backup.service               openstack-swift-container-auditor.service
openstack-cinder-scheduler.service            openstack-swift-container-reconciler.service
openstack-cinder-volume.service               openstack-swift-container-replicator.service
openstack-glance-api.service                  openstack-swift-container.service
openstack-glance-scrubber.service             openstack-swift-container-sharder.service
openstack-gnocchi-api.service                 openstack-swift-container-sync.service
openstack-gnocchi-metricd.service             openstack-swift-container-updater.service
openstack-gnocchi-statsd.service              openstack-swift-object-auditor.service
openstack-heat-api-cfn.service                openstack-swift-object-expirer.service
openstack-heat-api.service                    openstack-swift-object-reconstructor.service
openstack-heat-engine.service                 openstack-swift-object-replicator.service
openstack-nova-api.service                    openstack-swift-object.service
openstack-nova-compute.service                openstack-swift-object-updater.service
openstack-nova-conductor.service              openstack-swift-proxy.service
```

Chapter 5

RESULTS AND DISCUSSIONS

5.1 MANAGING INSTANCES

To access instances in OpenStack using SSH (Secure Shell), you typically follow these steps:

a. **Prepare Your Environment:**

   Ensure that you have an OpenStack user account with appropriate permissions to launch and access instances. Have the necessary SSH key pair ready. If you don't have one, you can generate it using tools like `ssh-keygen`.

b. **Launch an Instance:**

   Log in to the OpenStack dashboard (Horizon) or use the OpenStack CLI commands to launch an instance. During instance creation, you'll specify the SSH key pair to be injected into the instance for authentication.

c. **Associate a Floating IP (Optional):**

   If you want your instance to have a publicly accessible IP address, associate a floating IP to it. This step is optional but useful for external access.

d. **Retrieve Instance Details:**

   Once the instance is launched and running, retrieve its details such as the IP address and SSH key pair used.

e. **SSH Access:**

   Open a terminal or SSH client on your local machine.

   Use the `ssh` command to connect to the instance. The command syntax typically looks like this:

   ssh -i /path/to/your/private/key username@instance-ip-address

   Replace `/path/to/your/private/key` with the actual path to your private SSH key.

Replace `username` with the appropriate username for the instance (e.g., `ubuntu`, `centos`, `root`, etc.).
Replace `instance-ip-address` with the IP address of your OpenStack instance (either the floating IP if associated or the internal IP if not).

f. Authenticate and Access:

When prompted, authenticate using the SSH key pair. If everything is set up correctly, you should now have a secure shell connection to your OpenStack instance.

Security Groups: Ensure that the security group associated with your instance allows SSH traffic (port 22) from your IP address or a specific range of IP addresses.

Key Pair Management: Store your private SSH key securely and avoid sharing it publicly. If you lose your private key, you may lose access to the instance unless you have other access mechanisms enabled.

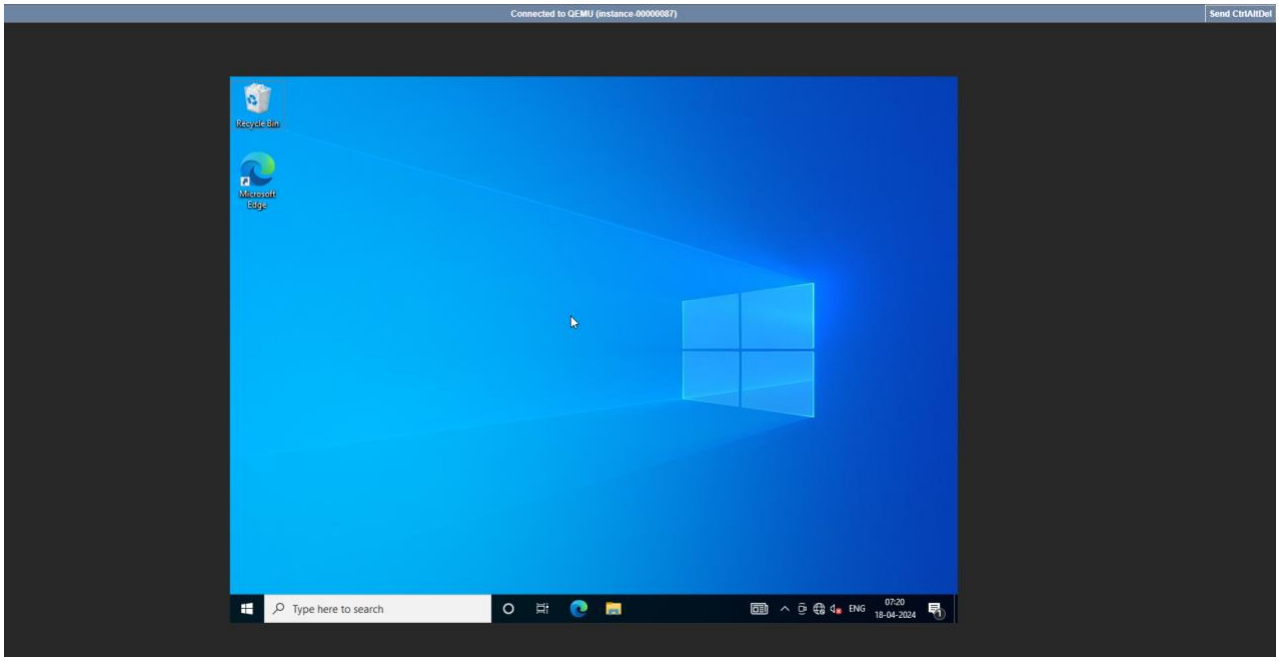
SSH Client Configuration: Depending on your SSH client and operating system, you may need to adjust configurations such as the SSH key file permissions (`chmod 600`) or SSH client settings.

5.2 RESULTS

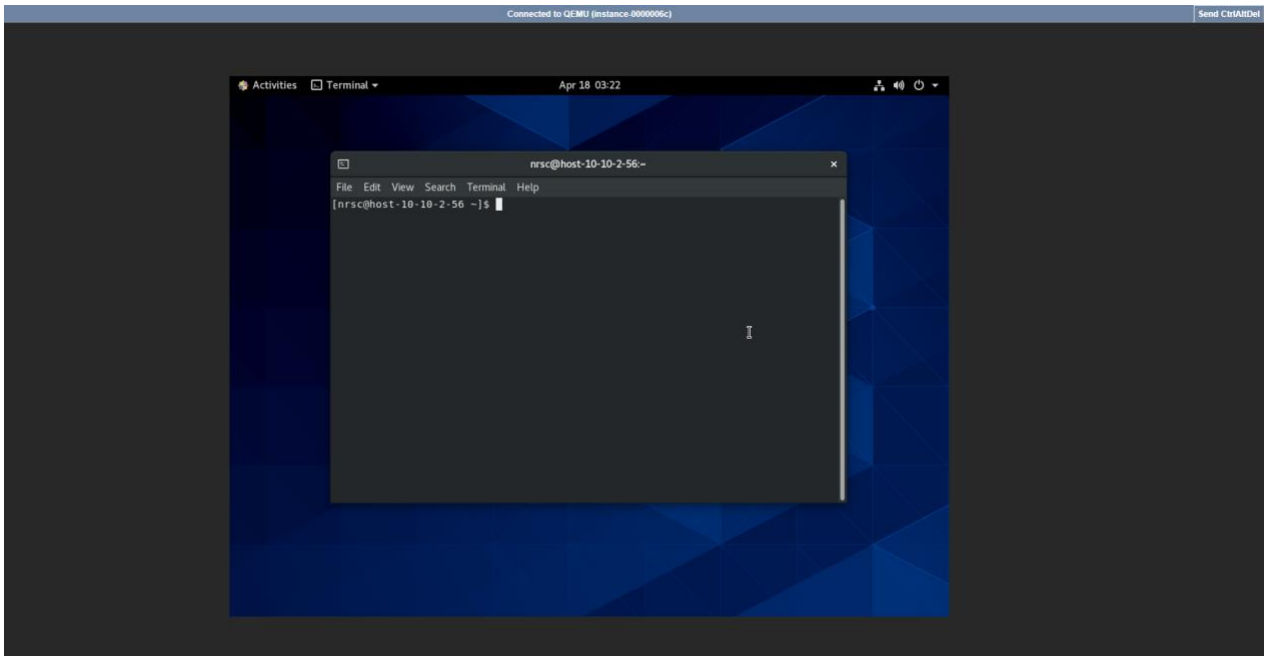
The cloud computing platform was built with the capability of provisioning compute instances by allocating storage resources, and managing networking configurations. A user-friendly web interface is built which allows admins to easily provision, manage, and increase cloud resources according to the user requirements. Enhanced security measures ensuring the confidentiality, integrity, and availability of data and resources are implemented which are within the OpenStack cloud. Monitoring tools and optimization strategies are enabled and available for proactive management of performance and resource utilization.

The following screenshots shows the instances created through OpenStack Interface.

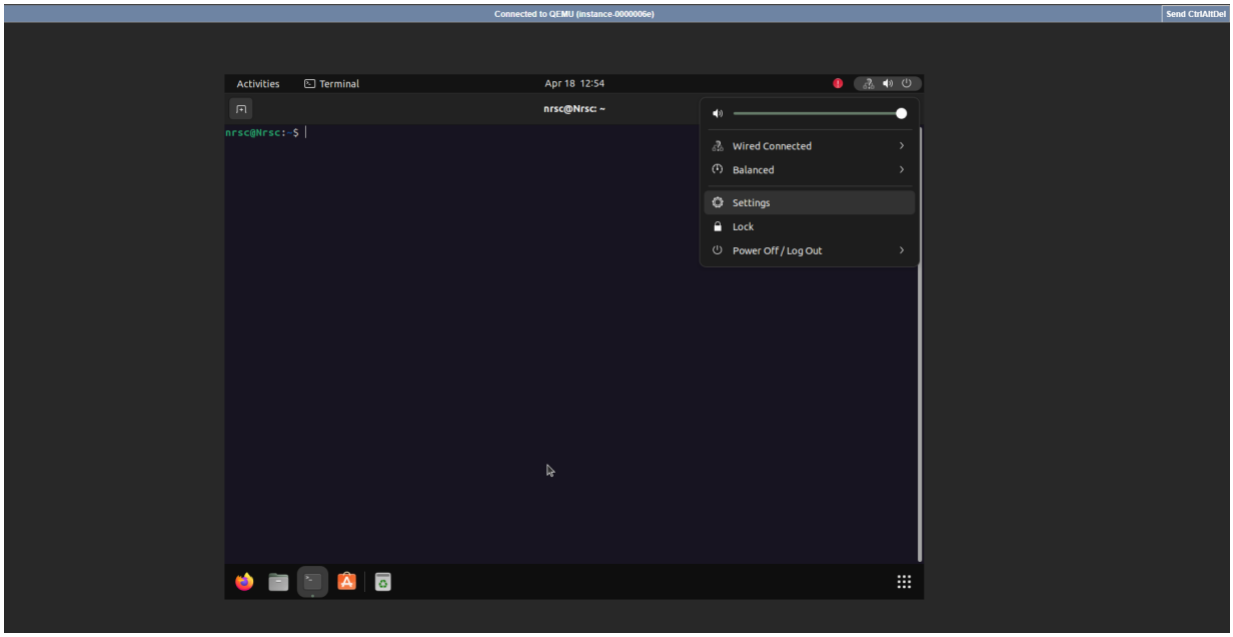
a. Windows instance in openstack



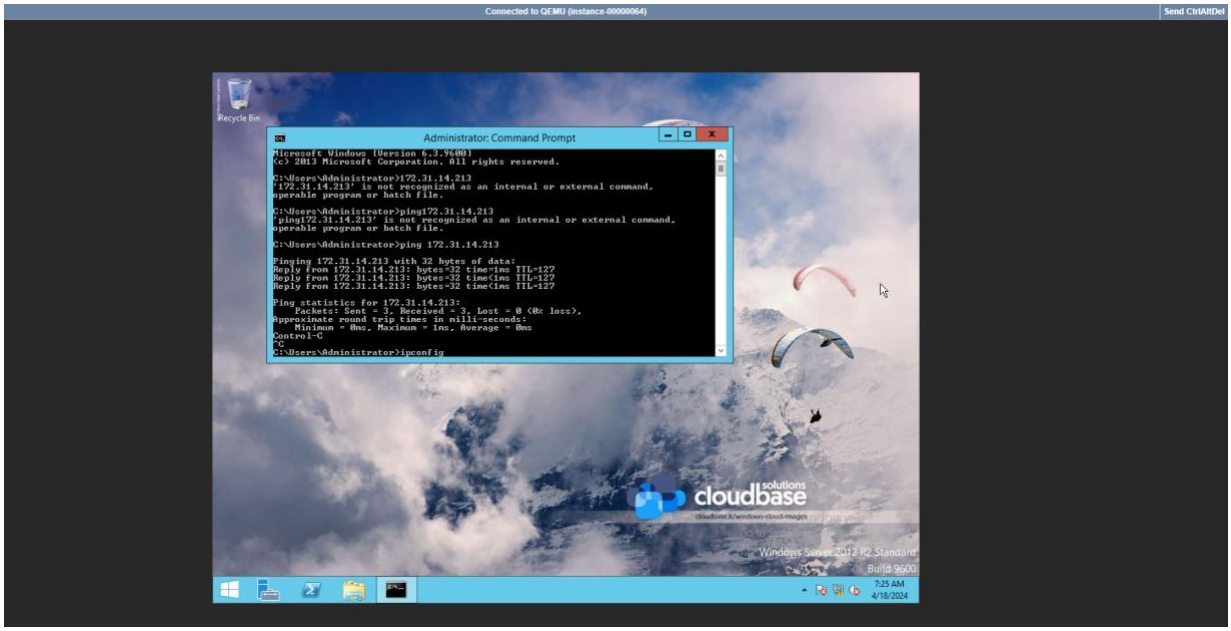
b. Centos 8 instance



c. Ubuntu 22.04 instance



d. Windows 2012 Server instance



5.3 DISCUSSIONS

a. Flexibility and Customization:

OpenStack's modular architecture allows for significant customization to meet specific organizational needs. Each service can be configured and extended independently, providing flexibility that is not always available in proprietary cloud solutions. Custom plugins and drivers can be developed and integrated into OpenStack, further enhancing its capabilities and adaptability.

b. Community and Support:

The strong OpenStack community provides extensive documentation, forums, and support channels, which are invaluable for troubleshooting and optimizing deployments. Regular updates and contributions from the community ensure that OpenStack stays up-to-date with the latest technologies and industry trends.

c. **Complexity of Setup:**

Despite its powerful capabilities, the initial setup and configuration of OpenStack can be complex and time-consuming. Organizations may need to invest in specialized training or consulting services to deploy OpenStack effectively. However, once set up, the management and operation of the OpenStack environment become more straightforward, especially with the use of automation tools like Ansible.

d. **Integration with Existing Systems:**

OpenStack integrates well with existing IT infrastructure, including legacy systems and other cloud platforms. This interoperability is crucial for organizations transitioning to a cloud-based model without disrupting their existing operations. APIs provided by OpenStack allow for seamless integration with external systems and services, enhancing its utility in hybrid cloud environments.

e. **Future Prospects:**

The future of OpenStack looks promising, with ongoing developments focusing on containerization, edge computing, and support for emerging technologies such as IoT. The integration with Kubernetes and other container orchestration platforms is expected to enhance OpenStack's capabilities in managing containerized applications and microservices architectures.

f. **Real-world Applications:**

Case studies of organizations using OpenStack highlight its versatility and effectiveness in various sectors, including telecommunications, academia, and government.Successful implementations have shown improvements in operational efficiency, scalability, and cost savings, reinforcing OpenStack's value proposition as a cloud infrastructure solution.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The deployment of an OpenStack Virtualized cloud platform brings substantial advantages aiming to streamline the IT infrastructure at NRSC. Granting users the autonomy to provision and oversee resources enhances the cloud environment's flexibility, effectiveness, and scalability. For instance, OpenStack's various components like Nova for compute, Neutron for networking, and Cinder for block storage work together to deliver a comprehensive cloud environment. The modular architecture allows for flexible configurations and efficient resource management. Additionally, OpenStack's Heat service for managing stacks, Swift for object storage, and Keystone for identity services all contribute to a robust ecosystem, ensuring effective operations and scalability. Robust security measures embedded within OpenStack Keystone ensure data confidentiality and integrity, fostering trust and adherence to regulatory standards.

6.2 FUTURE WORK

Future work on enhancing the effectiveness of the OpenStack Virtualized cloud through automatic scaling could include:

a. **Dynamic Resource Allocation**: Implementing algorithms to automatically adjust computing resources based on demand, ensuring optimal performance during peak usage periods.

b. **Automation and Orchestration:** A potential avenue for future work involves integrating automation and orchestration capabilities through tools like OpenStack Heat and Nova Scheduler. This would focus on streamlining resource provisioning and improving workload placement, ultimately optimizing cloud resource management.

c. **Predictive Analytics:** Leveraging machine learning to analyze historical usage patterns and predict future resource needs, enabling proactive scaling to meet anticipated demand.

d. **Integration with Monitoring Tools:** Enhancing integration with monitoring tools to trigger automatic scaling actions in response to predefined thresholds or anomalies in system metrics.

e. **Cost Optimization:** Developing algorithms to optimize resource allocation and scaling decisions based on cost considerations, minimizing expenses while maintaining performance levels.

f. **User-Centric Interface:** Enhancing the user interface to provide visibility into automatic scaling events, allowing users to monitor resource usage and adjust scaling policies as needed to align with their specific requirements.

6.3 REFERENCES

a. RDO Project. (2023). Packstack installation guide. Retrieved from https://www.rdoproject.org/install/packstack/

b. OpenStack Documentation. (2023). Yoga release. Retrieved from https://docs.openstack.org/2023.1/

c. OpenStack Horizon Documentation. (2019). Retrieved from https://docs.openstack.org/horizon/latest/

d. OpenStack. (2018). API quick start guide. Retrieved from https://developer.openstack.org/api-guide/quick-start/

e. OpenStack Marketplace. (2024). Retrieved from https://www.openstack.org/marketplace