Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that ~~Mr~~/Miss Niyati Savant

of Computer Department, Semester VI with

Roll No. 2103156 has completed a course of the necessary

experiments in the subject CCL under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

8/4/24

Teacher In-Charge

Head of the Department

Date 08/04/24

Principal

TSEC

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1.	Introduction and Overview of Cloud computing.	1 - 8	19/01/24	
2.	To study and implement Hosted virtualization using Virtual Box and KVM.	9 - 24	29/01/24	
3.	To study and Implement Bare metal Virtualization using XEN.	25 - 28	02/02/24	
4.	To study and Implement Infrastructure as a Service using AWS EC2.	29 - 37	09/02/24	
5.	To study and implement Platform as a Service using AWS Elastic Beanstalk.	38 - 44	23/02/24	
6.	To study and Implement storage as a service using OwnCloud / AWS S3.	45 - 60	26/02/24	
7.	To study and Implement IAM practices on AWS	61 - 95	01/03/24	
8.	To study and implement Database as a Service on AWS RDS.	96 - 113	11/03/24	
9.	To study and Implement containerization using Docker.	114 - 126	15/03/24	
10.	To study and implement container Orchestration using Kubernetes.	127 - 135	18/03/24	
11.	Assignment 1	136 - 141	22/2/24	
12.	Assignment 2	142 - 145	05/03/24	
13.	Mini Project Report	146 - 157	08/04/24	

8/4/24

Niyati Savant
C31
2103156

Experiment - 01

Aim - Introduction and overview of Cloud Computing

Theory -

- Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models and four deployment models.

- Essential Characteristics:

1) On-demand self-service - A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider

2) Broad Network Access - Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g. mobile phones, tablets, laptops, and workstations).

Niyati Savant
C31
2103156

3) Resource Pooling. The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of provided resources but may be able to specify location at a higher level of abstraction (e.g. country, state or datacenter). Examples of resources include storage, processing, memory, and network bandwidth.

4) Rapid elasticity - Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

5) Measured Service - Cloud systems automatically control and optimize resource use by leveraging a metering capability 1 at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both provider and

Niyati Savant
C31
2103156

consumer of the utilized service.

• Service Models

1) Software as a Service (SaaS)
The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure 2. The applications are accessible from various client devices through either a thin client-interface (like a web browser) Eg.- web based email or program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, OS, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

2) Platform as a Service (PaaS)
The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, OS or storage but has control over deployed applications and possibly configuration settings for the application hosting environment.

Niyati Savant
C31
2103156

3) Infrastructure as a Service (IaaS)
The capability provided to the consumer is to provision processing storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include OS and applications. The consumer does not manage or control underlying cloud infrastructure but has control over OS, storage, and deployed applications; possibly limited control of select networking components (e.g., host firewalls)

- Deployment Models:

1) Private cloud
The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them

2) Community cloud
The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations) May be owned, managed, operated by one or more organizations in community, third party or some combination..

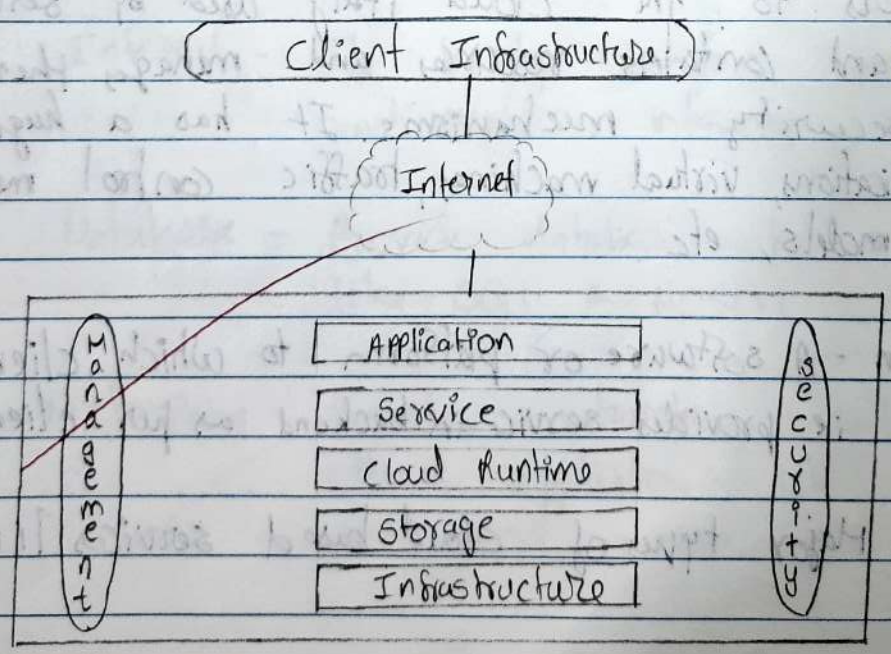
Niyati Savant
C31
2103156

3) Public Cloud -
The Cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, operated by a business, academic or government or some combination

4) Hybrid Cloud
It is a composition of two or more distinct cloud infrastructures that remain unique entities but are bound together by standardized or proprietary technology (e.g., cloud bursting for load balancing between clouds).

- Cloud Computing Architecture

The architecture is divided into two parts - Frontend and Backend



Frontend

Backend

Niyati Savant
C31
2103151

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Architecture of cloud computing is the combination of both SOA (Service Oriented Architecture) and EDA (Event Driven Architecture). Client Infrastructure, application, service, runtime cloud, storage, infrastructure management and security all these are the components of architecture

1) Frontend:
- It refers to the client side of the cloud system ie contains all UIs and applications used by client to access cloud services/resources. For eg: use of a web browser to access the cloud platform
- Client Infrastructure is a part of the frontend component. It contains the applications and UIs which are required to access the cloud platform ie provide GUI to interact

2) Backend:
It refers to the cloud itself used by service provider and contains resources and manages them and provides security mechanisms. It has a huge storage virtual applications, virtual machines, traffic control mechanisms, deployment models, etc.

- Application - A software or platform to which client accesses ie provides service in backend as per client requirement

- Service - Major types of cloud based services like

Niyati Savant
C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

SaaS, PaaS and IaaS and manages the type of servic the user accesses

- Runtime Cloud - Provides the execution and Runtime platform/ environment to Virtual Machine.

- Storage - Provides flexible and scalable storage servic and management of stored data

- Infrastructure - Hardware and software components like server, storage, network devices, virtualization software

- Management - Management of components like application, servic, runtime cloud, storage, infrastructure and other security mechanisms

- Security - Implement mechanisms for secure resources, systems, files and infrastructure

- Internet - The bridge between front and back end and establish interaction & communication

- Database - Provide database to store structured data like SQL & NOSQL

- Networking - Provide networking infrastructure for cloud application like load balancing, DNS and VPNs.

Niyati Savant
C31
2103156

TSEC
ENGINEERING COLLEGE

- Analytics - Provide analytic capabilities like, ML warehousing, BI.

- Benefits of Cloud Computing

1) Backup and data restoration
2) Improved collaboration
3) Quick and easy accessibility
4) Low maintenance cost
5) Mobility
6) IServices in pay-per-use model
7) Unlimited storage capacity
8) Data security

- Limitations

1) A good internet connection is must
2) Vendor lock-in - Difficult to move from one cloud to another due to different platforms
3) Limited control over function and execution of services
4) Though secure, ~~do~~ organizations sensitive data is send to the service provider which may be misused.

13/24

Nivati Savant
TE - C31
2103156 Experiment 2

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE
02102124

Aim - To study and Implement Hosted virtualization using Virtual Box & KVM

Theory -

Virtualization in Cloud computing -

Virtualization is a technique that allows to share single physical instance of an application or resource among multiple organizations. This is done by assigning a logical name to a physical resource and providing a pointer to that on demand. The machine on which virtual machine is created is host machine and virtual machine is guest machine. The virtual machine is managed by a software or firmware called hypervisor.

Types of Virtualization -

1) Hardware - The virtual Machine manager (VMM) is installed directly on hardware system. After virtualization of hardware system, different OS can be installed on it.

2) Operating System - The VMM is installed on Host operating system instead of directly on hardware. Mainly used for testing

3) Server Virtualization - The VMM is directly

installed on Server system. It is done because a single physical server can be divided into multiple servers on the demand basis and for balancing the load.

4) Storage - Grouping physical storage from multiple network storage devices so that it looks like a single storage device. Done for back-up and recovery.

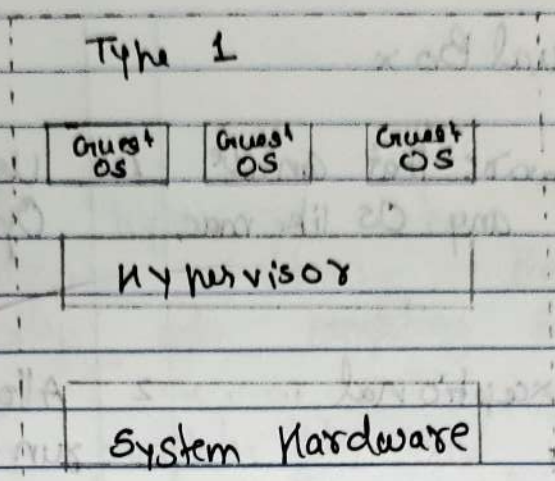
Benefits of virtualization:
1) More flexible and efficient resource allocation
2) Enhance development productivity.
3) Lowest IT infrastructure cost and easier management
4) Better scalable and more agile business processes.

Hypervisor
A form of virtualization software used in Cloud hosting to divide and allocate the resource on various pieces of hardware and the program that provides partitioning, isolation or abstraction is virtualization hypervisor. It is also called VMM. They help reduce space usage, energy and maintenance requirements of servers. Its types are-

Type - 1 Hypervisor-
It runs directly on underlying host system. Also called "Native Hypervisor" or "Bare metal

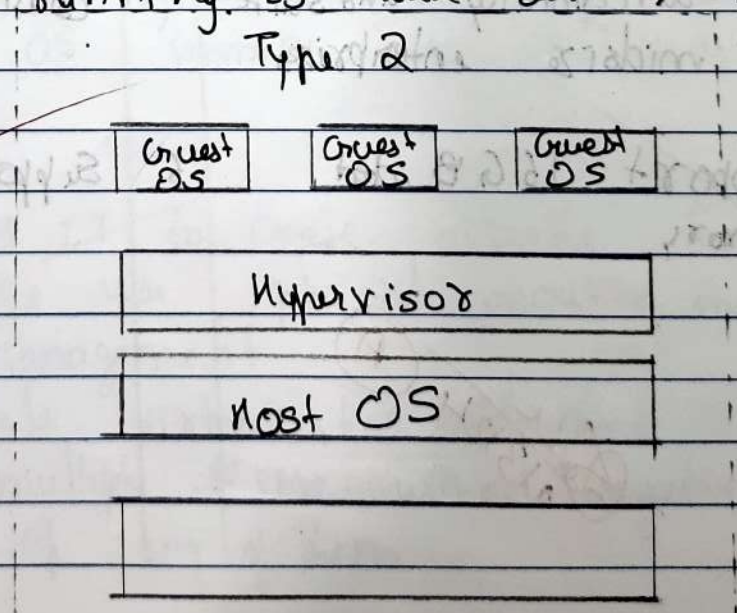
Niyati Savant
TE - C31
2103156

They have direct access to physical hardware resources Examples include VMware ESX and ESXi, Oracle VM.



Type 2 · Hypervisor

Also called Hosted Hypervisor, the host OS runs on underlying host system. They run as an application in a Host system (physical machine) Allow quick and easy access to guest OS alongside host machine running. Eg VMware Server, KVM



Comparison between Virtual Box and KVM

	Virtual Box		KVM
1	Free software that can be used on any OS like mac, windows	1	Used only on Linux Operating System
2	Offers exceptional stability	2	Allow computer servers to run efficiently
3	Allow server flexibility	3	Provide better and finer scaling
4	slower	4	Faster since experience is nearer to real machine
5	Has attribute of copying bi-directionally and suited for midsize enterprises	5	Has all attributes of command line
6	Support 16 GB dat memory	6	Support 4 TB memory

A

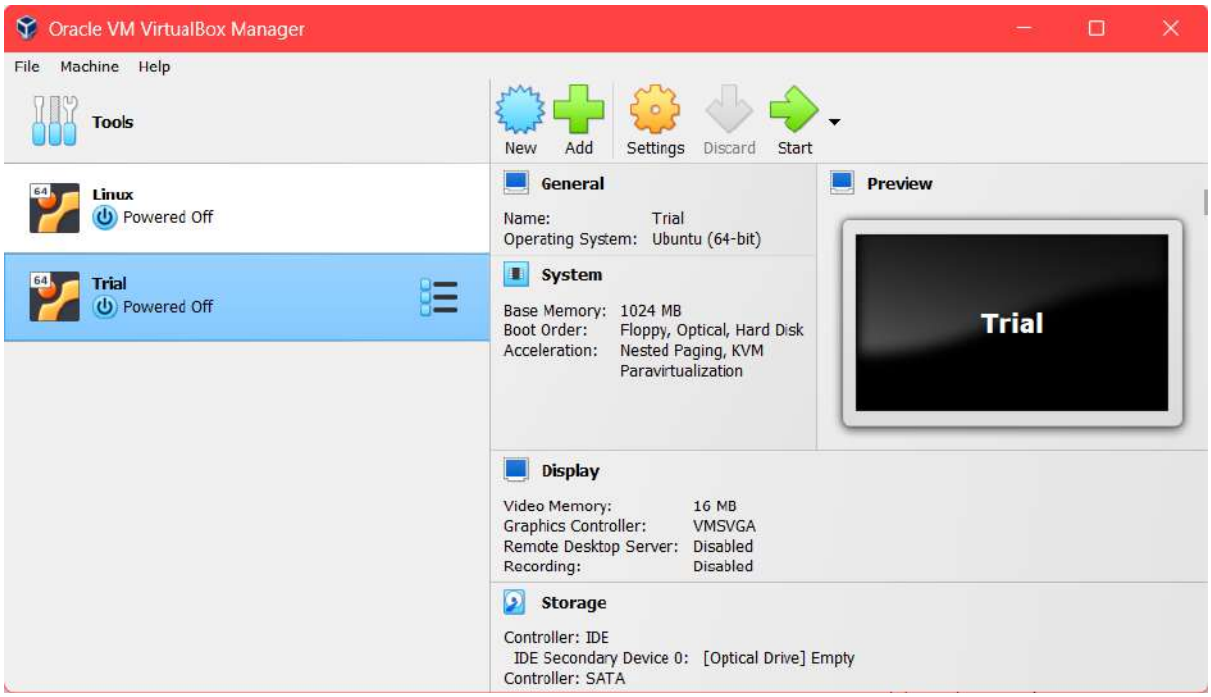
Experiment 2

Creating and running virtual machines on Hosted Hypervisor like Virtual Box and KVM

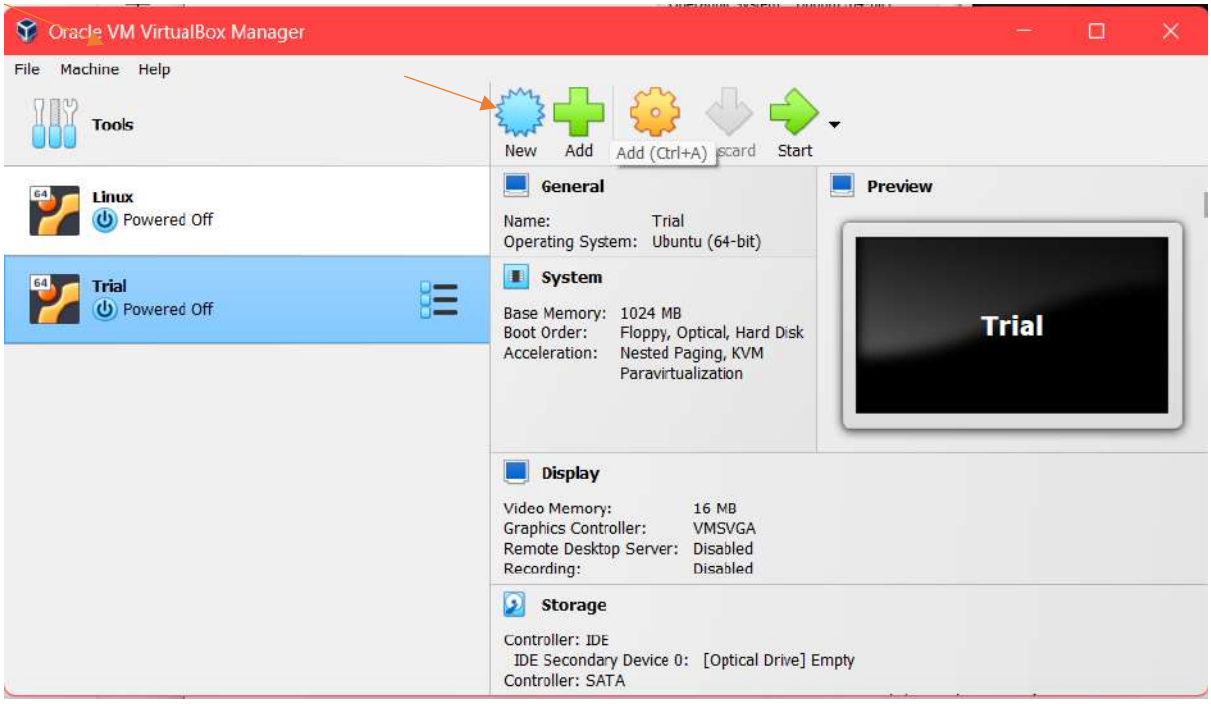
1) Hosted Virtualization on Oracle Virtual Box Hypervisor

Step 1: Download Oracle Virtual box from https://www.virtualbox.org/wiki/Downloads

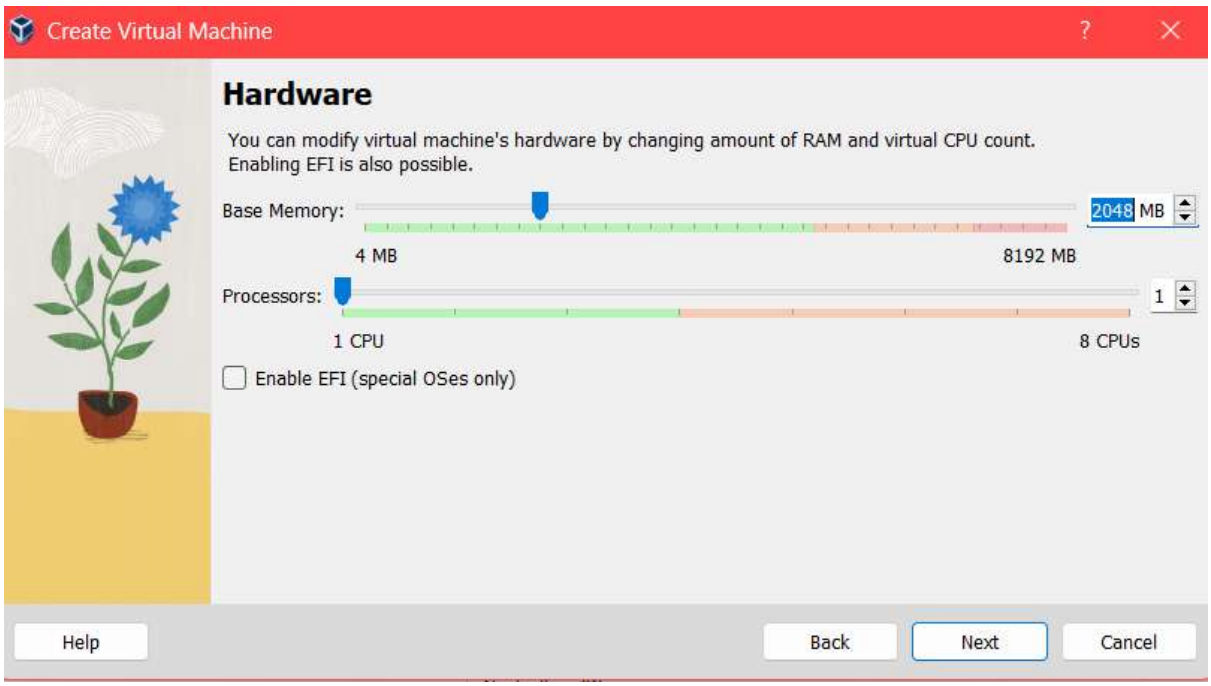
Step 2: Install it in Windows, Once the installation has done open it.



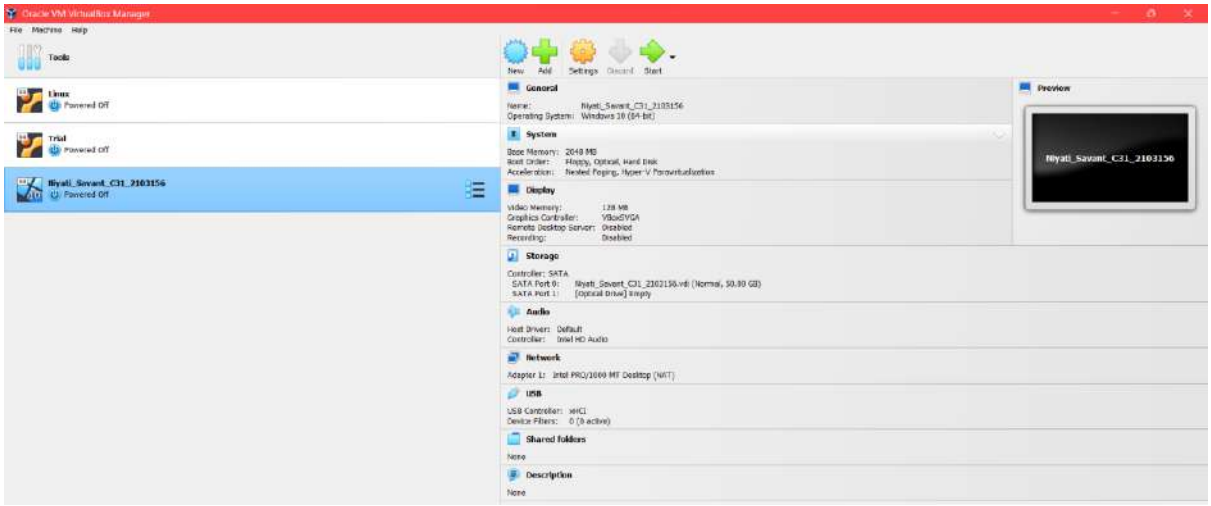
Step 3:-:Create Virtual Machine by clicking on New



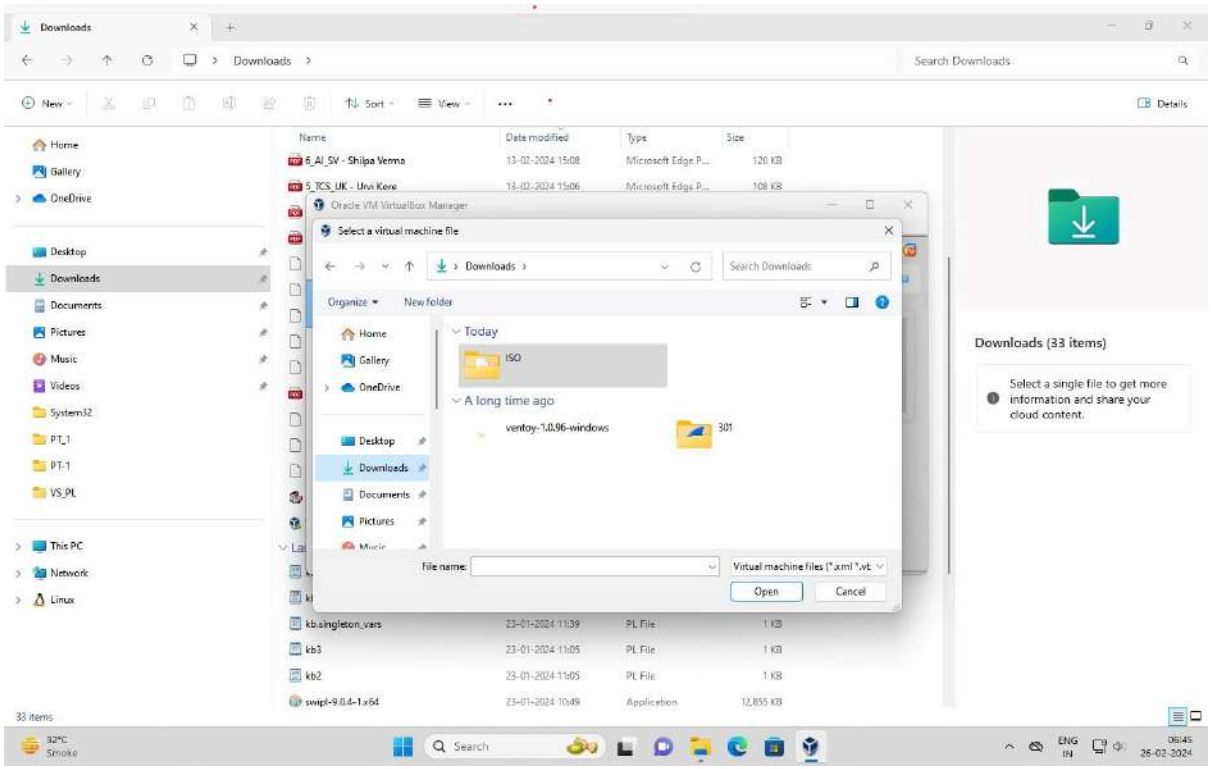
Step 4-: Specify RAM Size, HDD Size, and Network Configuration and Finish the wizard



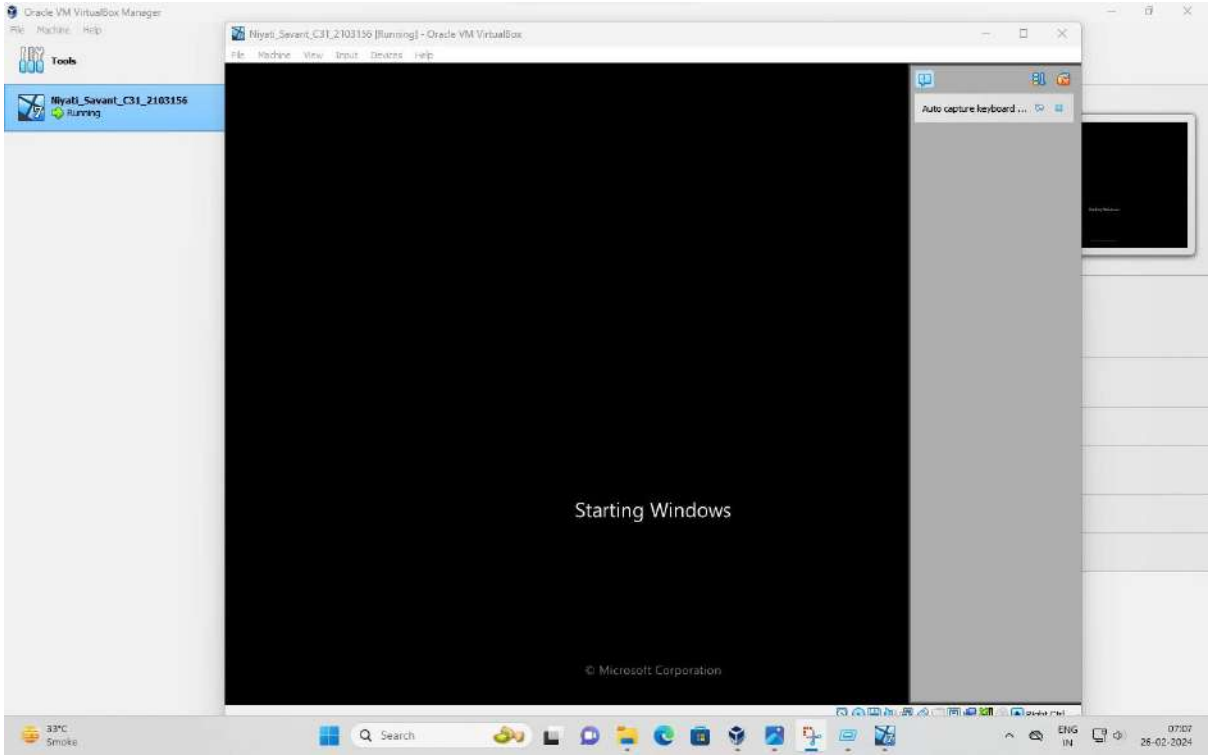
Niyati_Savant_TE_C31_2103156

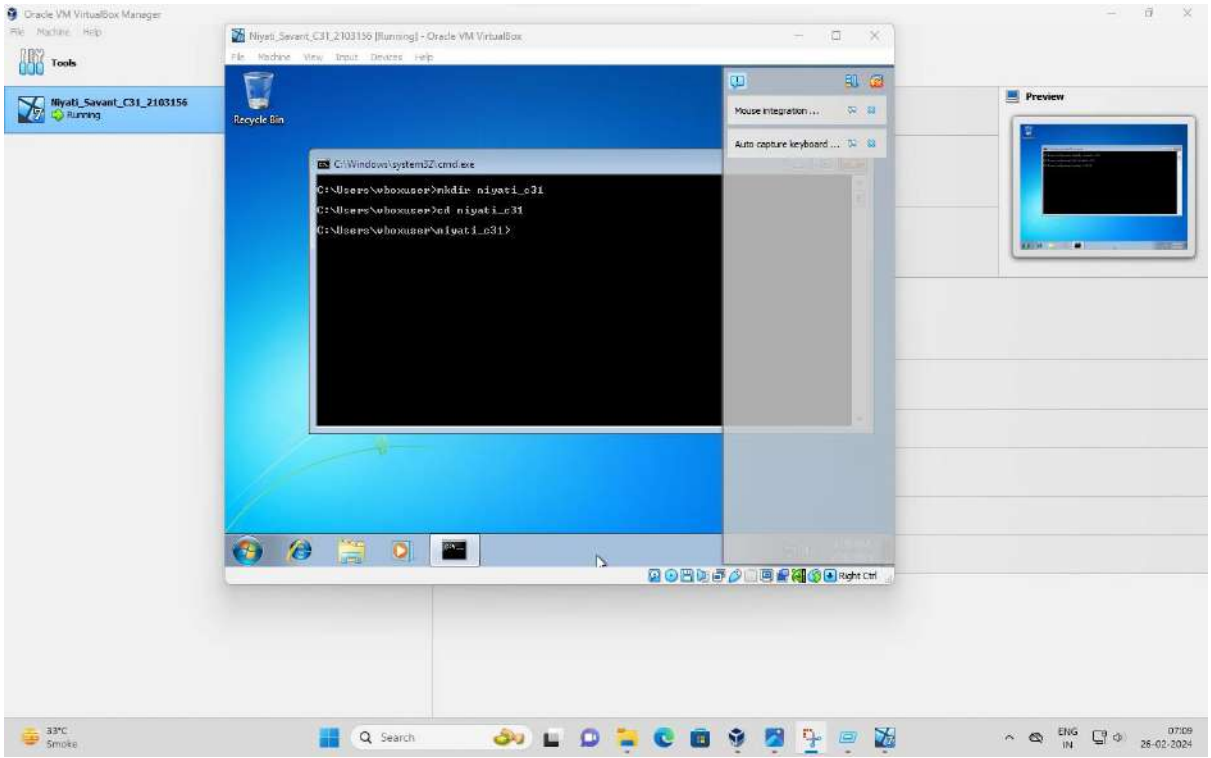


Step 4-: To Select the media for installation Click on start and browse for iso file



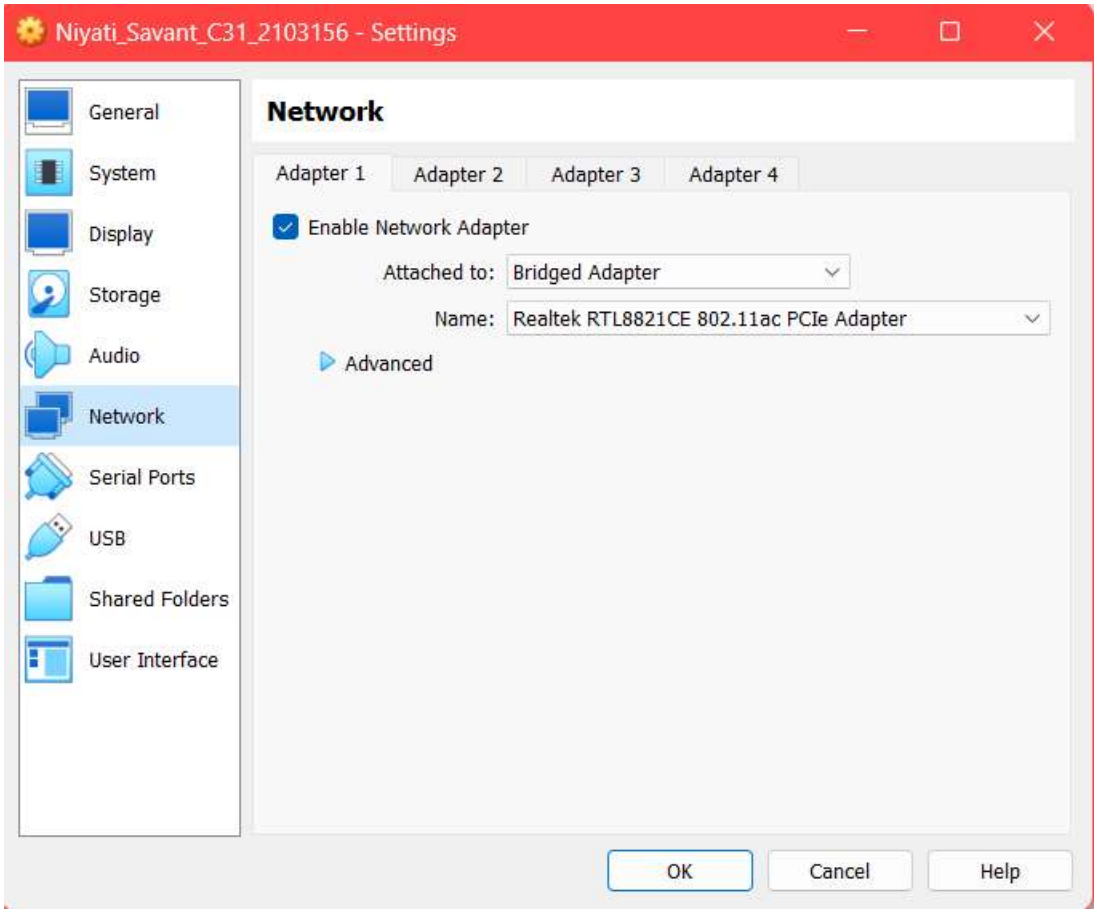
Step 5:Complete the Installation and use it.





Step 6: To Connect OS to the network change network Mode to Bridge Adaptor

Niyati_Savant_TE_C31_2103156



2) Hosted Virtualization on KVM Hypervisor

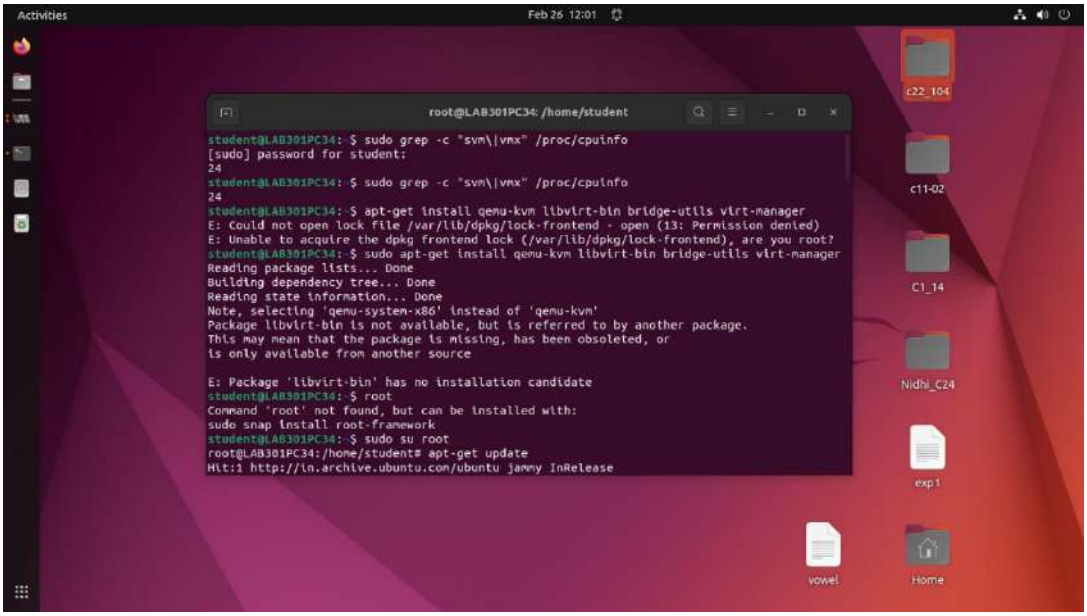
The Steps to Create and run Virtual machines in KVM are as follows

1) Check whether CPU has hardware virtualization support.

   KVM only works if your CPU has hardware virtualization support – either Intel VT-x or AMD□V. To determine whether your CPU includes these features, run the following command:

   #sudo grep -c "svm\|vmx" /proc/cpuinfo

   A 0 indicates that your CPU doesn't support hardware virtualization, while a 1 or more indicates that it does.

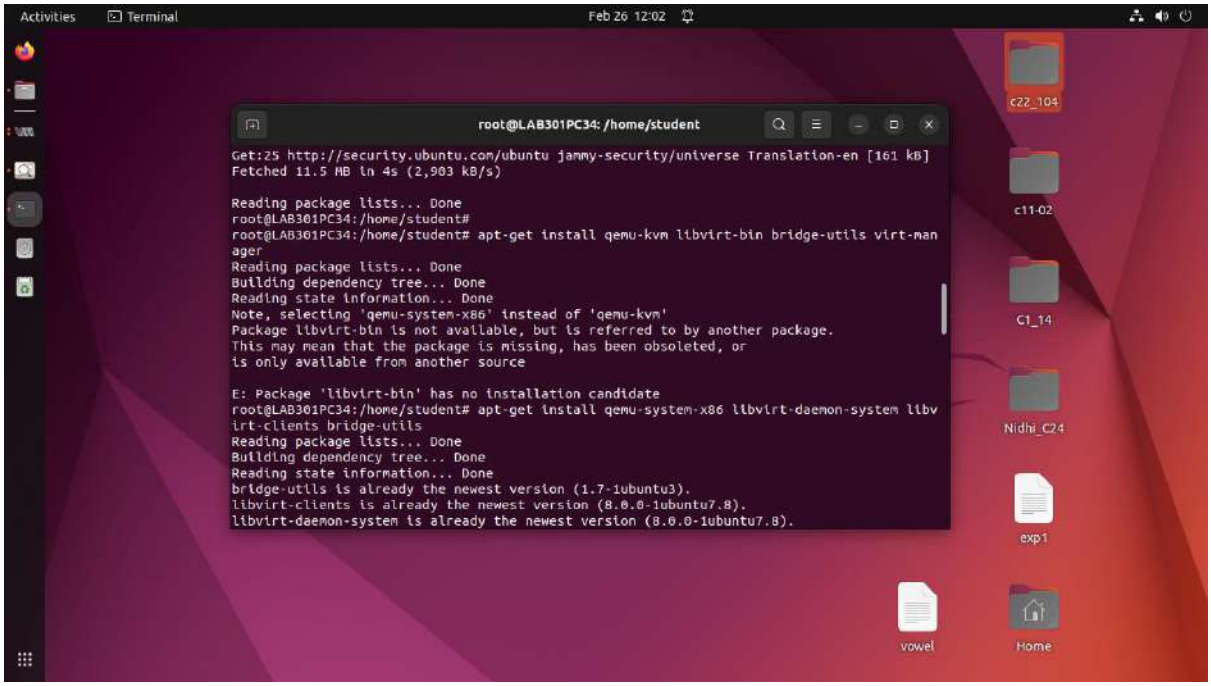


2) Install KVM and supporting packages.

Virt-Manager is a graphical application for managing your virtual machines.you can use the kvm

command directly, but libvirt and Virt-Manager simplify the process.

#sudo apt-get install qemu-kvm libvirt-bin bridge-utils virt-manager

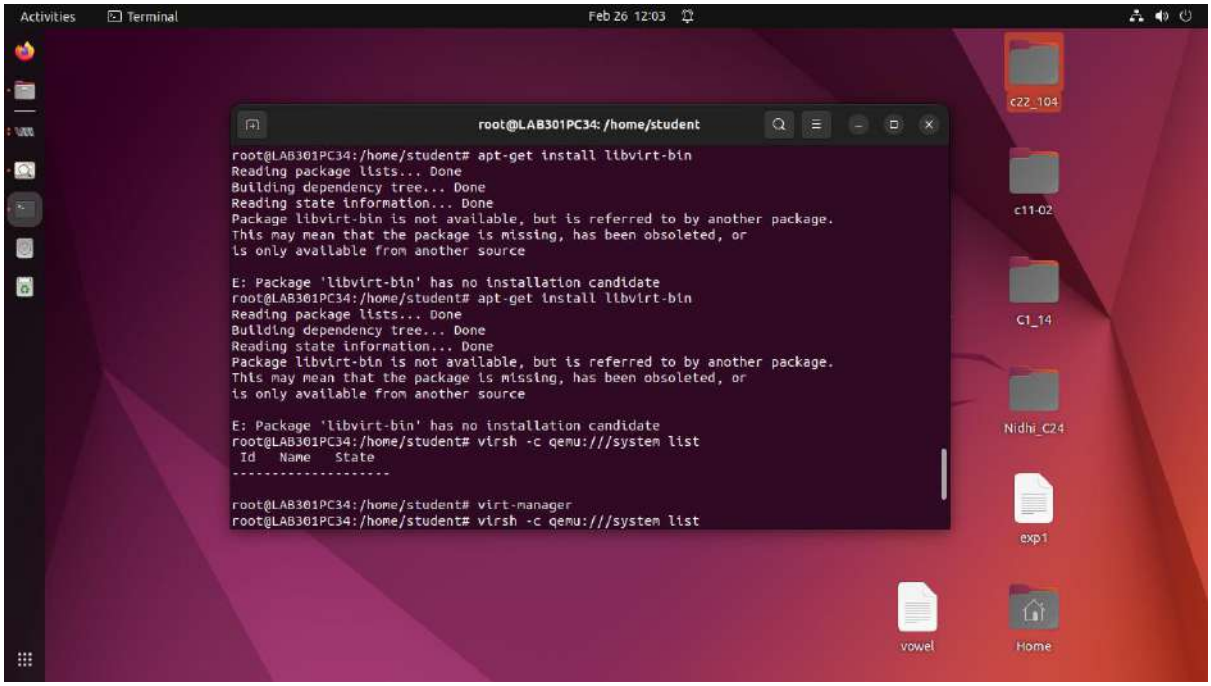


3) Check whether everything is working correctly.

Run following command after logging back in as tsec and you should see an empty list of virtual machines.

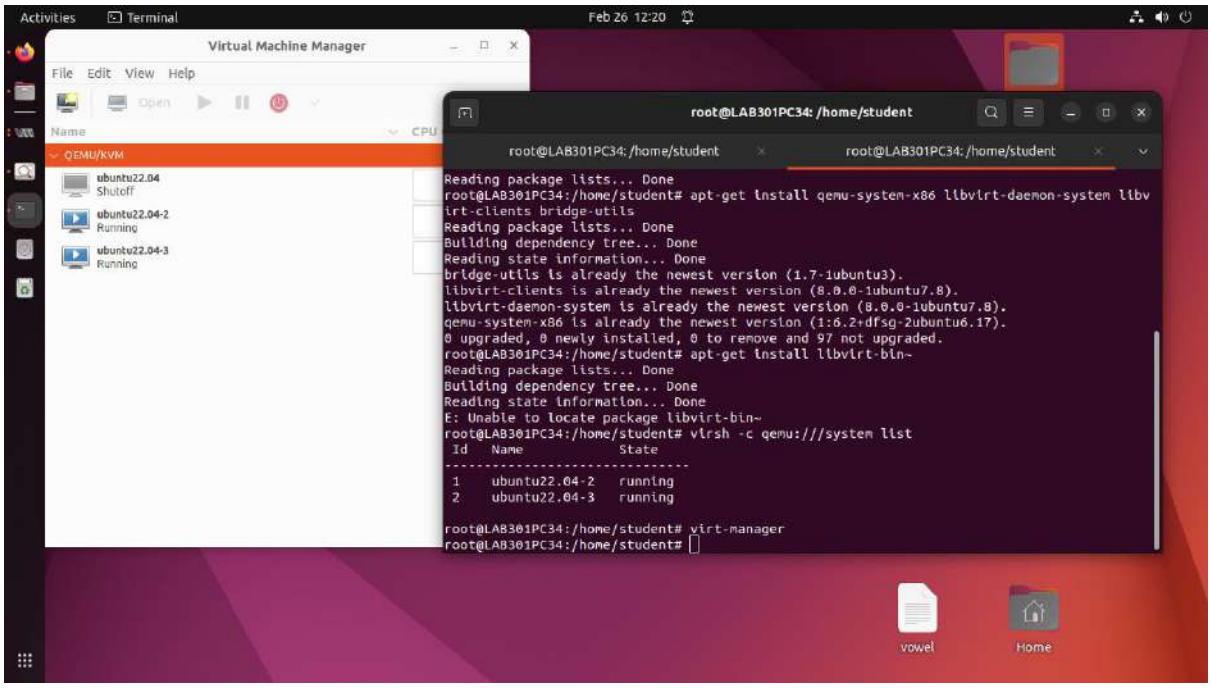
This indicates that everything is working correctly.

#virsh -c qemu:///system list

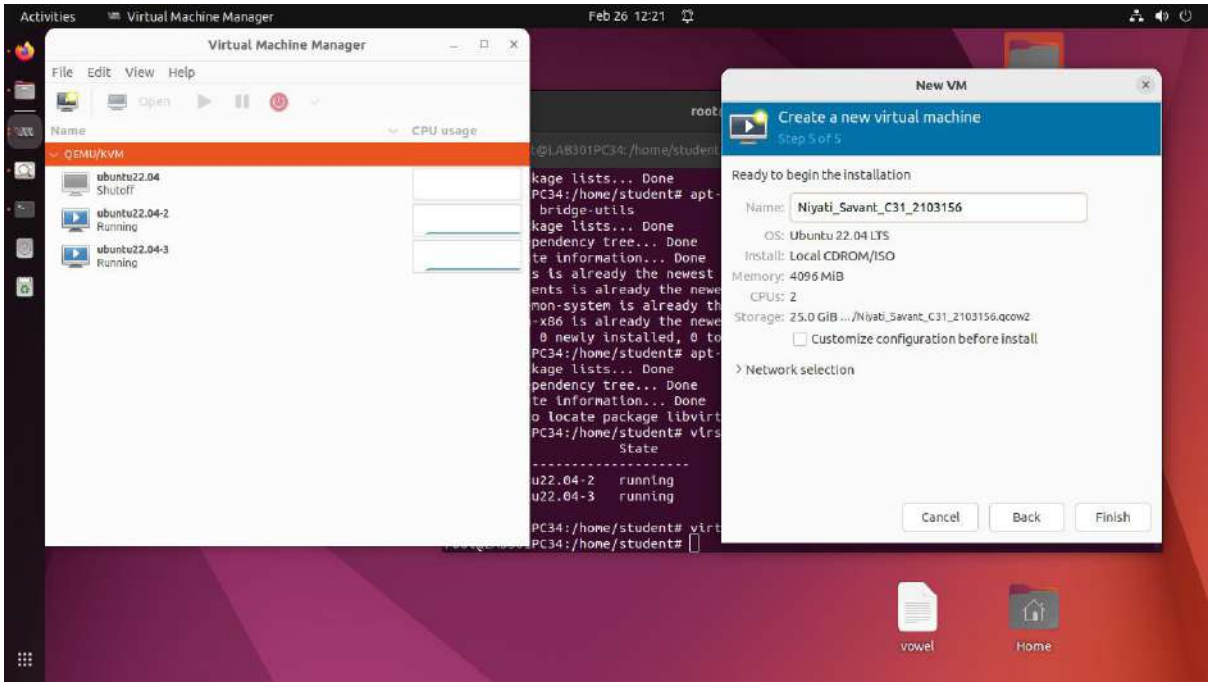


4)Open Virtual Machine Manager application and Create Virtual Machine

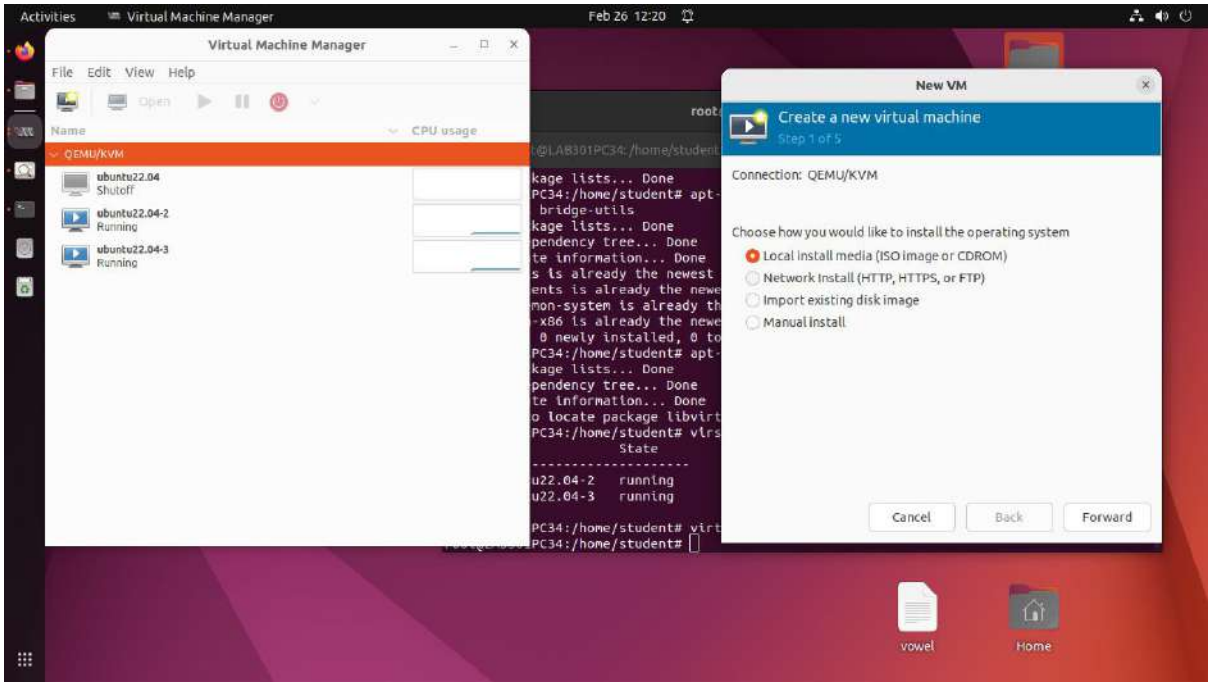
#virt-manager

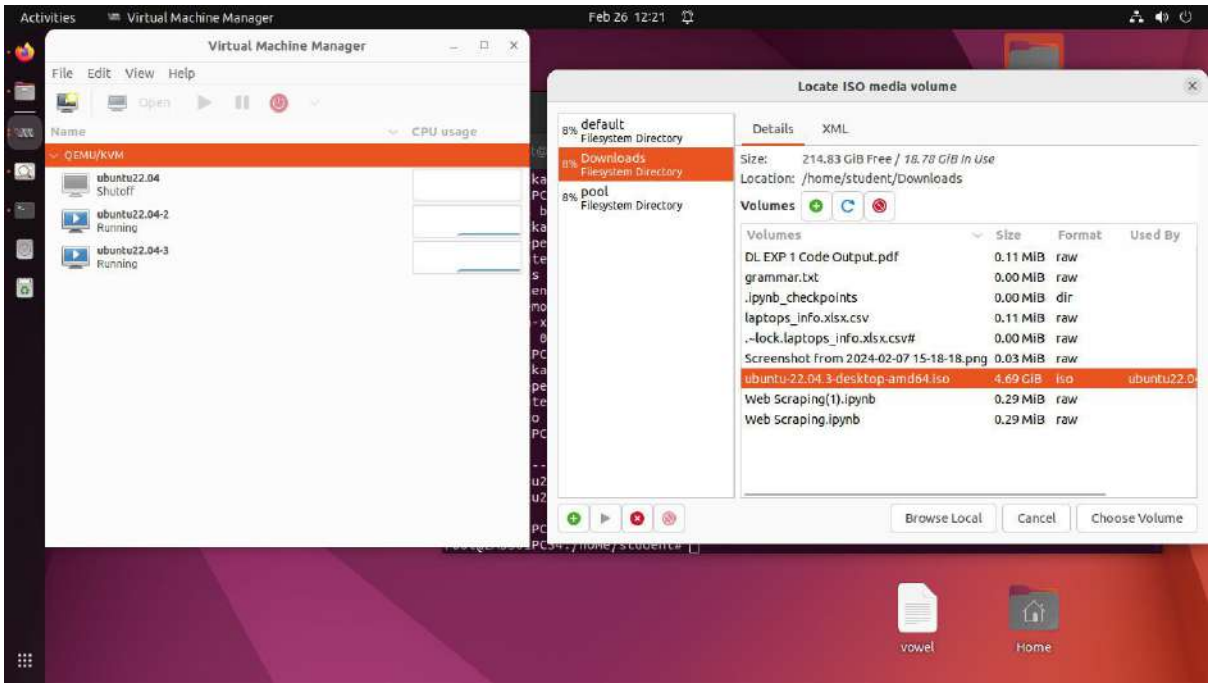


5) Create and run Virtual Machines

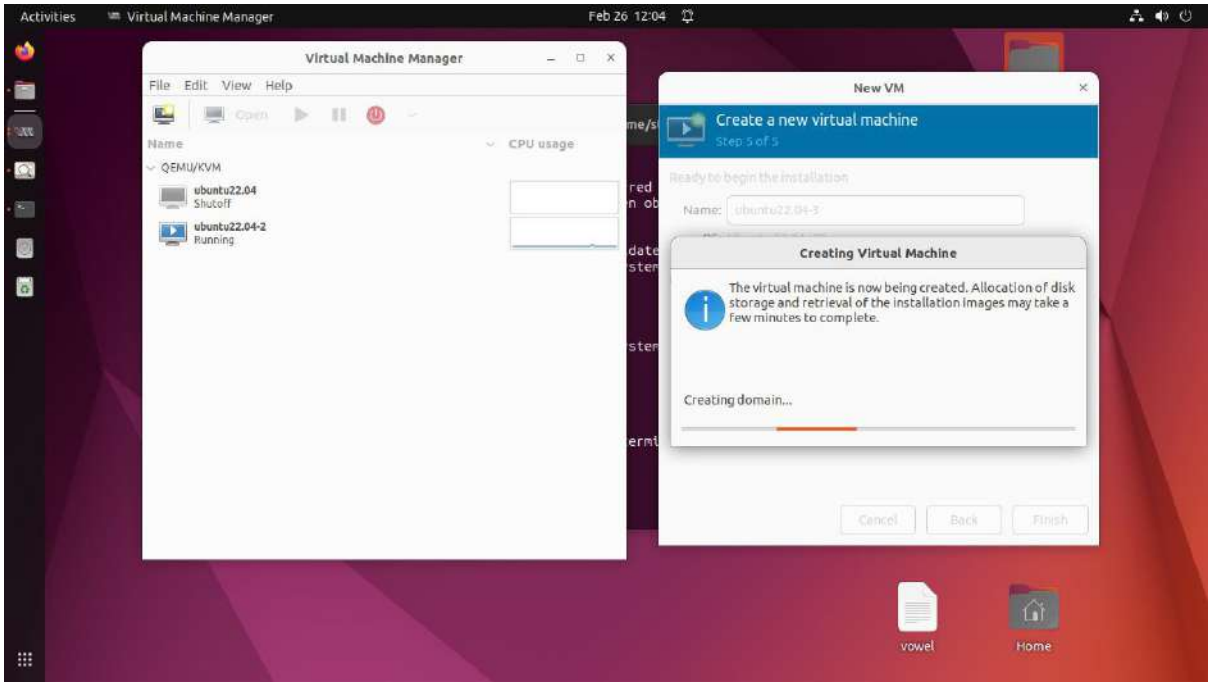


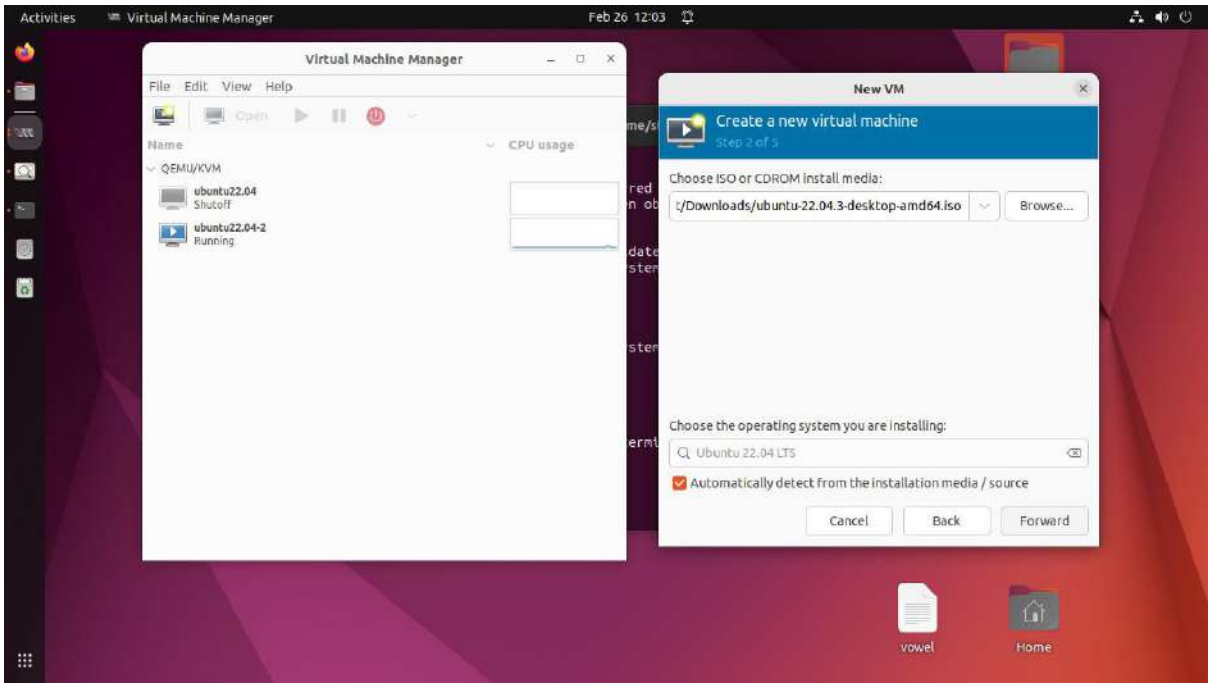
Niyati_Savant_TE_C31_2103156



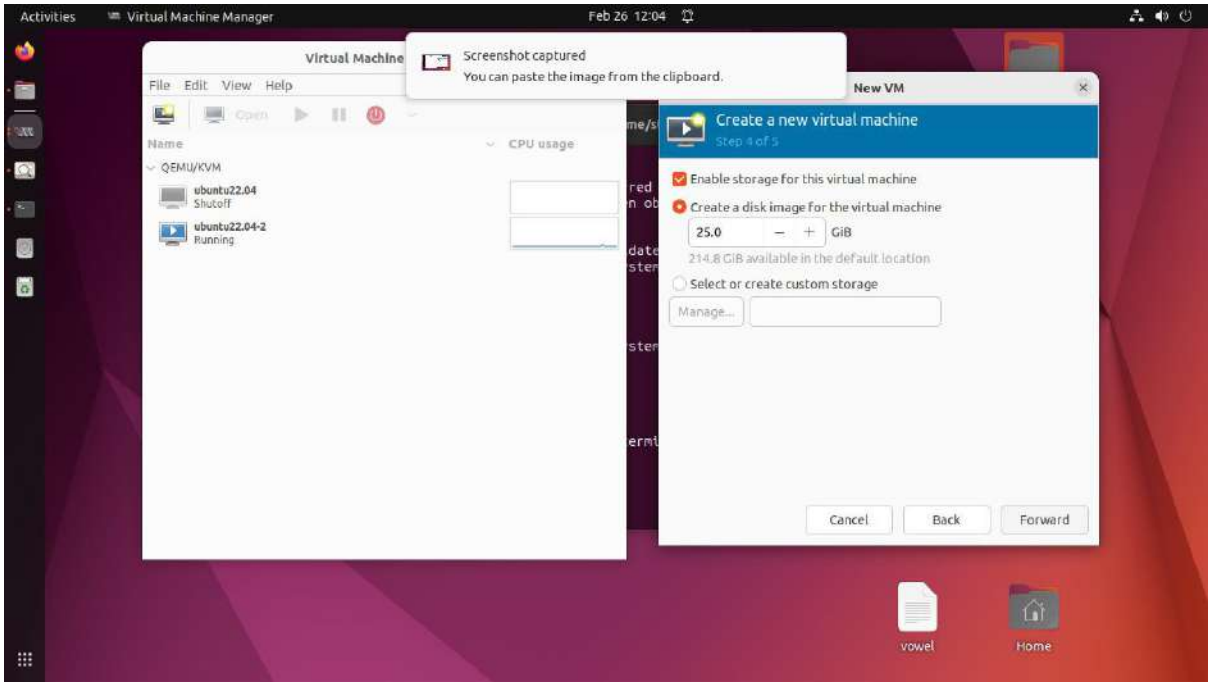


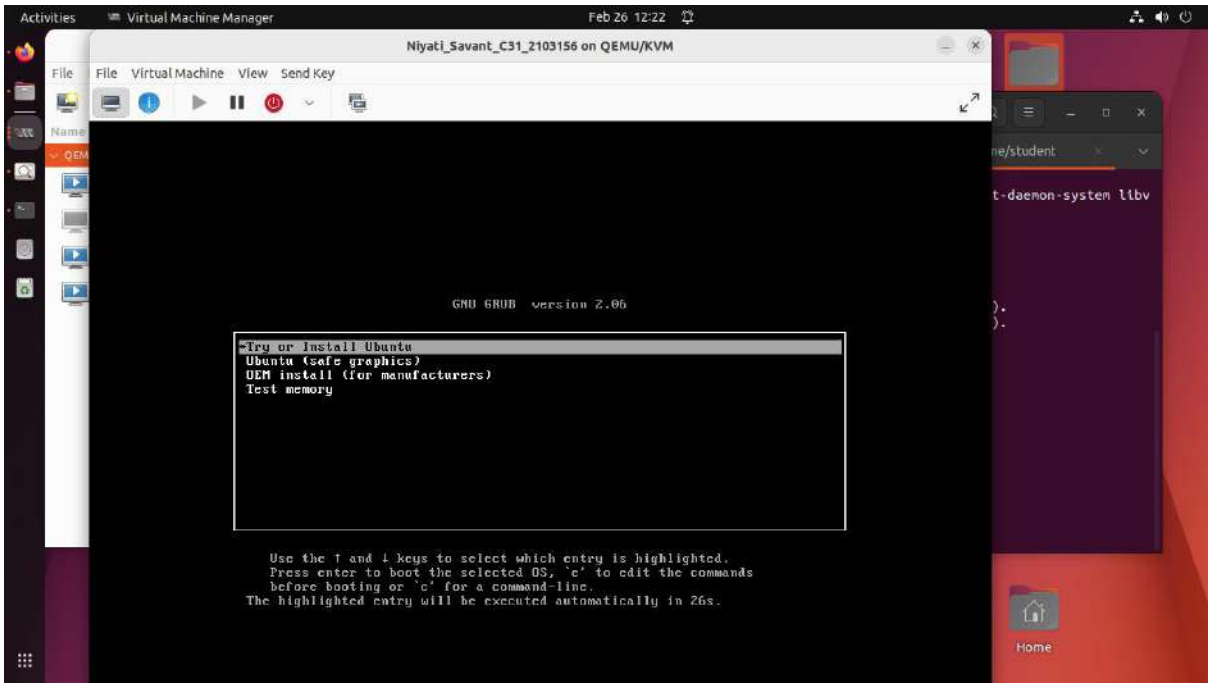
Niyati_Savant_TE_C31_2103156



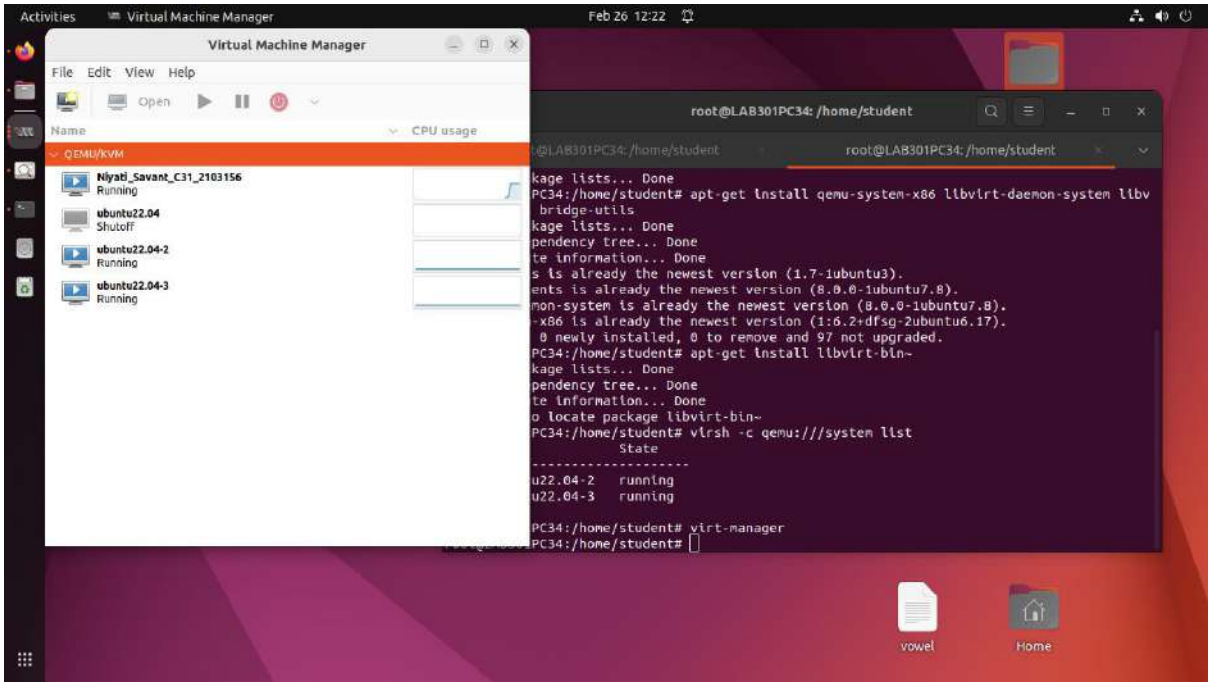


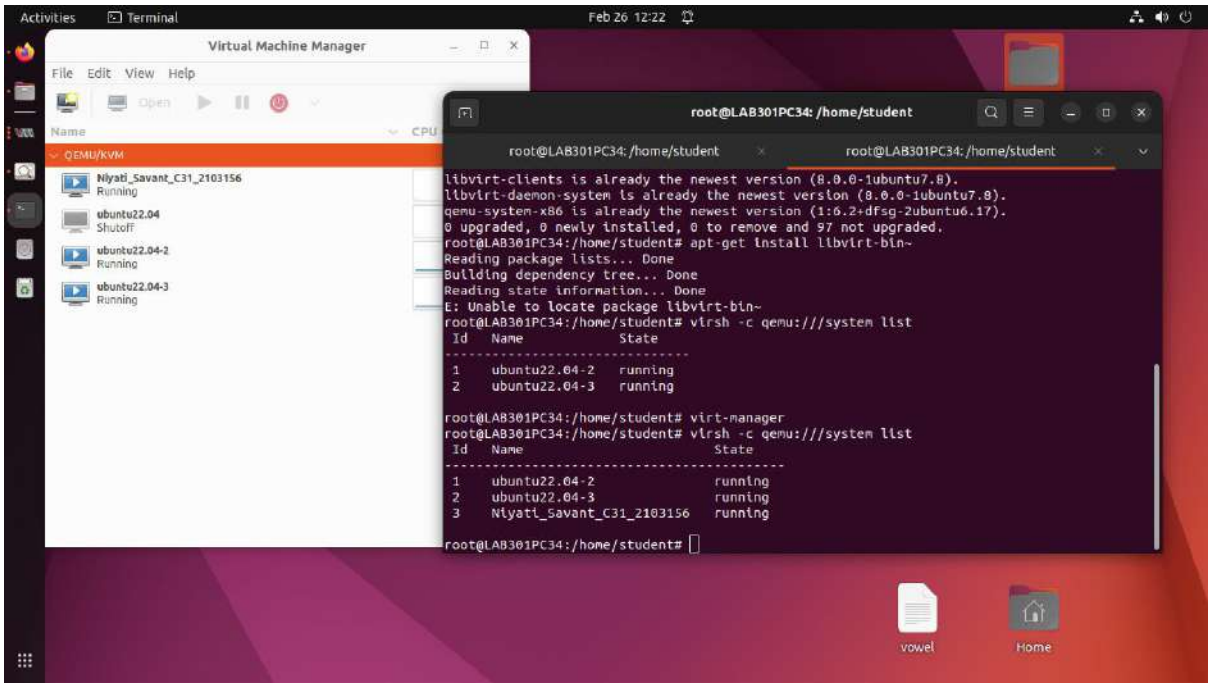
Niyati_Savant_TE_C31_2103156





Niyati_Savant_TE_C31_2103156





Niyati Savant
TE - C31
2103156

Experiment 3

5/2/23

Aim: To Study and implement Bare-metal Virtualization using Xen, HyperV or VMware Esxi.

Theory -

A bare metal hypervisor aka Type-1 hypervisor is a virtualization software that has been installed directly onto computing hardware. It controls hardware and one or more guest OS. They feature high availability and resource management; they also provide better performance, scalability, and stability due to direct hardware access but this can also limit the hardware support of built in device drivers.

Some functions performed -

1] Enhance security
By ensuring logical isolation of every guest VM, they make OS immune to malicious software and activities.

2] Scalability
Allows IT to create sizable failover clusters and help VMs use physical resources more efficiently

3] Resource management

4] Large-scale virtualization deployment
Manage multiple VMs without negatively impacting the performance of any of them

A bare metal is better than hosted one as the bare metal is installed directly onto the servers hardware. The hosted one runs on top of parent OS. A VM can access directly the hardware in case of bare metal but the one running on hosted must pass hardware requests through parent OS.

Horizontal Scaling

The architecture of application is designed to scale out by adding more instances or machines to existing pool of resources. Each instance operates independently, handling its share of the incoming requests. The distribution of load allows it to effectively handle increased traffic loads. It is used in cloud computing environment where instances can be easily provisioned or deprovisioned based on demand. It offers better fault tolerance as failure of one instance does not necessarily bring down entire system but it may require additional efforts for managing distributed ~~sys~~ systems and ensuring data consistency across multiple instances.

Vertical Scaling

Involves increasing capacity of individual instances by adding more resources like CPU, RAM or storage to existing machine. It is suitable for applications that have a limit to how much they can scale horizontally or when

Niyati Savant
TE - C31
2103156

a single powerful instance can handle the workload efficiently. It is often used for scaling up traditional monolithic applications or legacy systems that cannot be easy distribute across multiple instances. It is less complex but also has limitations in terms of scalability and cost-effectiveness.

Auto-scaling

It is a dynamic provisioning technique used in cloud computing environments to automatically adjust the number of resources allocated to an application based on its current demand or workload. The main goal of auto-scaling is to ensure that the application can maintain optimal performance while efficiently utilizing resources and minimizing costs. These systems continuously monitor various metrics like CPU utilization, memory usage, network traffic, or custom application-specific metrics. Based on predefined policies or rules set by administrators these systems automatically trigger scaling actions to add more resources (scale out) or remove excess resources (scale in) as needed. It can be implemented using native cloud services provided by cloud providers or through third-party tools and platforms that offer auto-scaling capabilities.

Load Balancing

It is critical component of distributed systems architecture used to evenly distribute incoming network traffic across multiple servers, instances or resources. The primary goal is to optimize resource utilization, maximize throughput, minimize response time and ensure high availability and reliability of applications. It acts as intermediary between clients and backend servers, receiving incoming requests and forwarding them to appropriate servers based on various algorithms and criteria

Niyati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Experiment 4

Aim - To study and implement Infrastructure as a service (IaaS) using AWS

Theory -

Amazon EC2 (Elastic Compute Cloud)

Amazon Web Service EC2 is provided by AWS cloud which is secure, resizable and scalable. These VMs are pre-configured with OS and required software. Instead of managing infrastructure AWS will do it so that user can just launch and terminate the EC2 instance as per need. The EC2 instance can be scaled up and down as per incoming traffic. Plus, it a pay-as-you-go model.

Amazon Machine Image (AMI)

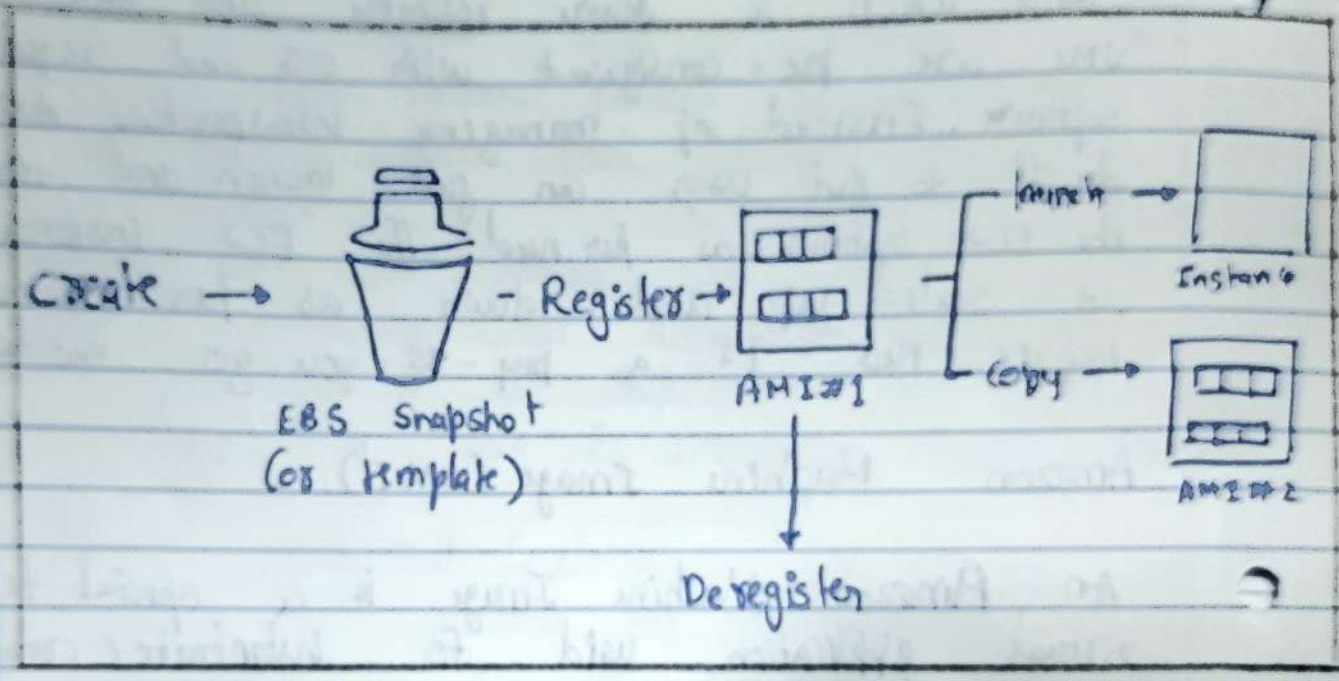
An Amazon Machine Image is a special type of virtual application used to instantiate (create) a VM within EC2. It serves as the basic unit of deployment for services delivered using EC2.

Whenever you want to launch an instance, the AMI must be specified.

(P.T.O)

Properties of AMIs

- A template for root volume for the instance
- Launch permissions that control which AWS accounts can use AMI to launch instances
- A block device mapping that specifies volumes to attach to the instance when it is launched



The different types of AMIs include -

a) Amazon-provided AMIs
b) Marketplace AMIs
c) Community AMIs
d) Custom AMIs

Niyati Savant
TE - C31
2103156

Types of EC2 Computing Instances

Different Amazon EC2 instance types are designed for certain activities.
These types are -:

a) General Purpose Instances
The computation, memory and networking resources in general purpose instances are balanced.
We can use general-purpose instances in scenarios like gaming servers, small databases etc.

b) Compute-optimized Instances
These instances are appropriate for applications that need a lot of computation and help from high performance CPUs. Eg: Web server, Gaming servers.

c) Memory-Optimized Instances
These are geared for workloads that need huge datasets to be processed in memory

d) Storage-Optimized Instances
They are made for workloads that demand fast, sequential read and write access to huge datasets. Eg: Distributed file systems, data warehousing

e) Accelerate Computing Instances
Co-processors are used in these instances

to execute specific operations more effectively than software running on CPUs.
Eg: Floating-point numeric computations, graphics processing and data pattern matching

Elastic IP address -

A static IPv4 address more suited for dynamic cloud computing is called an Elastic IP aka EIP address. They are used to hide instances or software from AWS that fail. The address is mapped to another instance that when feasible is made available in our account. The internet can be used to access this address just like any other public IPV4 address. If the instance lacks a public IPV4 address that allows communication from internet to instance, we can associate the EIP with it. Currently IPV6 addresses are not supported.

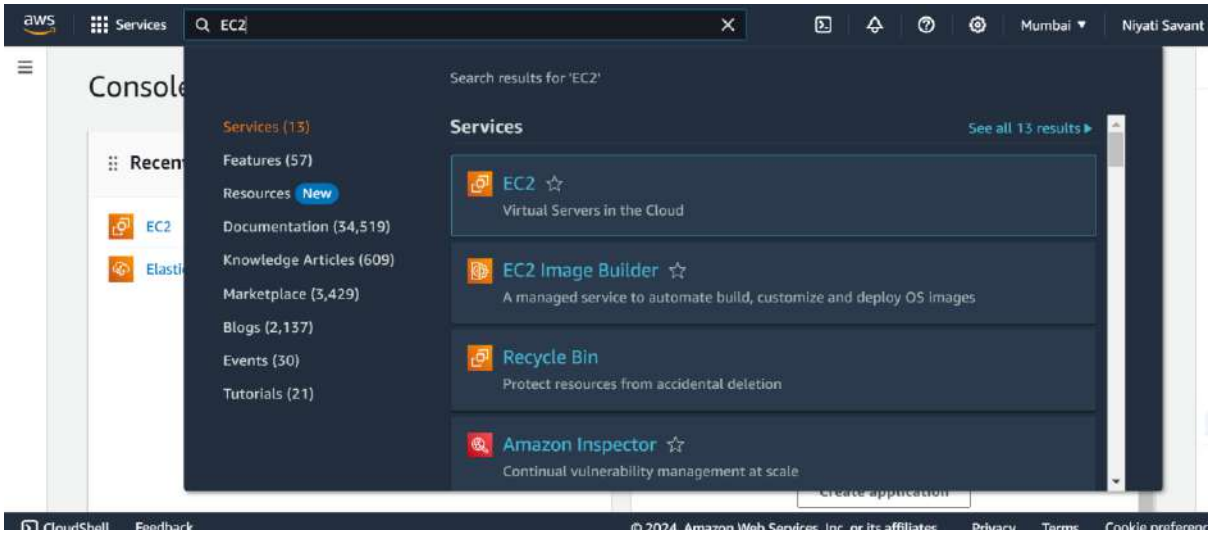
Remote Desktop Protocol (RDP)

RDP is secure network communications protocol developed by Microsoft. It enables network administrators to remotely diagnose problems that individual users encounter and gives users remote access to their physical work desktops. Used by work-from-home employees and support technicians and admins ~~sup~~ providing system maintenance.

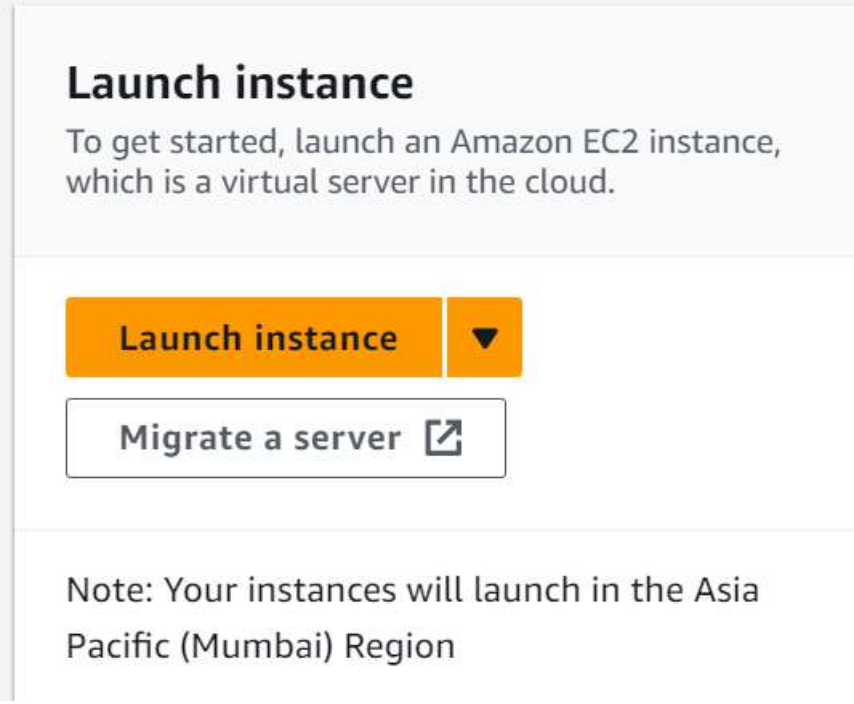
4/3/24

Experiment 4
Creating an EC2 Instance

Step 1 – Search for **EC2** in **Services**.

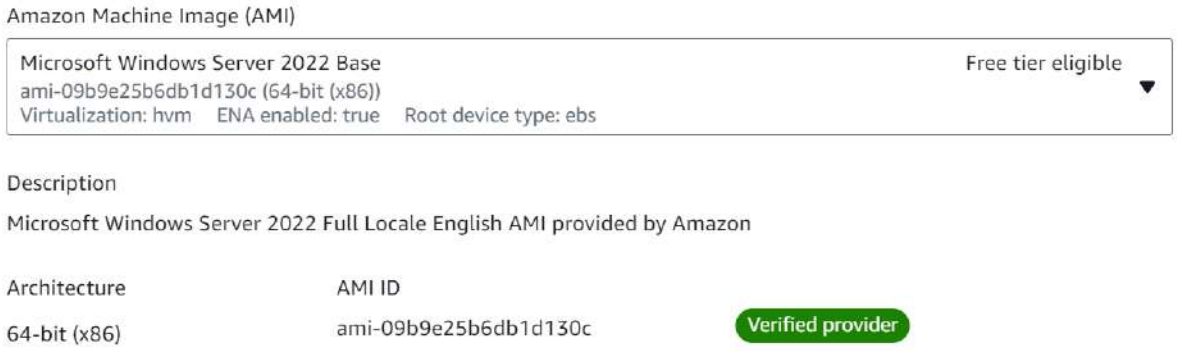


Step 2 – Click on **Launch Instance**.

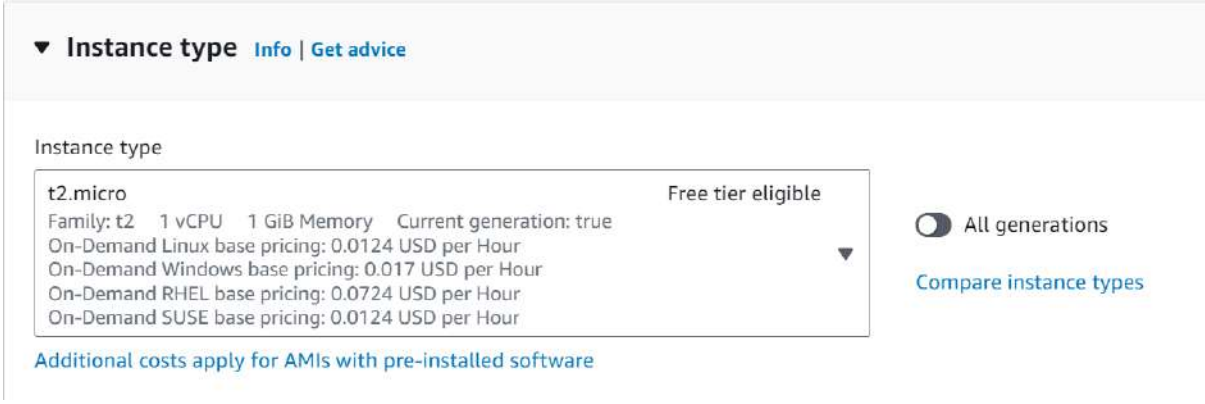


Step 3 – Now we will fill in the configurations for creating an EC2 Instance. Give a name to your Instance or remain its default.

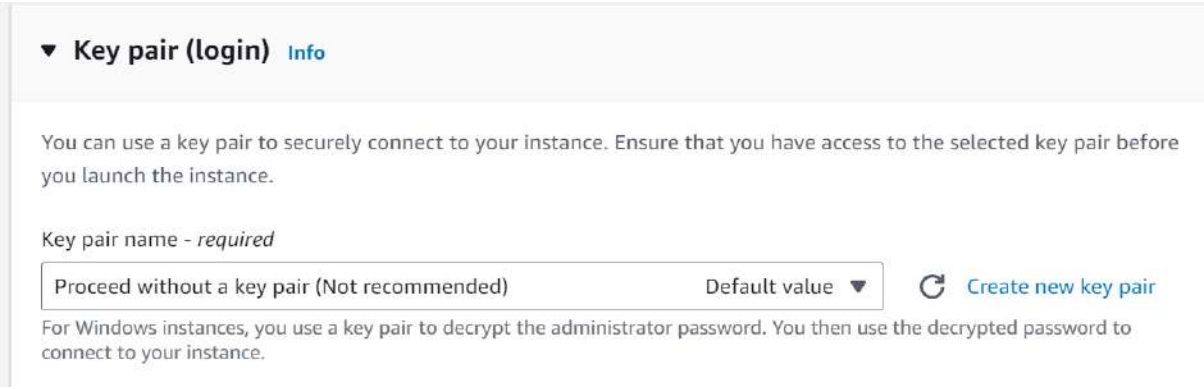
Then select the Windows AMI (Amazon machine Image). We are selecting Microsoft Windows server 2022 base.



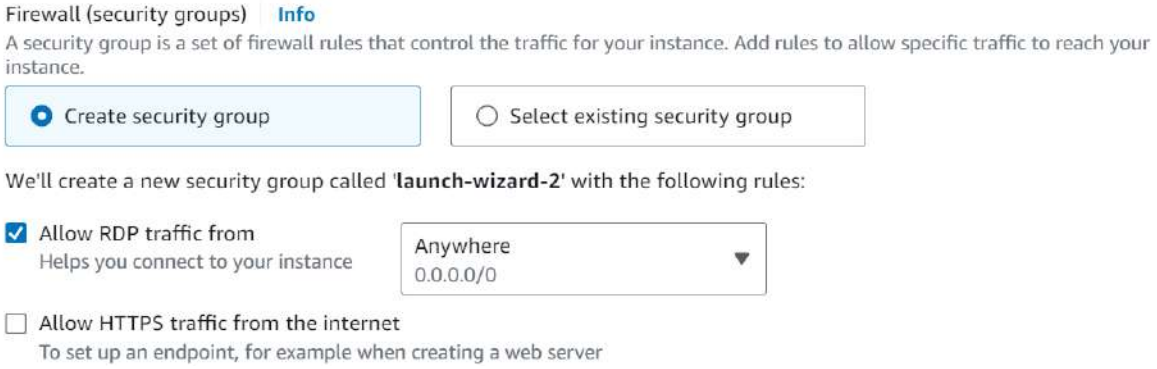
Now select the **Instance type**.



Select the key pair or create your own by clicking on **Create new Key pair**.



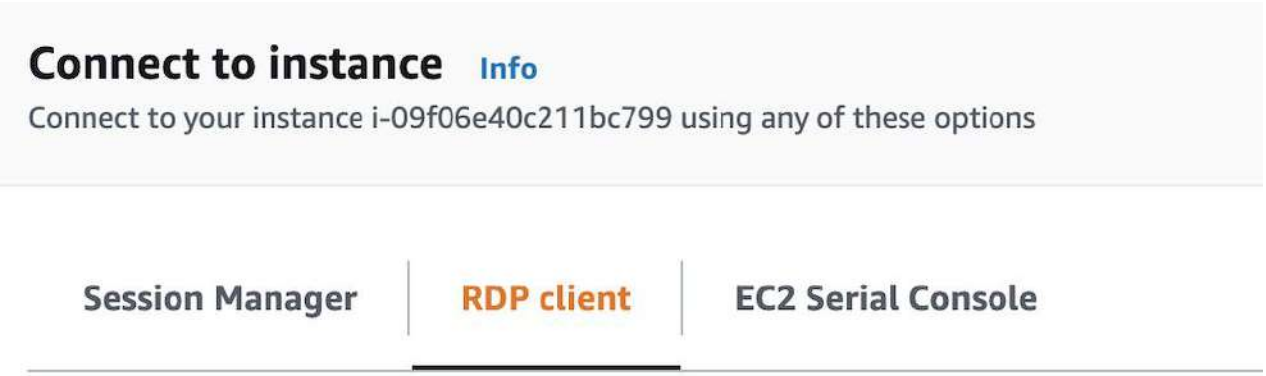
In the Network setting, make sure **RDP traffic** is allowed.



Set your storage up to your need. Now we are ready to launch our instance by clicking on **Launch Instance** on the right side.

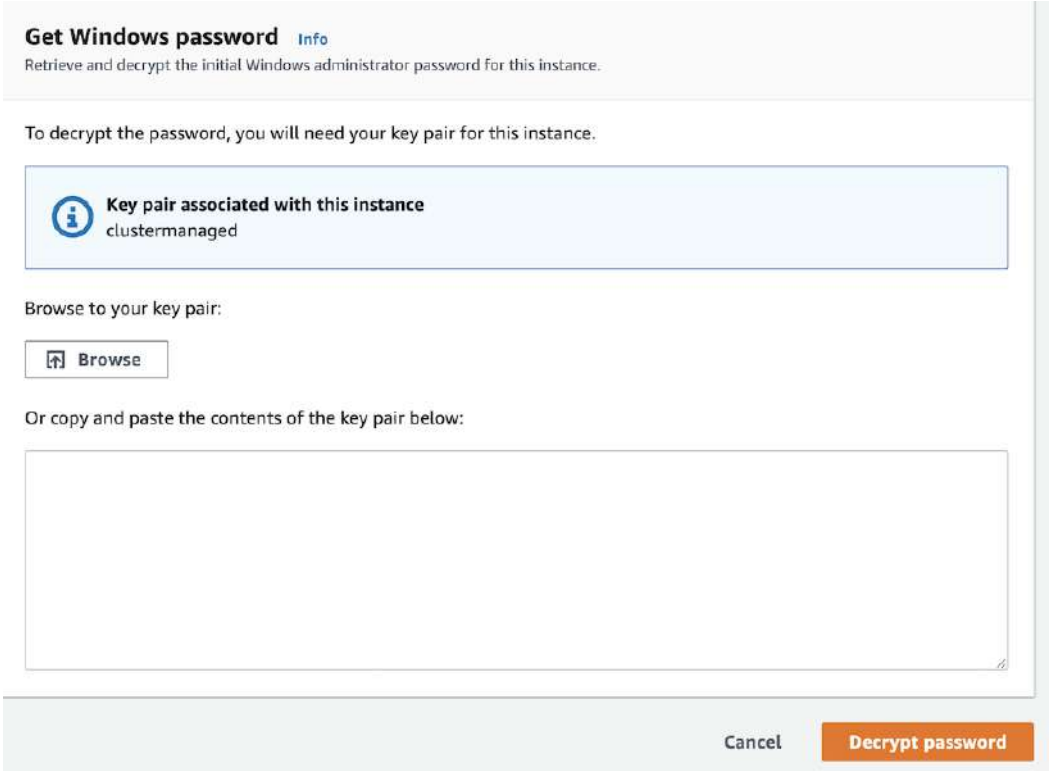
After some time your instance will be ready and in next step, we are ready to connect with our EC2 instance using RDP Client application.

Step 4 – The First download and install RDP client app to your instance list, select your instance and click on **Connect**. There are three ways to connect to the instance, click on **RDP Client**.

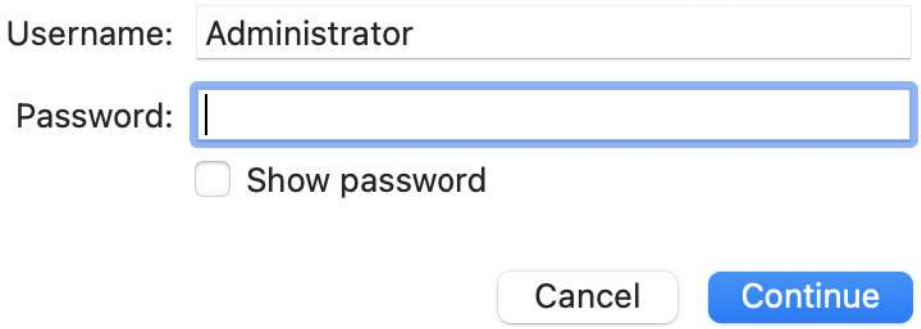


Now click on **Download remote desktop file**.

We also need a password to connect for that, click on **Get Password** and browse your key pair and click on **Decrypt Password**. After this, your password will be shown on the screen, copy it and save it somewhere.



Now double-click on the remote desktop file you downloaded earlier. Paste the password and press **Continue**. (If your RDP connection client application asks you for a username, type **Administrator**.)



You are connected to your Windows server.

Experiment 5

Aim - To study and Implement Platform as a Service using AWS Elastic Beanstalk Service

Theory -

- AWS Elastic Beanstalk

AWS offers a wide range of Storage service than can be provisiond depending on the project requirement and use case. It is a pre-configured EC2 server that can directly take up application code and environment configurations. Unlike EC2 i.e an example of IaaS, Elastic Bleanstalk is a PaaS since it allows user to use directly a pre-configured configuration. It abstracts the underlying configuration work and allows you as a user to focus on more pressing matters. It also makes deployment simpler but does not restrict developers from changing any configurations.

- Language / Framework supported by EBS

It provides a variety of platform on which you can build your app. You can design your own web app to one of their platforms and elastic Beanstalk deploys your code to the platform version you selected to create an active app environment

It provides for programming languages (Go, Java, node.js, PHP, Python, Ruby, applications server (Tom Cat, passanger, Docker containers

Elastic Beanstalk provisions resources need to run your app including one or more Amazon EC2 instance. The softwares are running on the Amazon EC2 instance demands on specific platform for your environment.

Supported Platform:

1) Docker
2) Go
3) Java SE
4) Tom Cat
5) NET Core on Linux
6) Node.js
7) PHP
8) Python
9) Ruby

Elastic Beanstalk can integrate with databases like Mysql, Oracle, and Microsoft SQL server. It also scales your application when demand increases using auto-scaling rules. Access control via AWS Identity and Access Management and built-in security features like SSL / TLS encryption are

are provided by (IAM)

Comparision between EC2 and EBS

EC2	EBS
1) Elastic Compute Cloud is service that provides resizable compute capacity by renting virtual machines and runing applications on them	1) Elastic Beanstalk simplifies the deployment, management, and scaling of web applications and services by auto handling infrastructure
2) Users have full control over instances they launch like manging OS, runtime, etc	2) Abstracts away underlying infrastructure by handing load balancing, auto scaling etc.
3) Manual intervention or using Auto scaling services for scaling instances	3) Provide inbuilt auto-scaling to automatically scale number of EC2 instances
4) High degree of flexibility and control in terms of choosing OS, instance types	4) Limits customization options due to abstraction but supports multiple languages and frameworks
5) Needs user to manually provision, configure and manage virtual machines	5) simplify deployment by providing a guided experience through console or CLI
6) Users are billed for EC2 instances based on usage and type selected.	6) Helps optimize costs by on demand scaling ie better resource utilization

Elastic Load Balancing

Elastic Load Balancing (ELB) is an AWS service that automatically distributes incoming application traffic across multiple targets like EC2 instances, containers, IP addresses and lambda functions. It plays a crucial role to ensure high availability, fault tolerance and application scalability.

Some key features include-

1) Distribution of Incoming Traffic

2) Health checks

3) Load Balancer Types
   a) Application Load Balancer (ALB)
   b) Network Load Balancer (NLB)
   c) Classic Load Balancer (CLB)

4) SSL / TLS Termination

5) Integration with Auto Scaling

6) Logging and Monitoring

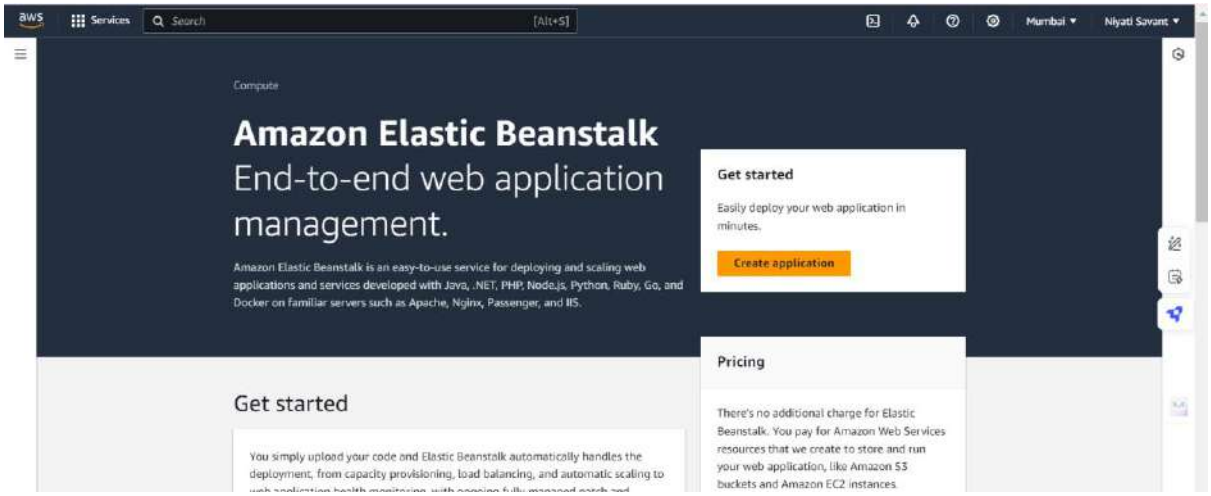
15/3/24 (A)

Experiment 5

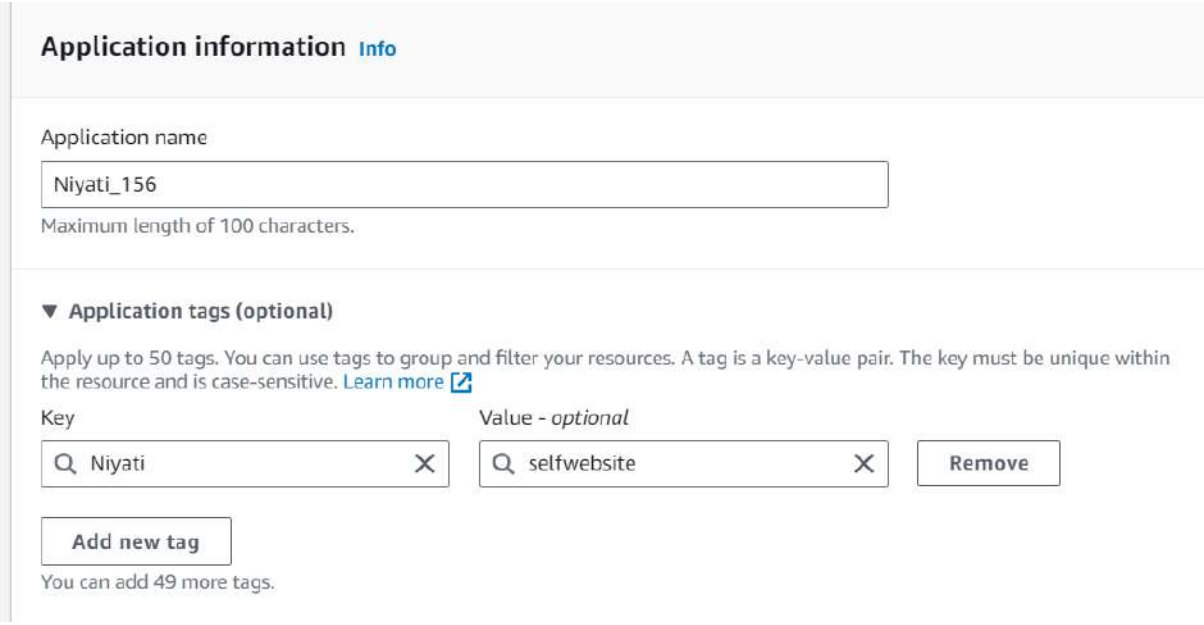
To study and Implement Platform as a Service using AWS Elastic Beanstalk Service.

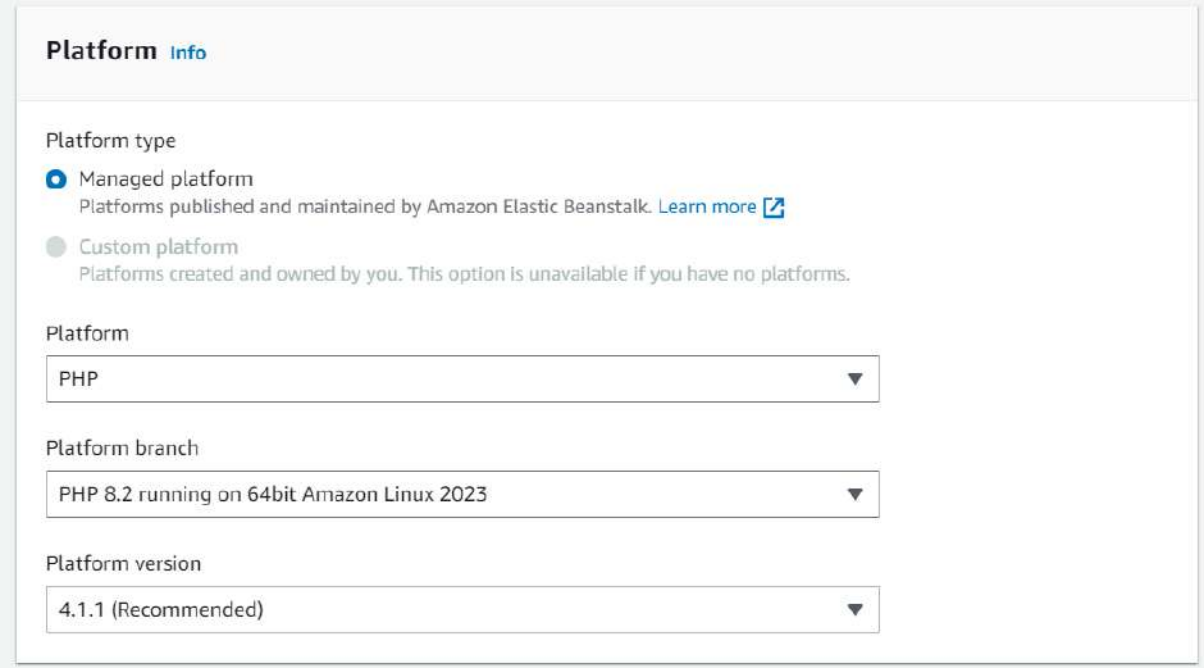
Step1 : Login to AWS console and go to Elastic Beanstalk.

Step 2: Click on Create Application



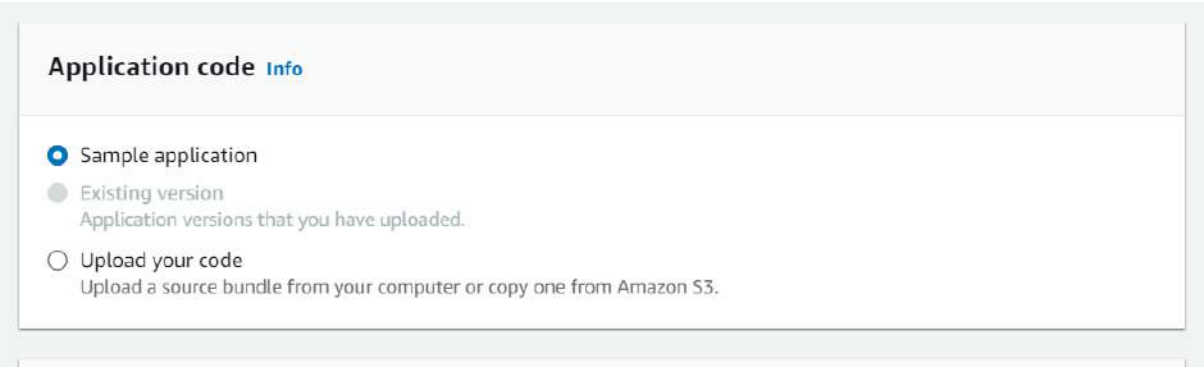
Step 3: Write Application information : Name, Tag,Platform etc.

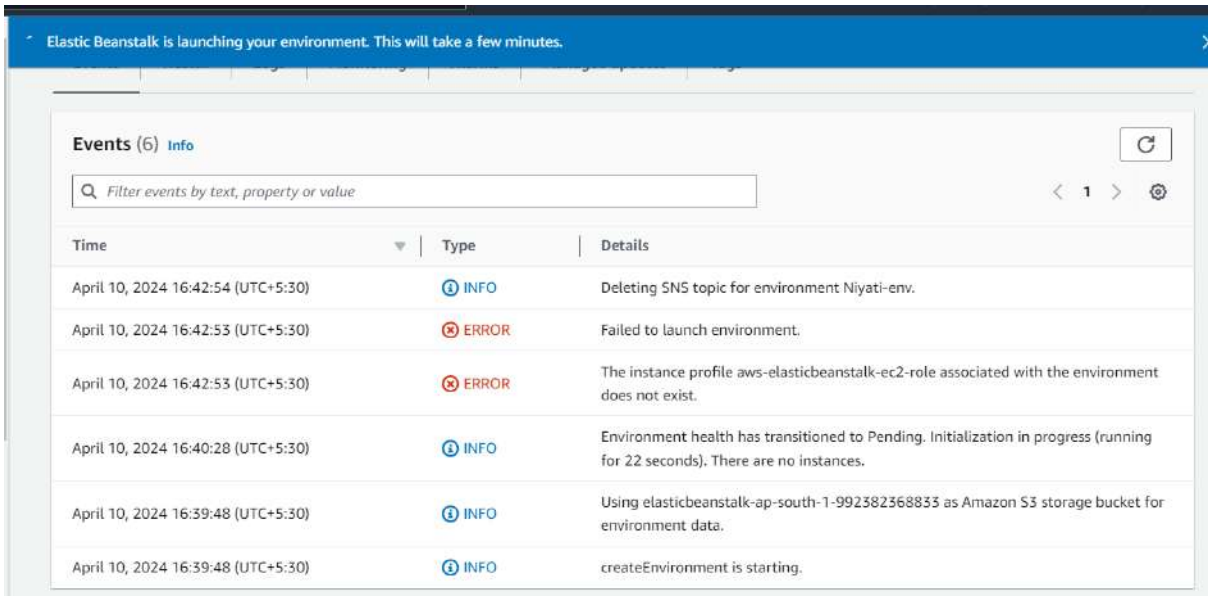


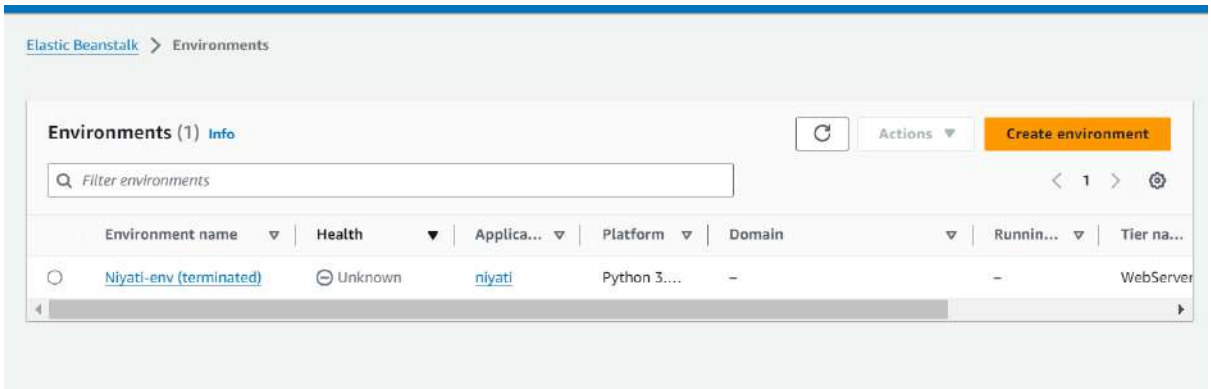


Step 4: In Application Code: select sample application and then Click on button Create Application

This will take a few minutes.







Step 5: Click on Environments -> Check the health of Environment wait till it becomes 'OK'

Step 6: Click the URL

To Delete the application and Environment (Select it and in Action -Delete/Terminate : giv

Experiment 6

Aim - To study and Implement Storage as a service using own Cloud

Theory.

Cloud Storage is made of computer data storage in which digital data is stored on server in off-side locations. The servers are maintained by a third party provider whose is responsible for hosting, managing and security data stored on its infrastructure.

It enables organizations to store, access and maintain data so that they don't need to own and operate their own data center moving expenses from a capital expenditure model to operational one.

Own Cloud and features

It is a self-hosted, open source file syncing and sharing service like dropbox, google drive, box, and more for, own cloud gives user access to ~~ete~~ files, calander, contacts and other user data. User can sync between ~~to~~ different devices and store files with others

Features -

1) Access your Data
2) Sync and share data
3) Encryption
4) Versionising
5) Drag and drop upload
6) Theming
7) Viewer for PDF files
8) API applications
9) Migration and Backup
10) Tasks
11) Application store

Advantages -

1) Cost efficient
2) Scalability
3) Accessibility
4) Automatic Updates
5) Collaborator
6) Disaster Recovery
7) Flexibility
8) No physical infrastructure
9) Managed Security

Limitations -

1) Dependency on Internet
2) security Converts
3) Data Transfer Speed
4) services downtime
5) Customization Constraints
6) Potential for vendor lockin
7) Regulatory Compliance
8) Cost predictability
9) Limited Control
10) Data Location

Storage Types

1) Object Storage

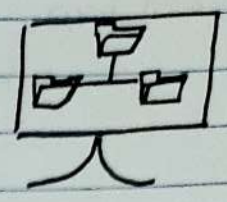
Transport : TC/IP
Interface : HTTP / REST
Use Case : Easily scaled with no limits accessible across LAN and WAN

Store data as objects ie contain actual data and associated metadata

2) File Storage

Transport : TCP/IP

Interface : NFS / SMB
Use Case : Good performance, file sharing, global file locking



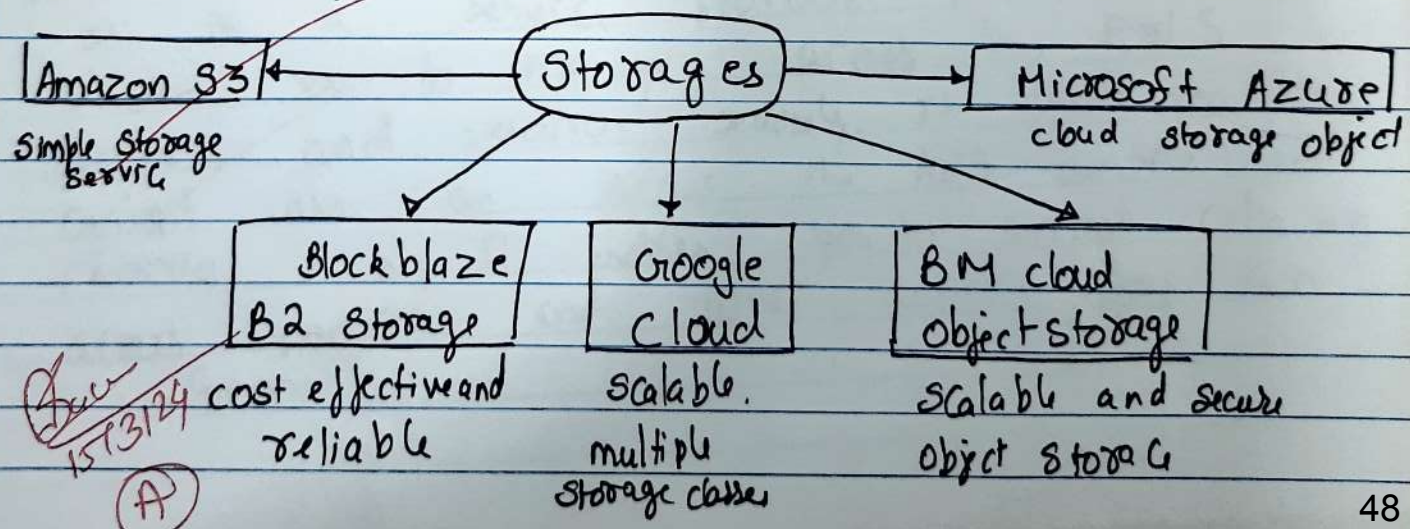
A hierarchical storage system that provides shared access to file data. Use remote infrastructure of servers.

3) Block storage

Transport : FC / iSCSI
Interface : Direct Attached or SAN
Use Case : Low Latency. Best for structured Data

Works by dividing data into fixed-sized blocks and storing them as individual units. Allows direct access to individual blocks.

Popular Storage as service vendors

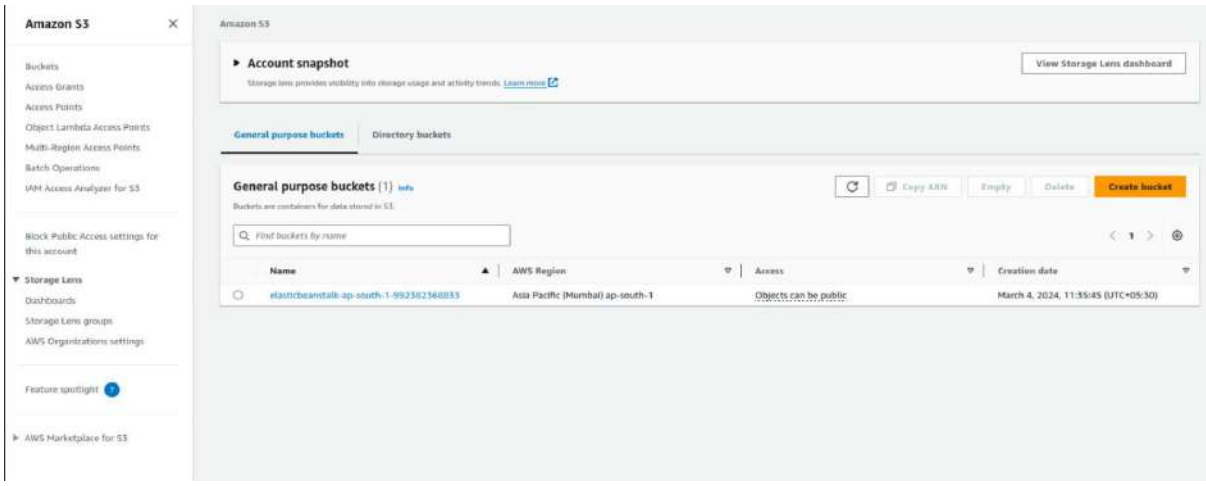


15/3/24

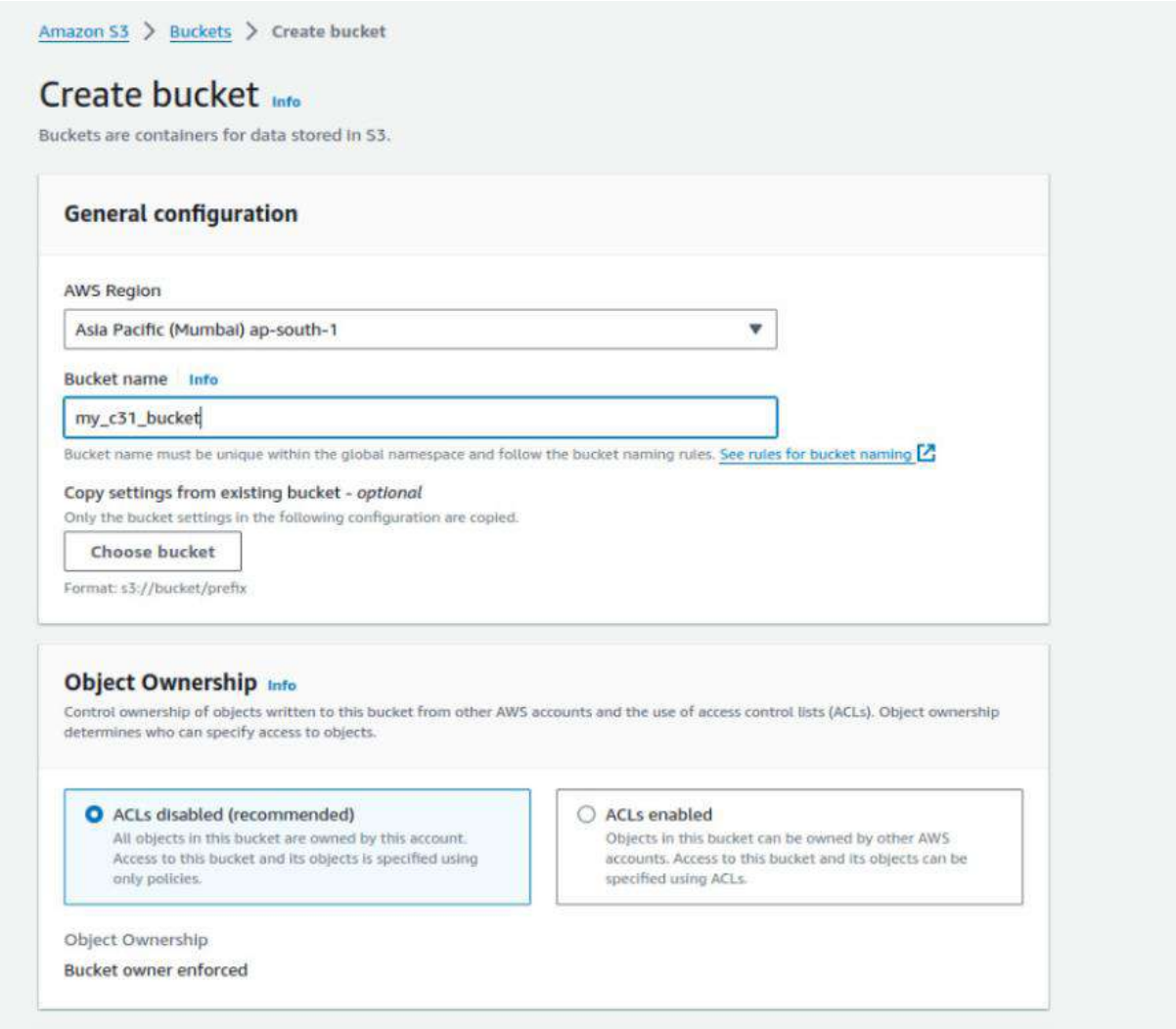
Experiment No-6

To study and Implement Storage as a Service using Own Cloud/ AWS, Glaciers

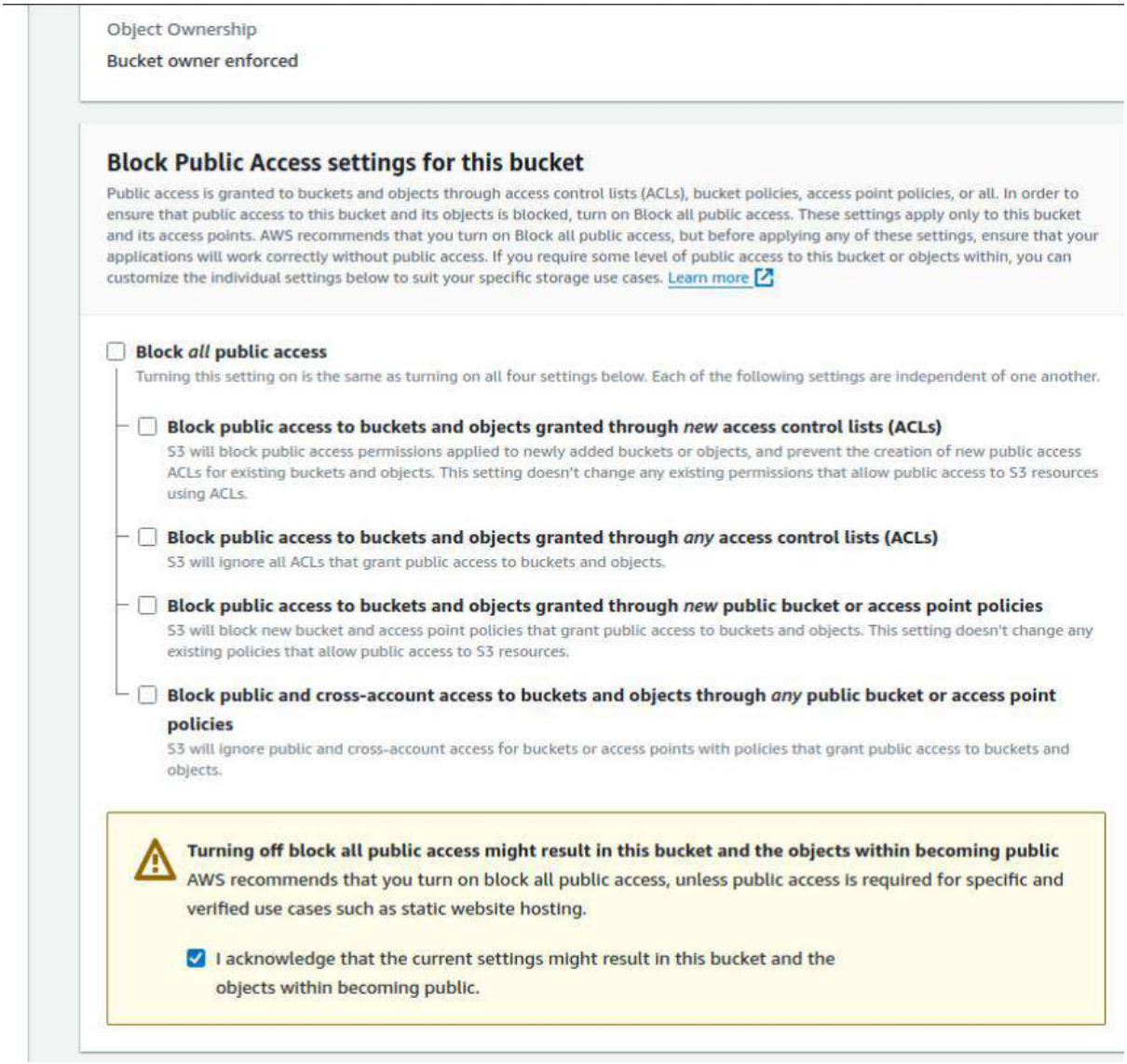
Step-1: Click on create bucket



Step-2: Give Bucket name & select region for storage



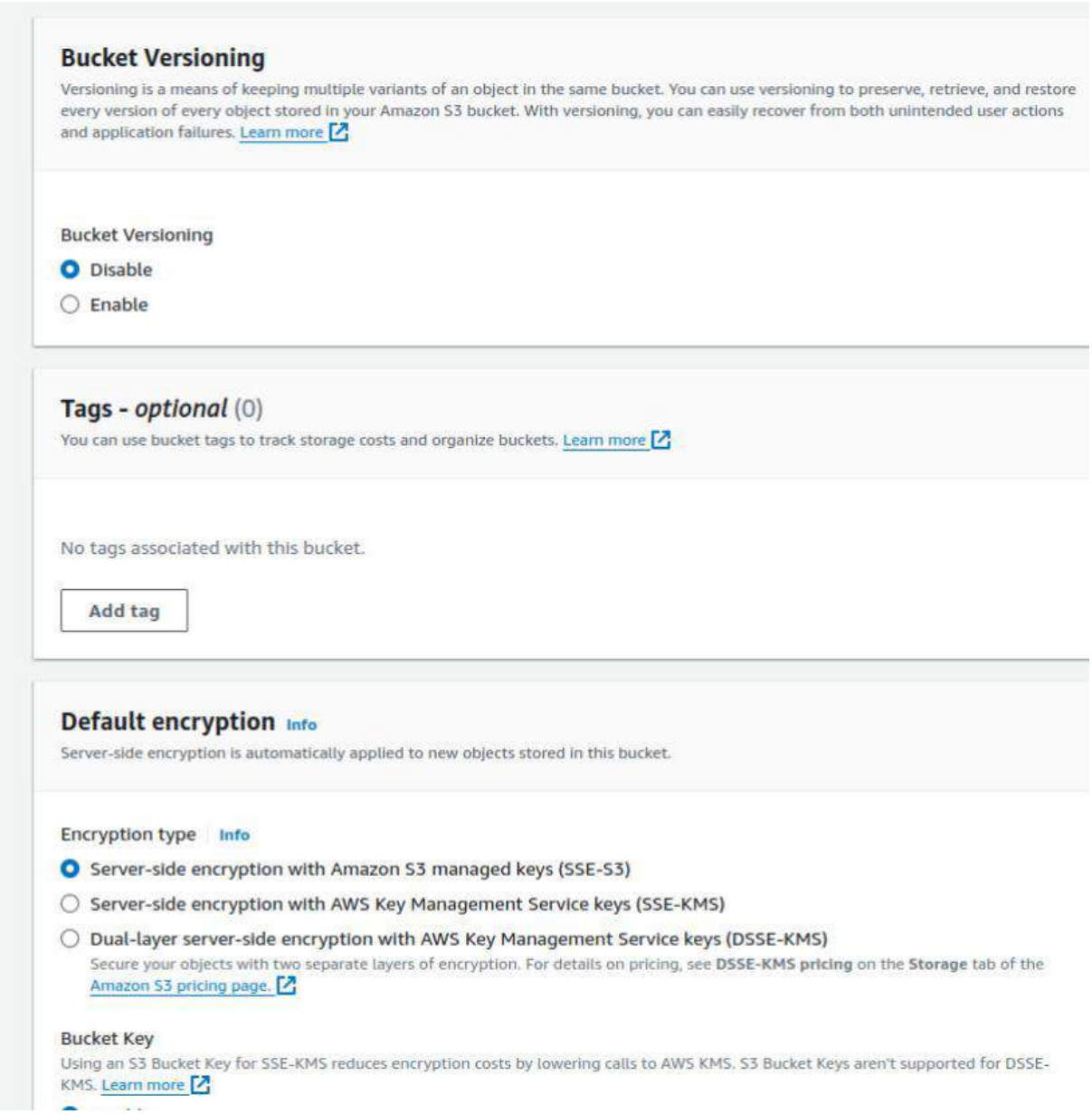
Step-3: Keep object ownership setting as ACLs Disabled as by-default



Step-4: Disable block all public access checkbox

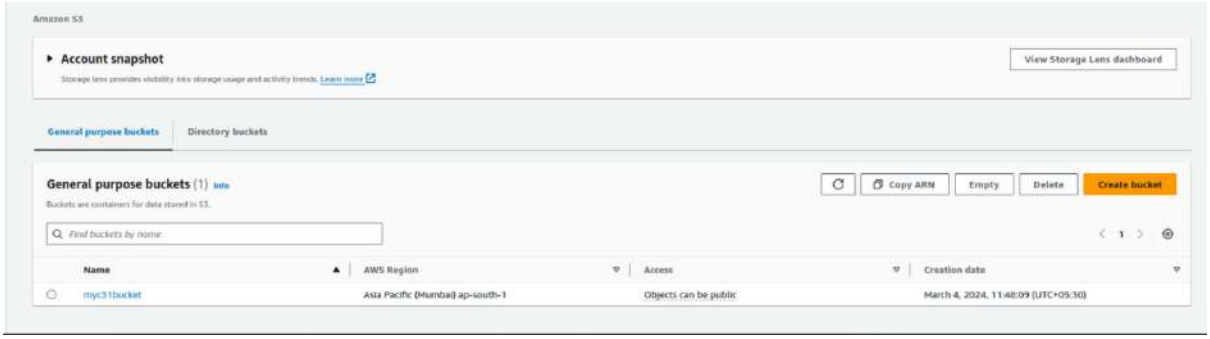
Step-5: Select the checkbox for Turning off block all public access might result in this bucket and the objects within becoming public

Step-6: Keep bucket versioning as disabled and add tags if required.

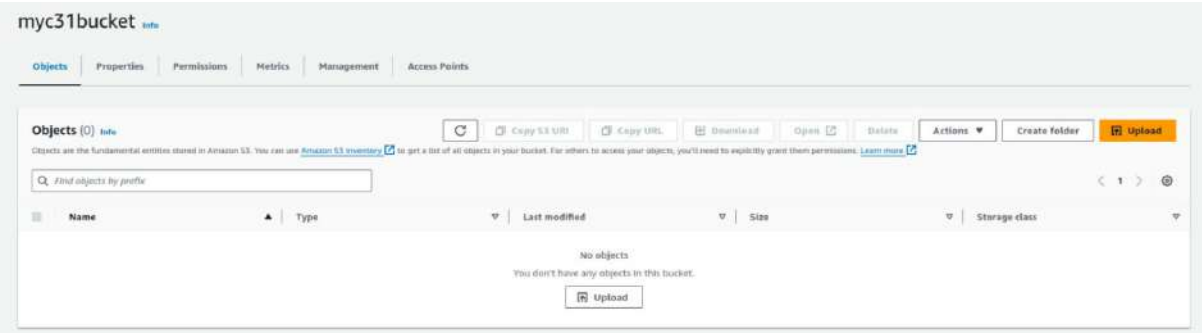


Step-7: Keep default encryption disabled and click on create bucket button

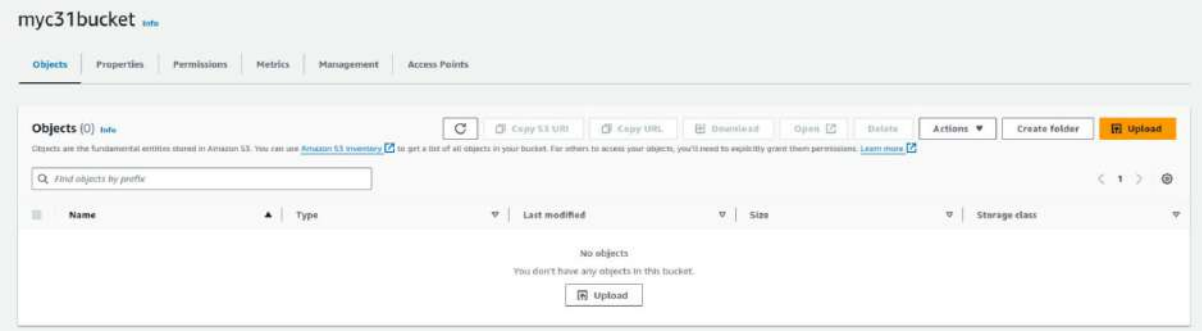
You can now see the successful creation of your bucket



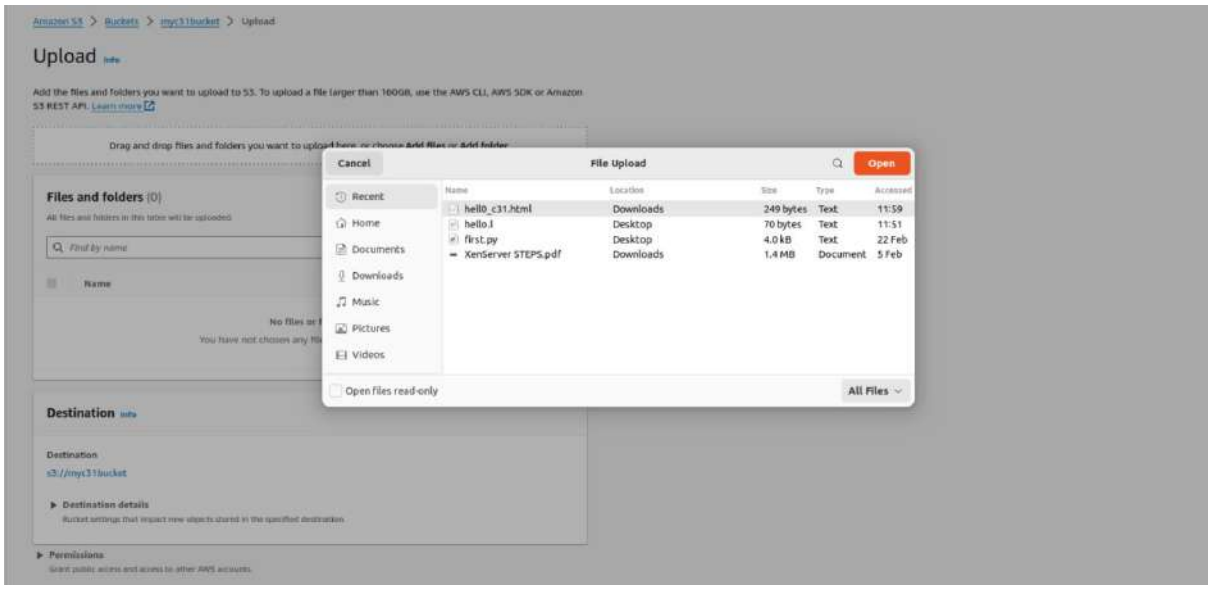
Step-8: now click on the bucket that you have created



Step-9: You can either create a folder here or upload an existing file in the bucket

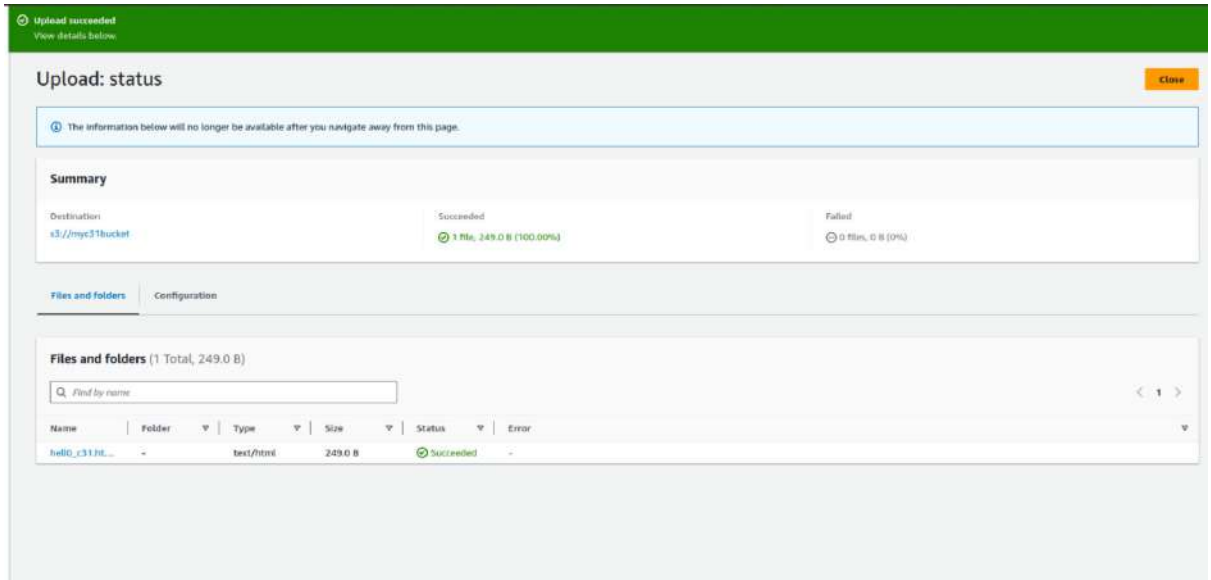


Step-10: now click on upload button and click on add files button browse your local machine and



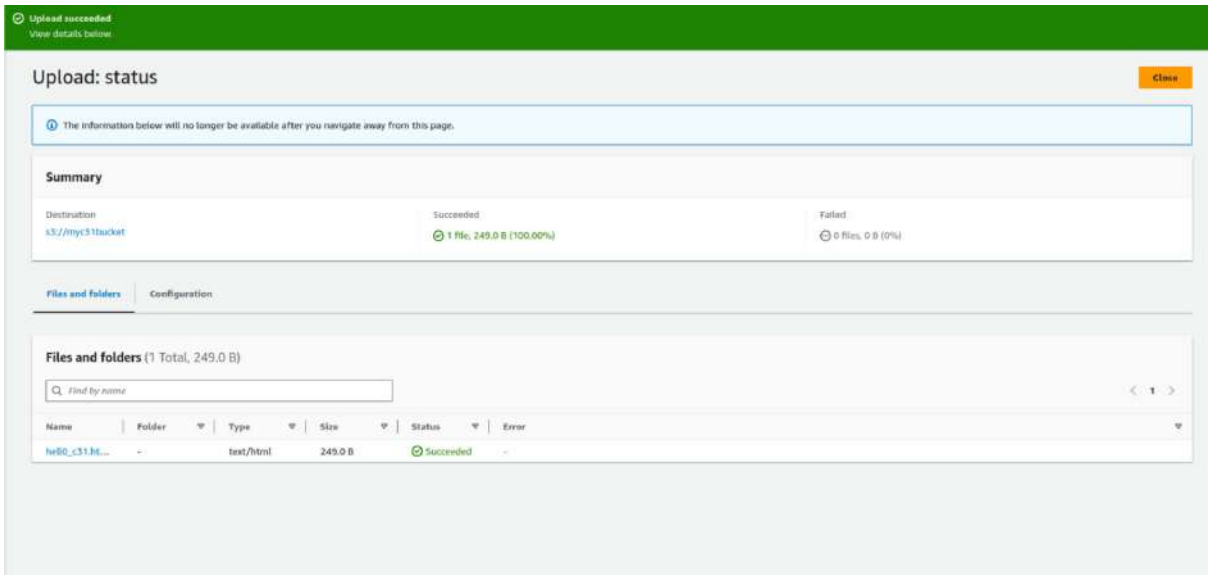
select which file you need to upload on S3 next click on upload button at bottom right end

Now you can check the upload status screen

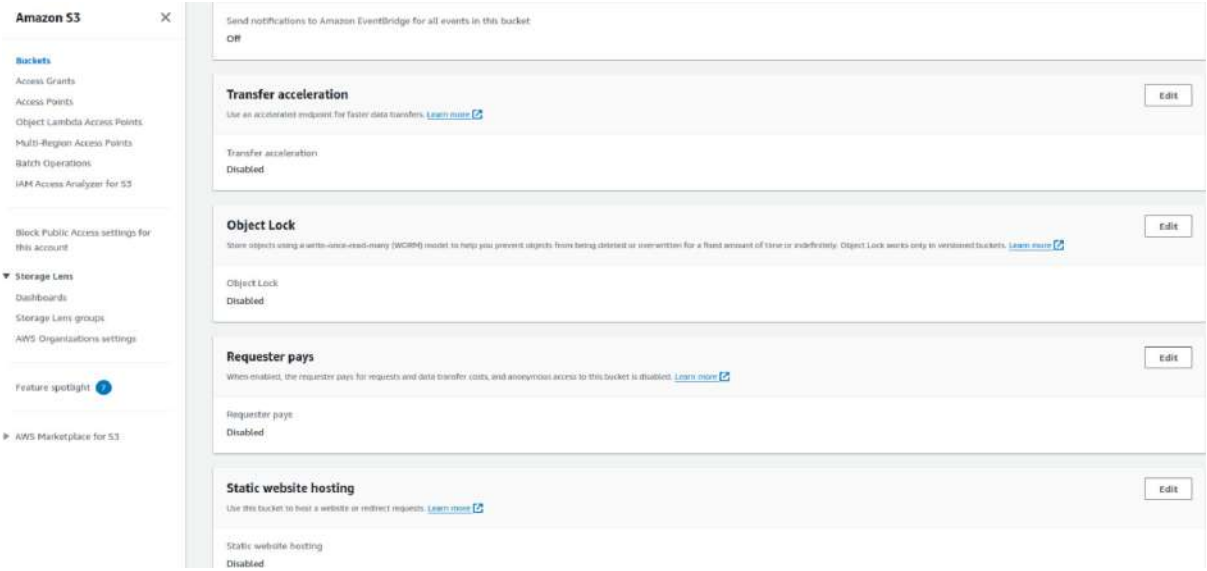


Now click on close button

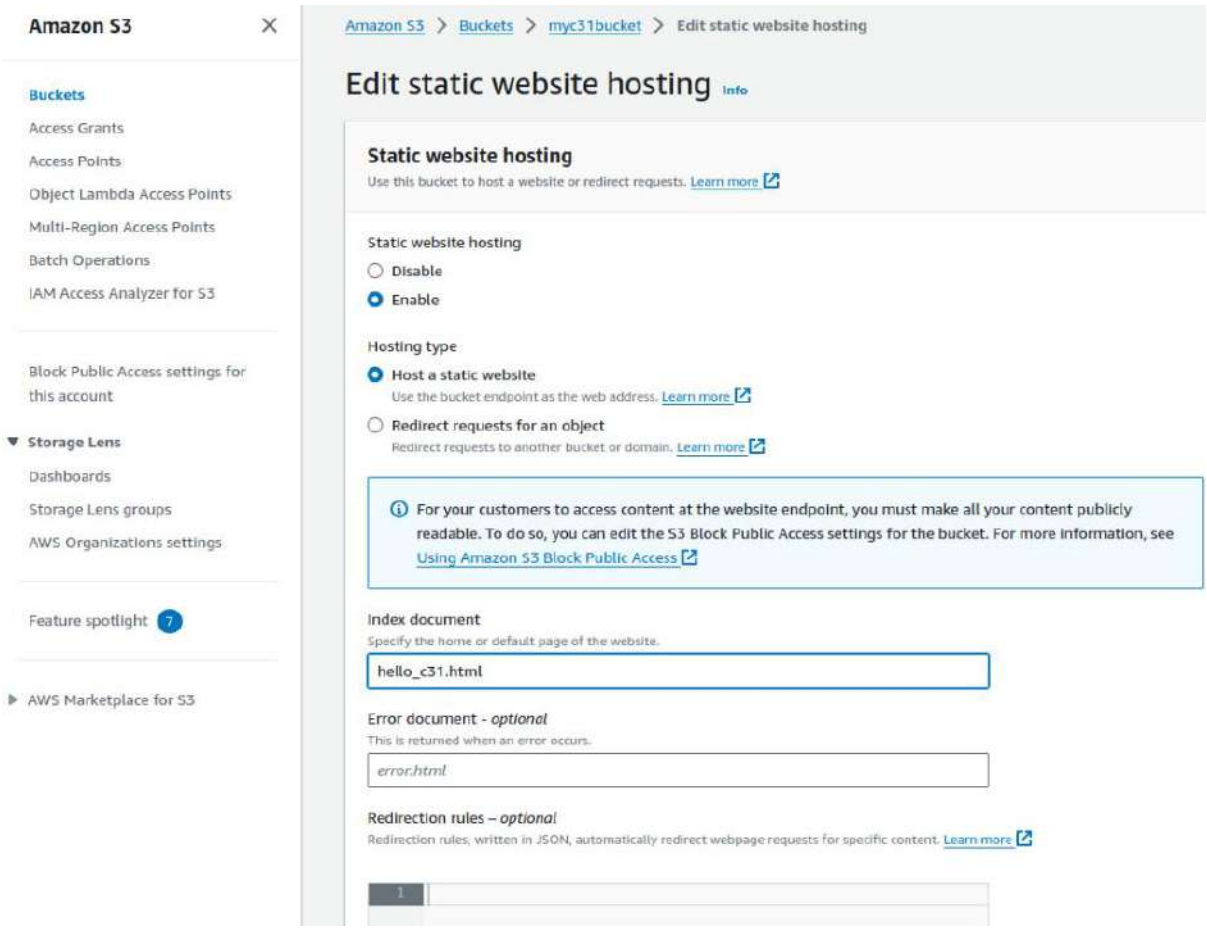
The screen will appear as below



Step-11: Select properties and scroll down to Static website hosting option which is disabled now click on Edit option on right side

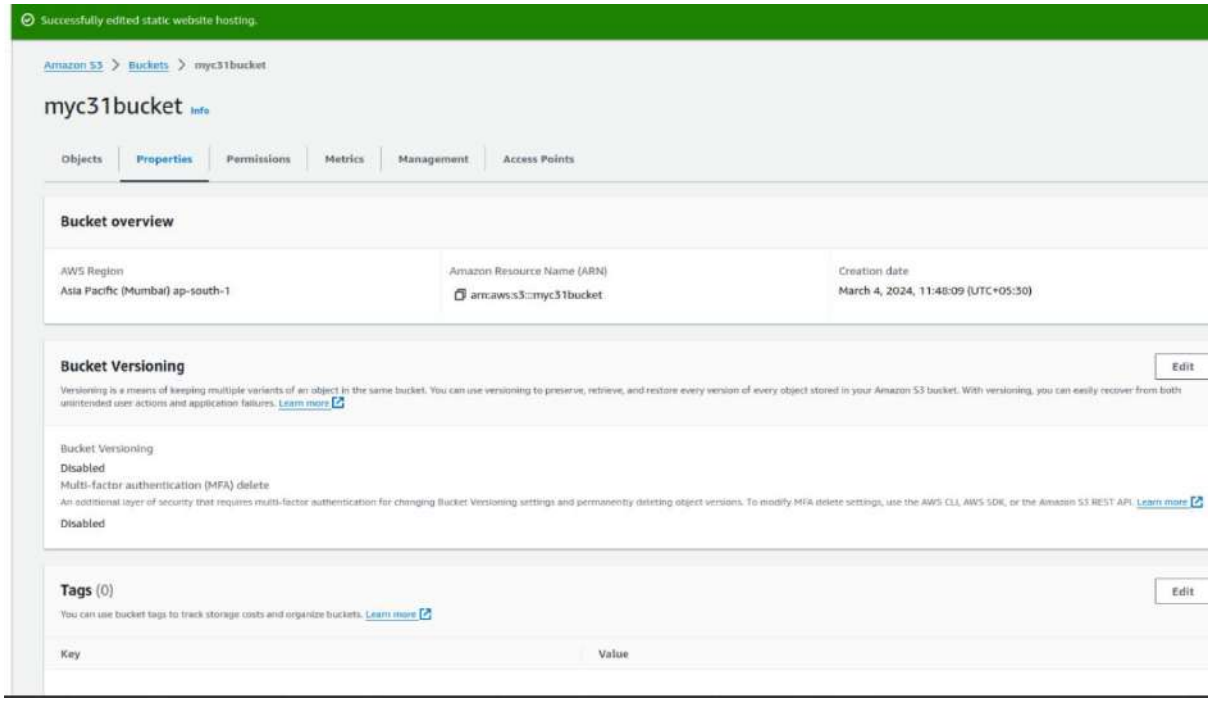


Step-12: Enable the radio button and specify the file name in Index document which you have added in S3



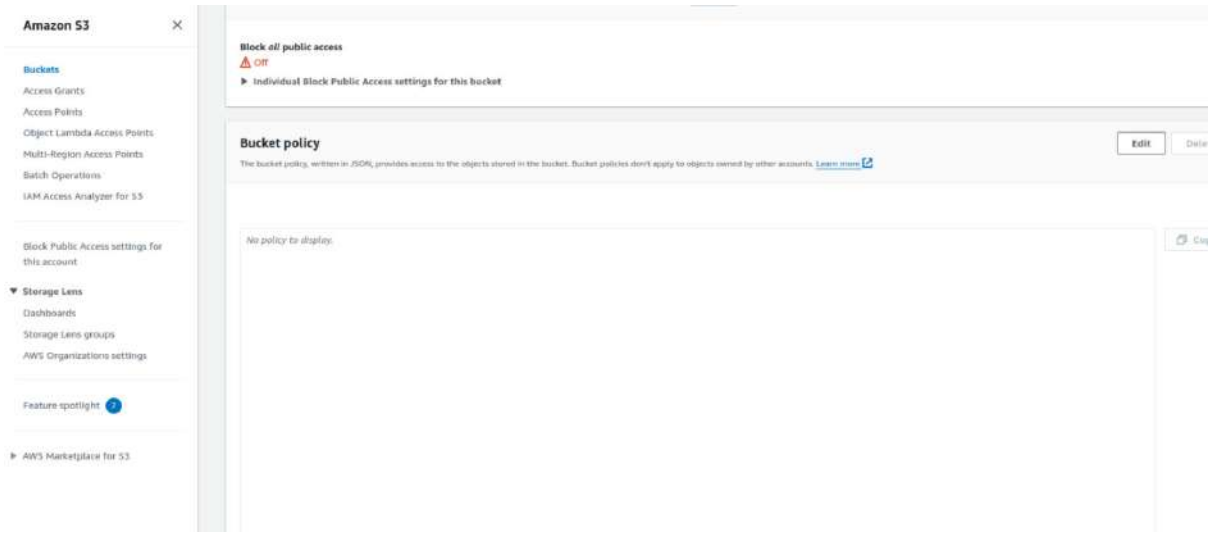
Scroll down and save the changes at bottom right

Following screen will appear

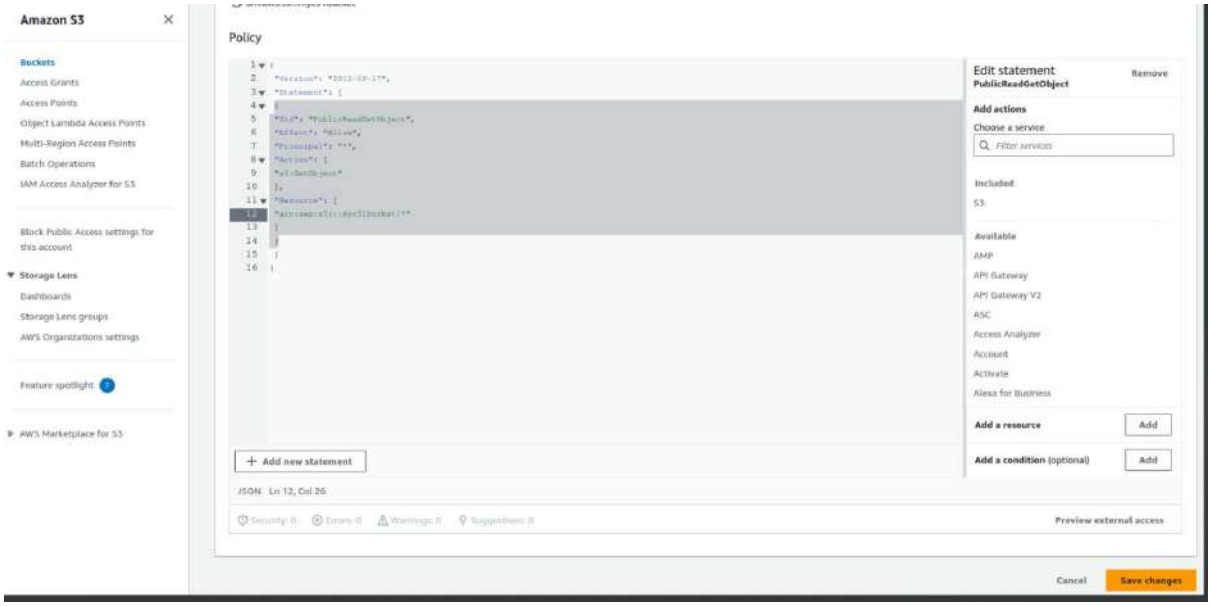


Step-13: Click on Permissions Tab

Step-14: In bucket policy click on Edit option



Step 15- after clicking on edit button paste the following code in bucket policy

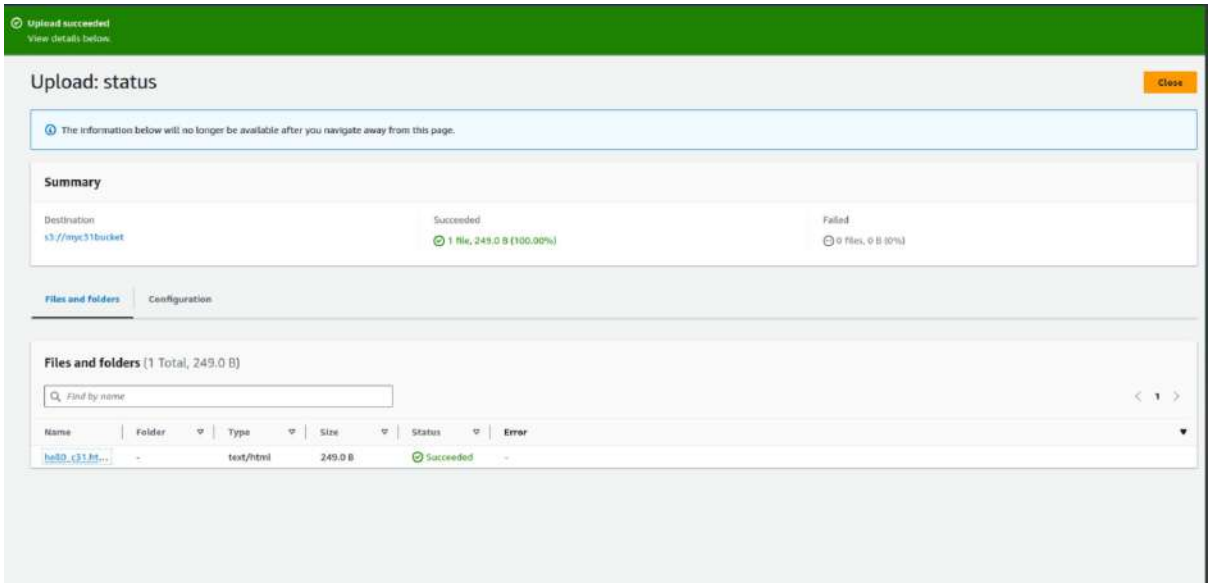


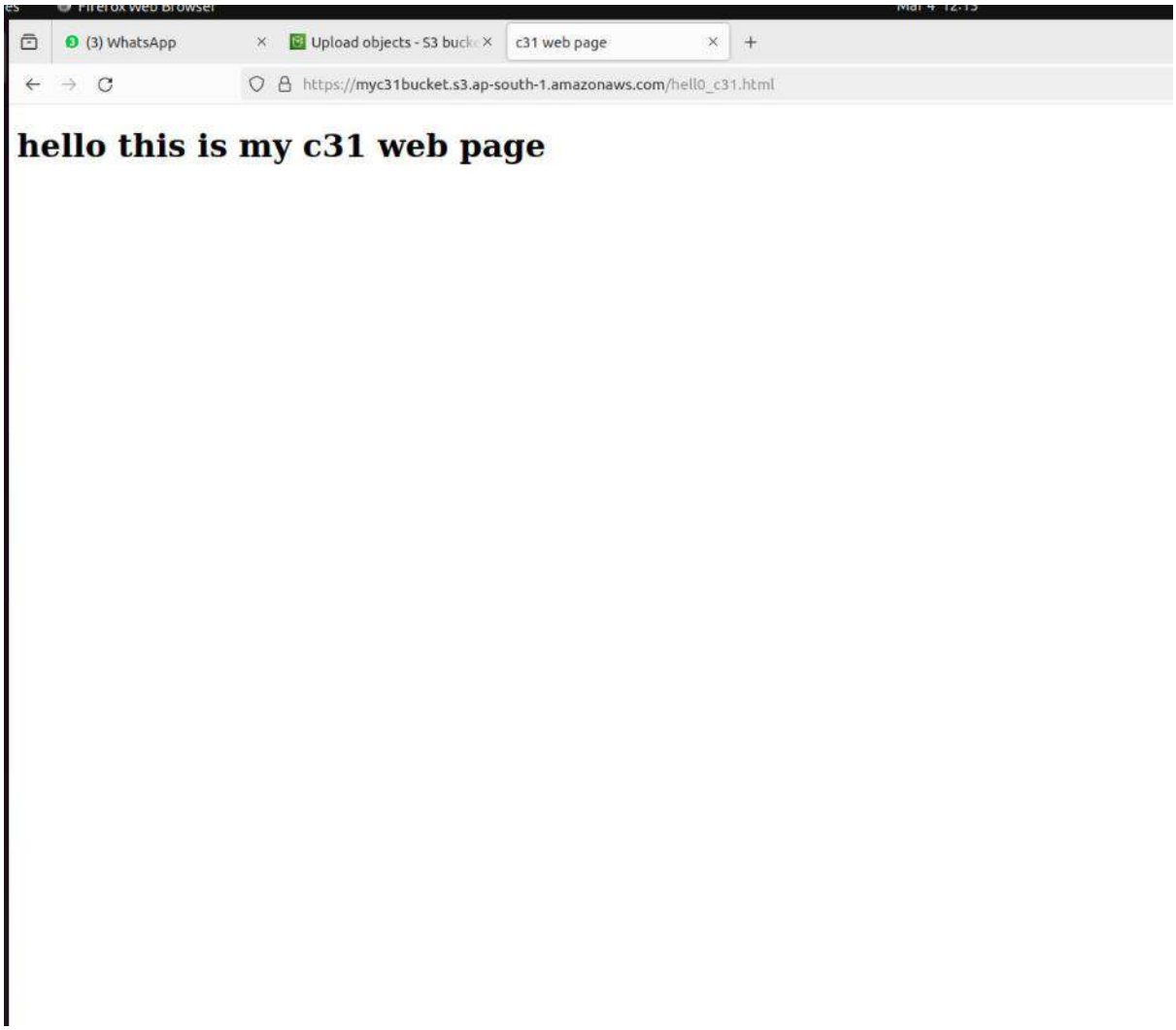

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Sid": "PublicReadGetObject",
 "Effect": "Allow",
 "Principal": "*",
 "Action": [
 "s3:GetObject"
 ],
 "Resource": [
 "arn:aws:s3:::Bucket-Name/*"
 ]
 }
 ]
}
```

Note-Make sure that you add your bucket name in the code above

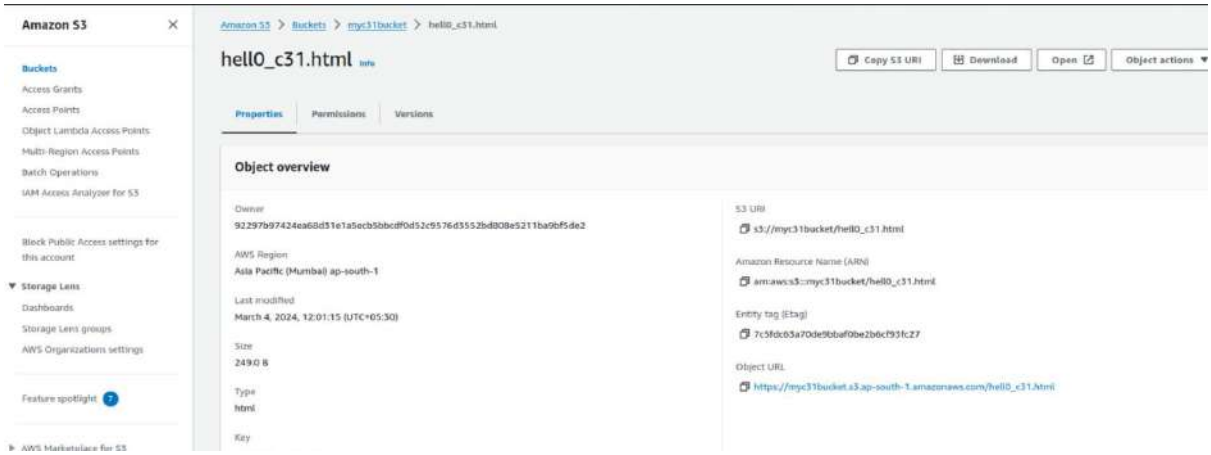
Scroll down and click on Save Changes button

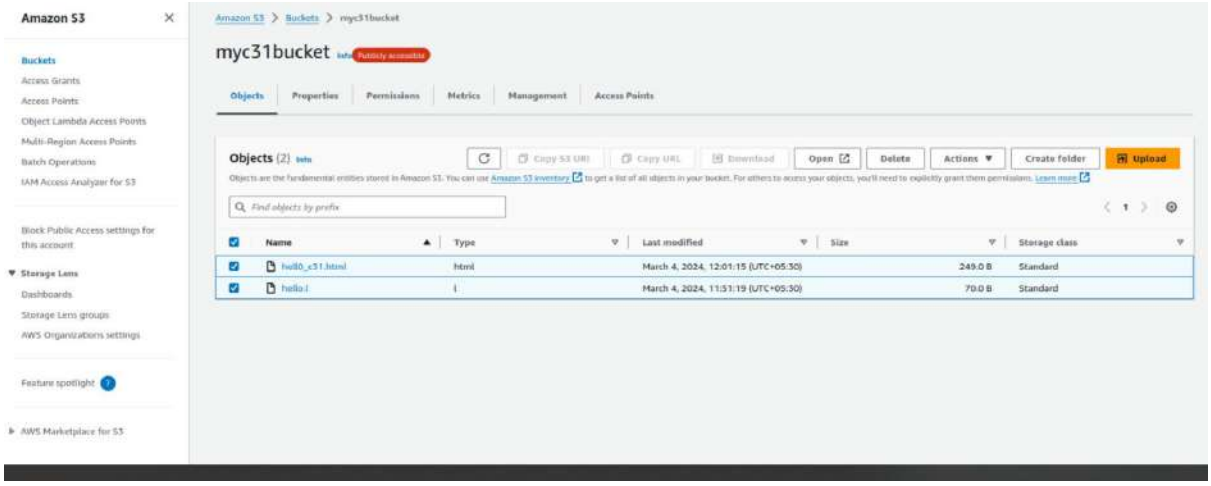
Step-16: open your html file and click on Object URL



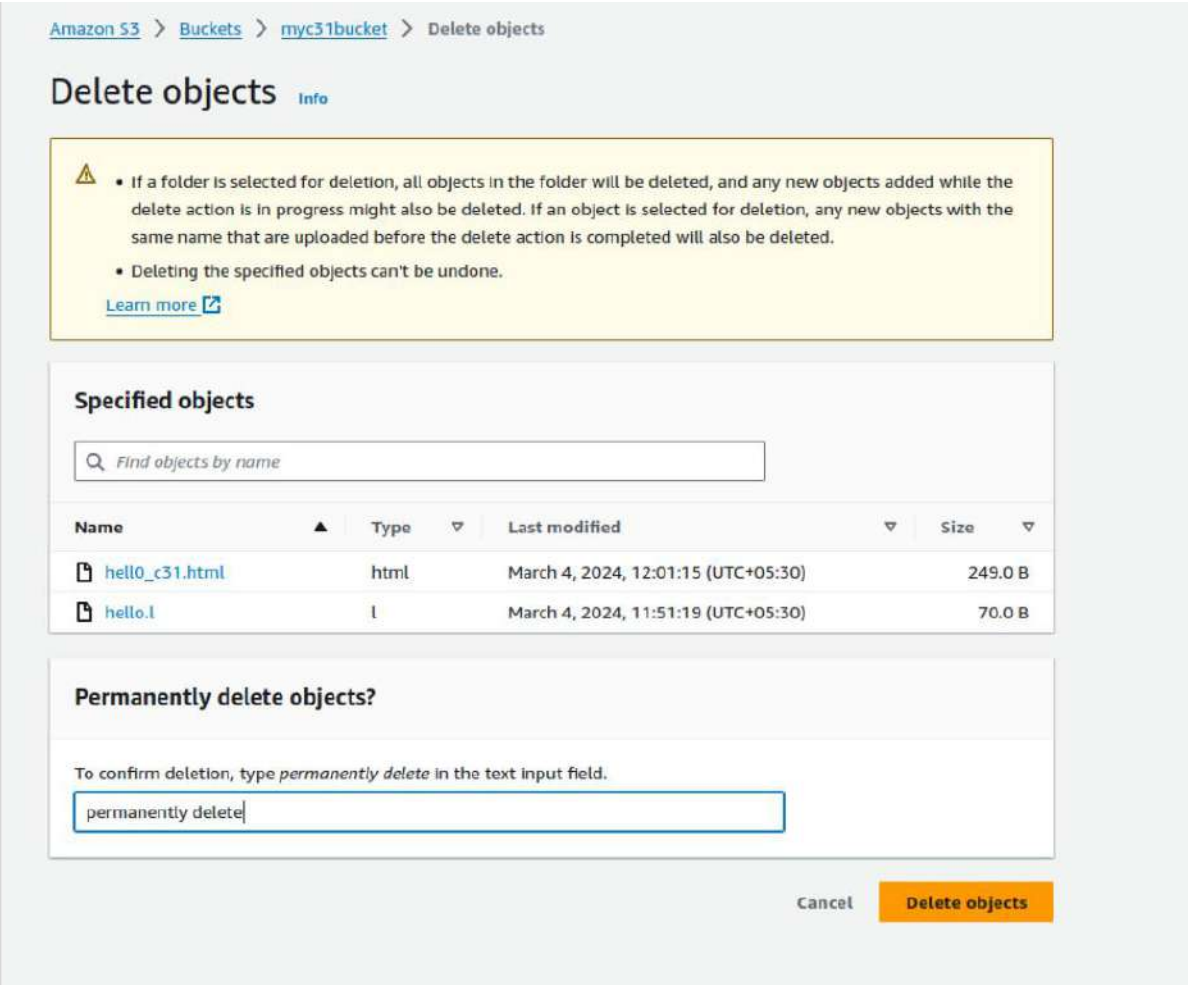


Step-17: Now for delete files click on checkbox of your file and then click on Delete Button

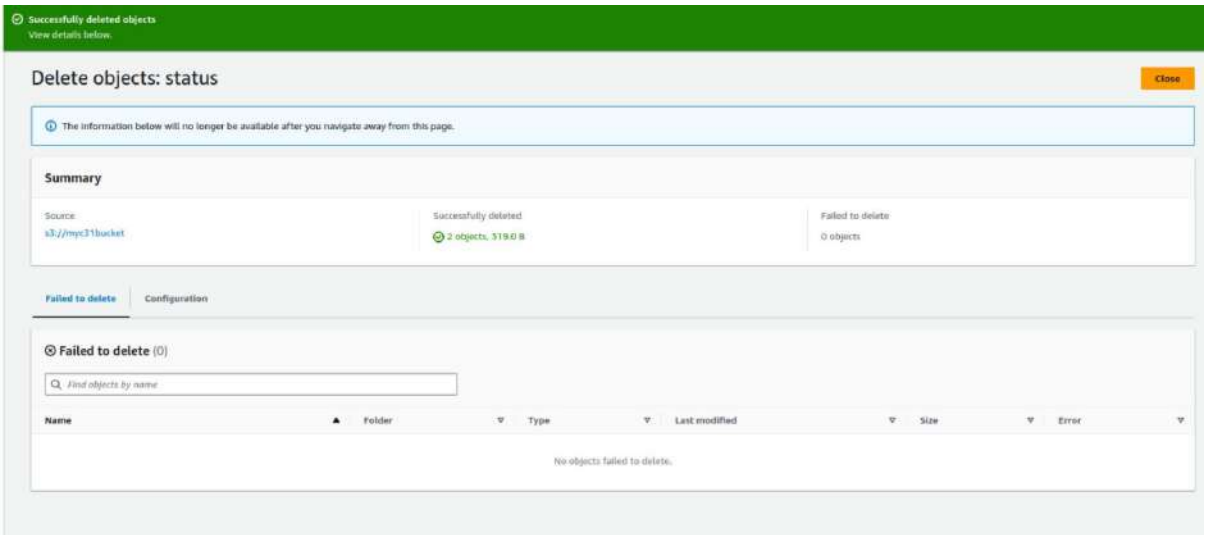




Write permanently delete and click on delete object button

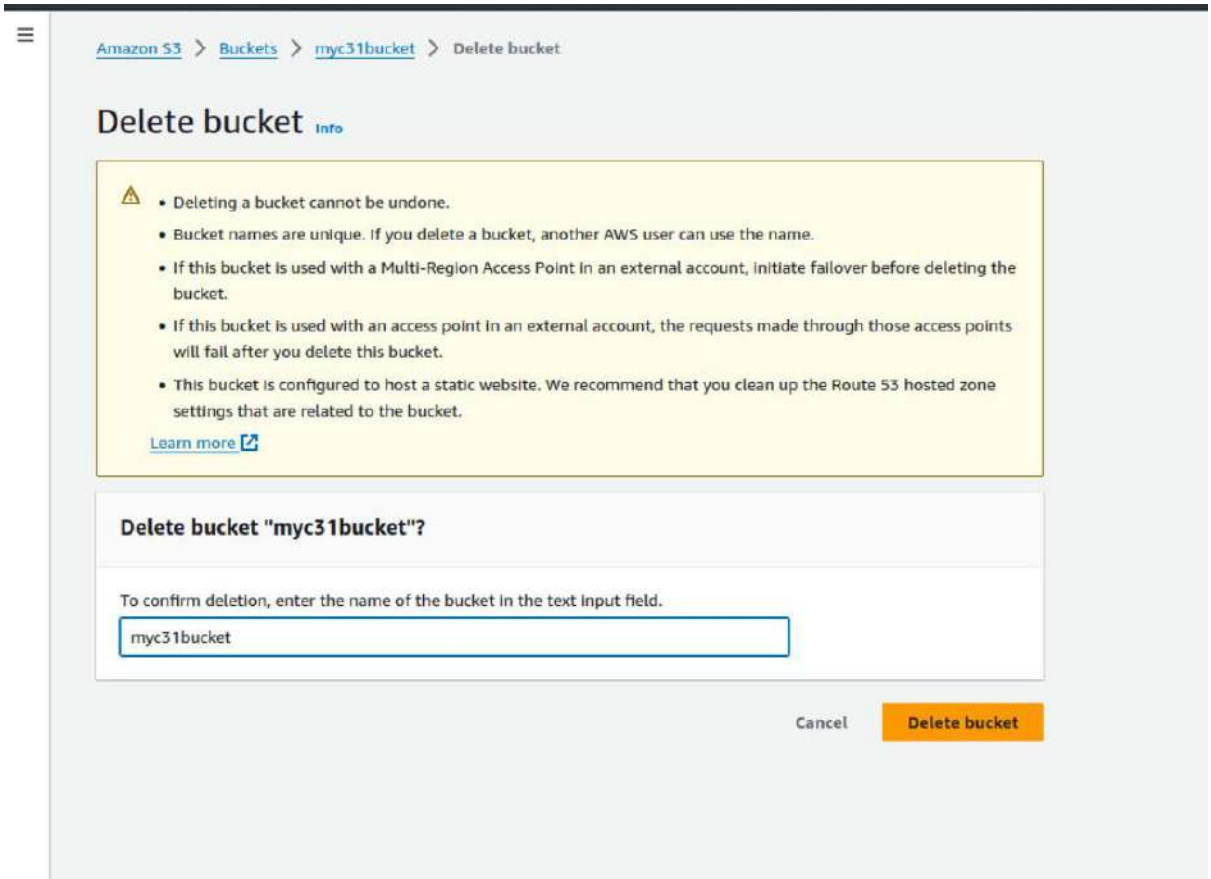


Now click on close button



Step-18: now come to Amazon S3 tab and select your bucket and then click on delete button

Write down your bucket name in delete bucket tab and click on delete button at bottom right



You can see that the bucket is deleted

Niyati Savant
TE - C31

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

2103156 Experiment - 7

Aim - To study and implement Identity and Access Management (IAM) practices on AWS

Theory -

Access Management

- It is the process of controlling and managing who has access to what resources within an organization's IT infrastructure. It includes physical resources within an organizations IT like files, databases and application
- It helps prevent unauthorised access to sensitive information and resources
- Limiting access to critical systems and data reduces potential impact of security incidents and mitigate risk.
- By granting employees access to needed resources to effectively do their job can improve workflow efficiency and productivity.

IAM (Identity & Access Management)

It is a service provided by AWS that allows you to manage access to AWS services and resources securely. It enables you to control who can access the AWS services and resources securely. It enables you to control who can access resources and what action they can perform

Niyati Savant
TE C 31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

The main components of IAM are.

1) Users :- Individual entities within the AWS account that interact with AWS resources. We can create, and manage users, assign them permission & control their access

2) Groups - Collection of IAM users instead of assigning permission to individual users, we can create group and assign permission to groups. They can be added or removed from groups as needed.

3) Policies -
JSON document that define permision and specify what actions are allowed or denied on AWS. Follow the principle of least privilege

4) Roles -
Similar to users but are not associated with a specific identity. AWS allows them to access resources securely without hard coding credentials

Niyati Savant
TE - C31
2103156

	Root Users		IAM users
1	Initial user created when signing up for AWS	1	Created within account to represent individual entities interacting with AWS
2	Has complete access to all AWS resources & services by default	2	Do not have access to resources by default: permission must be granted.
3	Root user credentials should be stored securely & used for only initial setup	3	Permission managed through IAM policies, allowing for granular control

Roles	Policies
Similar to IAM users but not associated with specific user	JSON documents that define permissions and specify allowed or denied actions on AWS resources
Meant to be assumed by users, applications or AWS services temporarily to access resources	Can be attached to users, groups, roles or resources

Nivati Savant
TE - C31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

MFA (Multifactor Authentication) in AWS

- Adds an extra layer of security to user authentication by requiring users to provide two or more forms of authentication before they can access account
- Can be enabled for individual IAM users within account. Users are required to set up MFA devices.
- When user attempts to sign in the AWS management console or access APIs, prompted to provide username, password or one code from MFA device
- Code is time based OTP generated by MFA device and synchronised within authentication service.
- MFA can be required for specific API operations or API that calls that perform sensitive action, adding an extra layer of security & prevent from data breaches

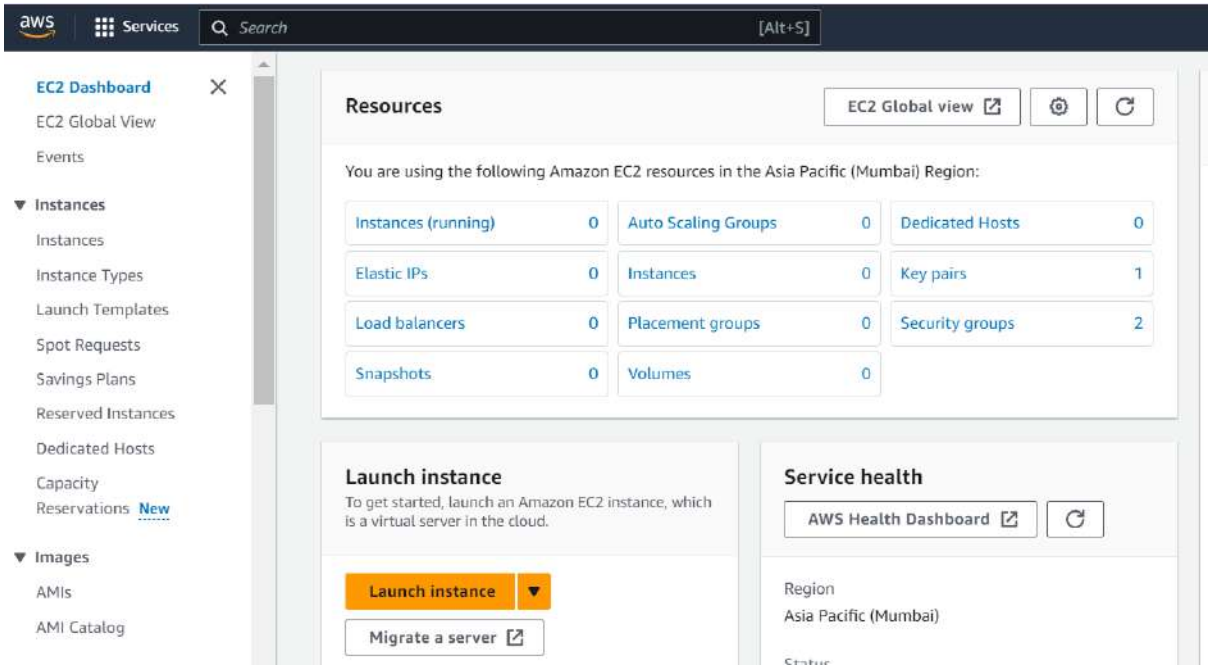
Sucu 25/3/24 (A)

EXPERIMENT 7

To study and Implement Identity and Access Management (IAM) practices on AWS.

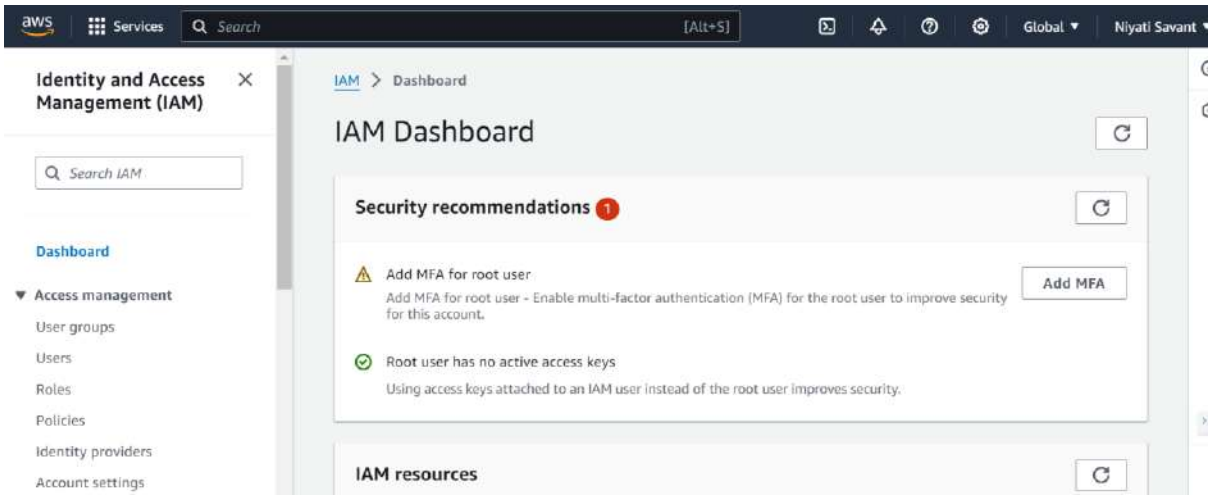
Login to AWS console

Make sure to check all Ec2 dashboard parameters



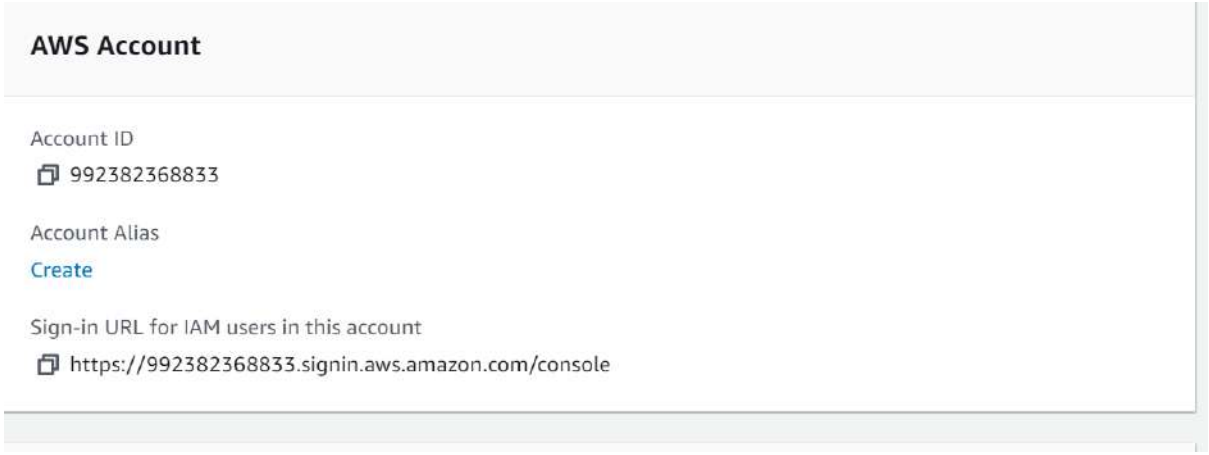
----- Configuring IAM Dashboard -----

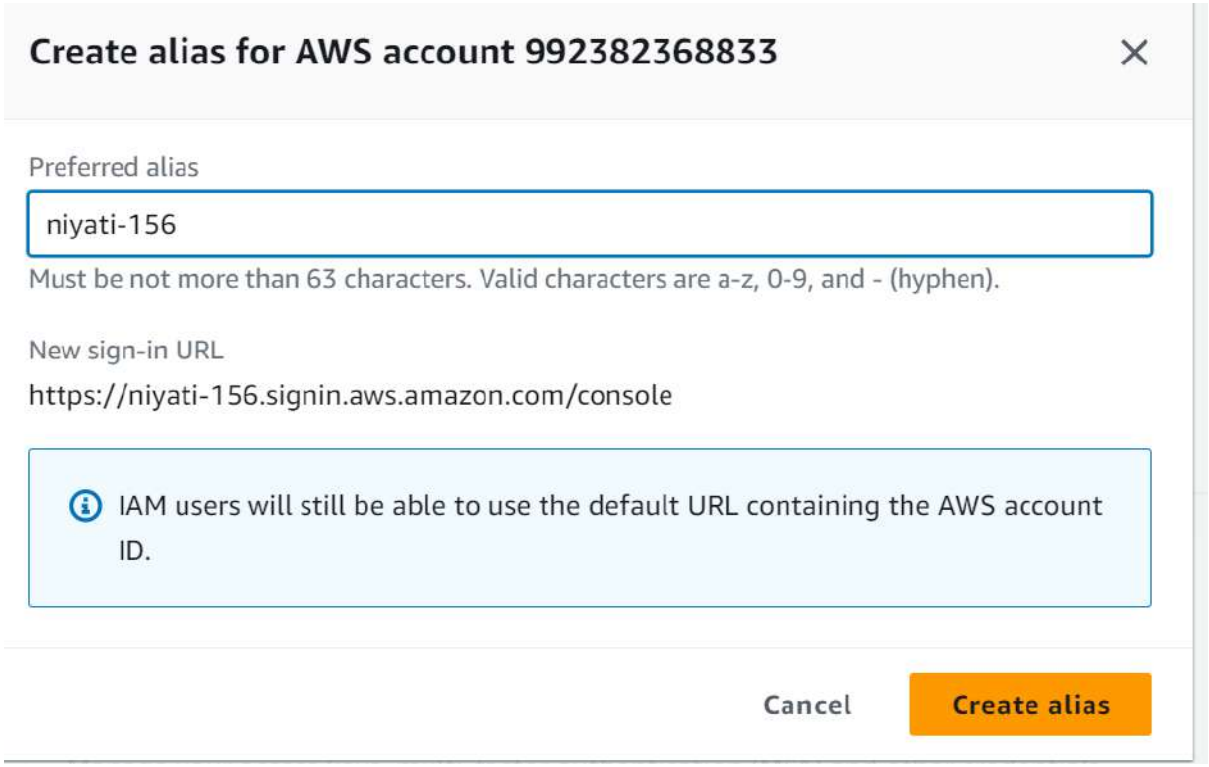
Go to IAM dashboard



Click on create option under Account Alias and give a valid name; save changes

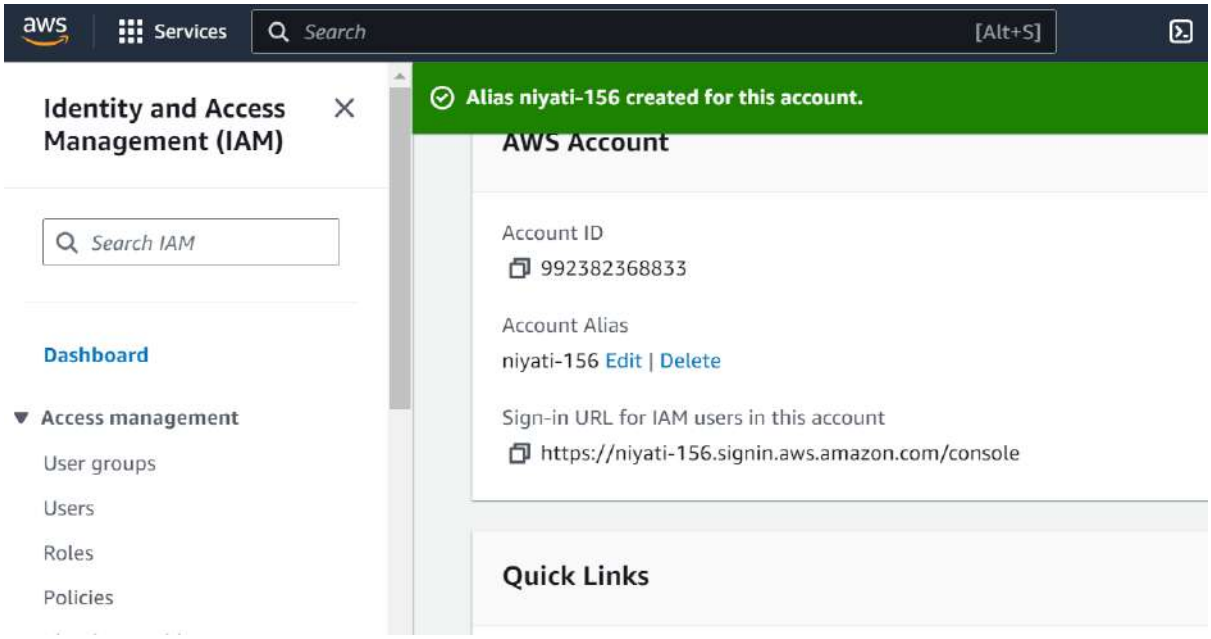
(Download Google Authenticator from PlayStore in your Mobile Phone)



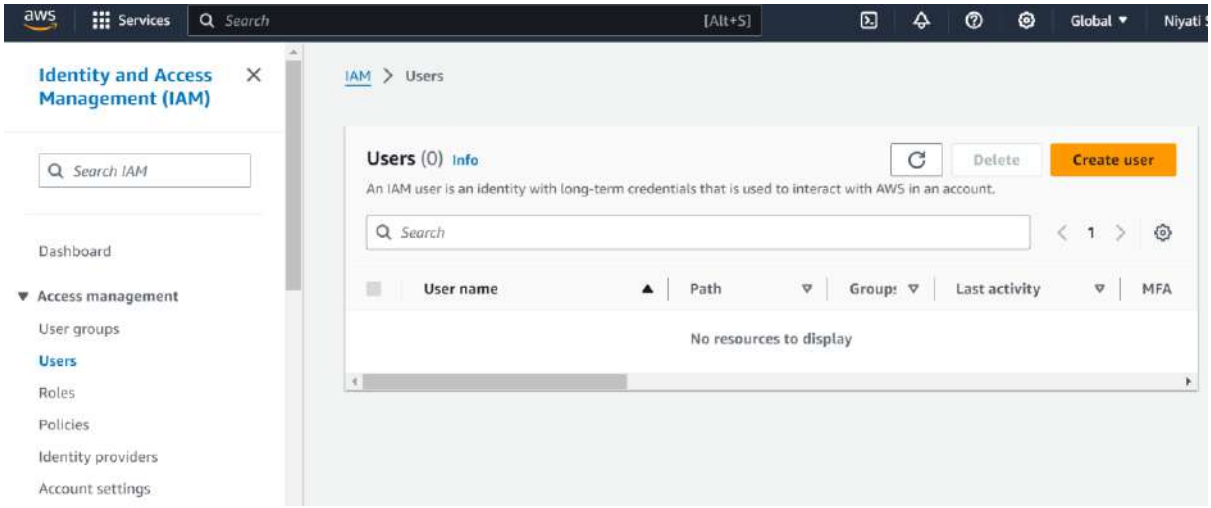


----- Creating a new User -----

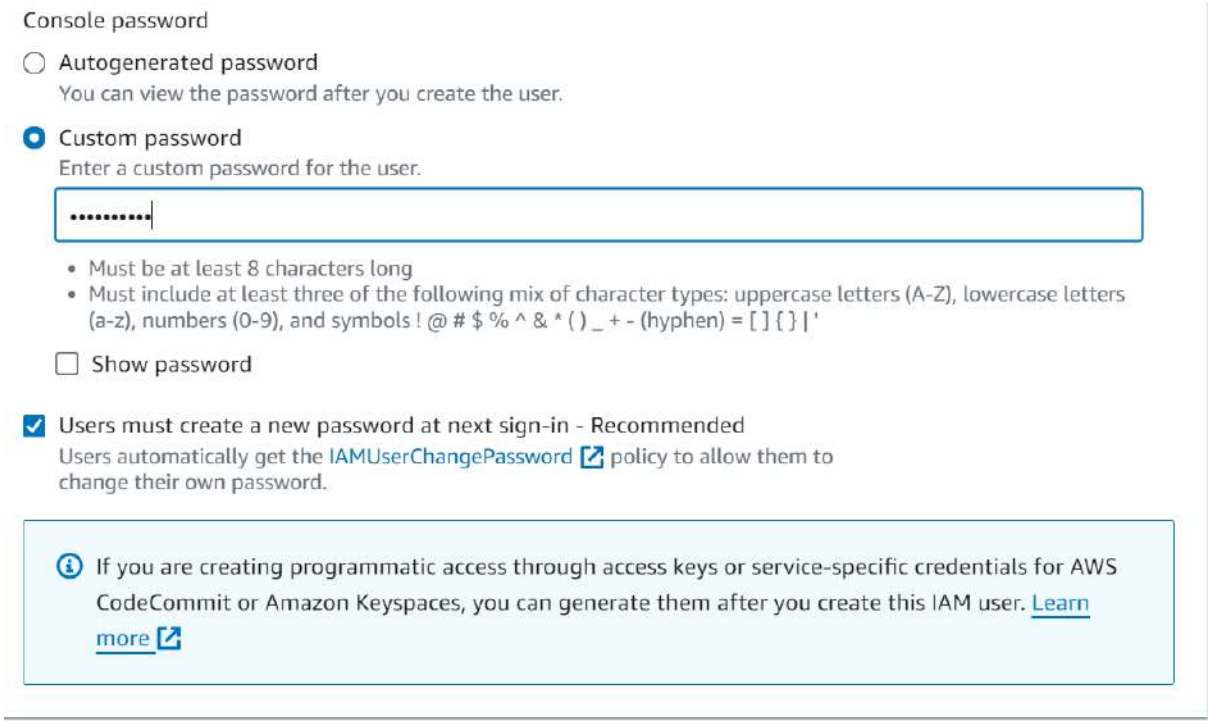
Click on “users” in the left column



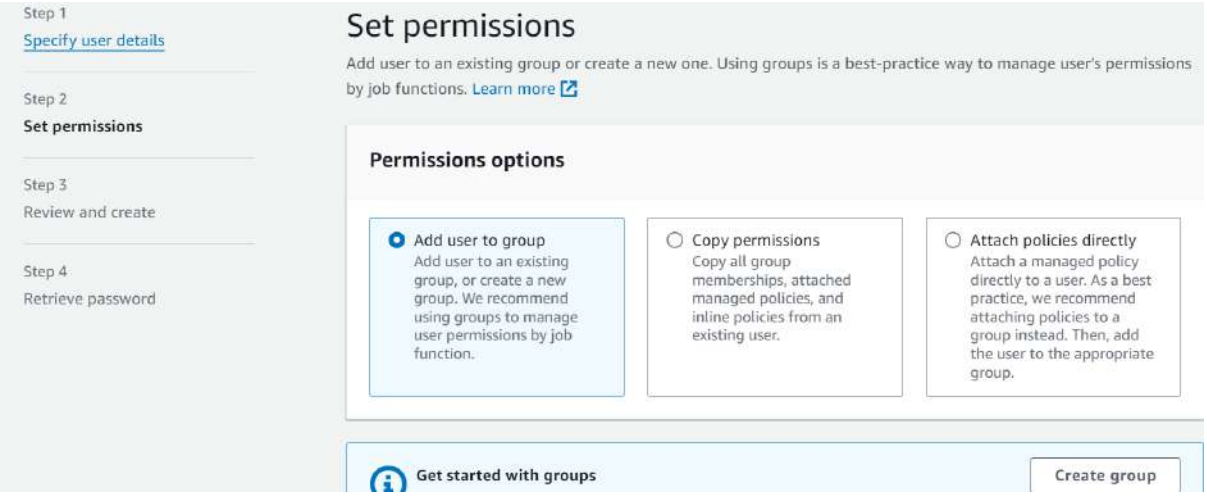
Click on Add users button



Set a custom valid psw (Imc: Qwertyuiop123) and check the Require psw rest box which will make you create a next psw in the next sign in

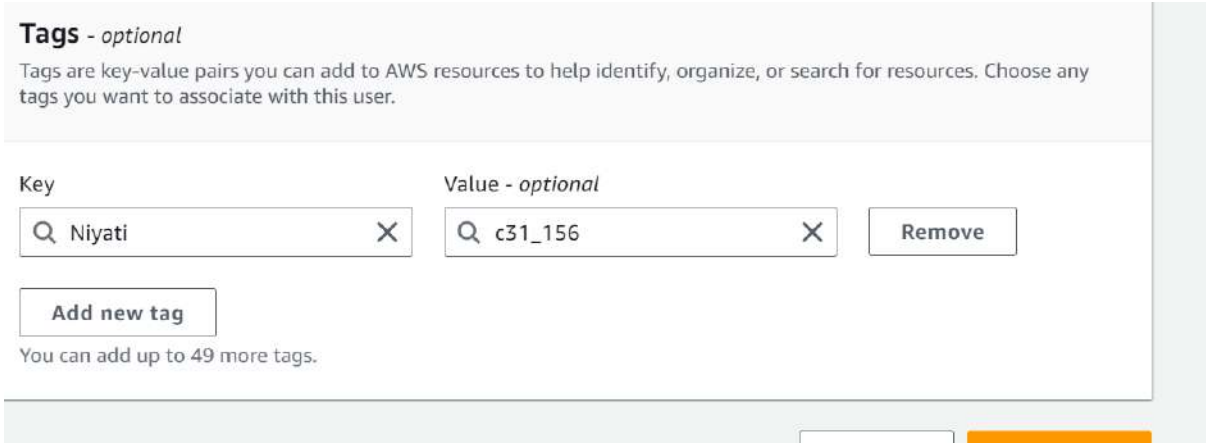


Click on Next: Tags

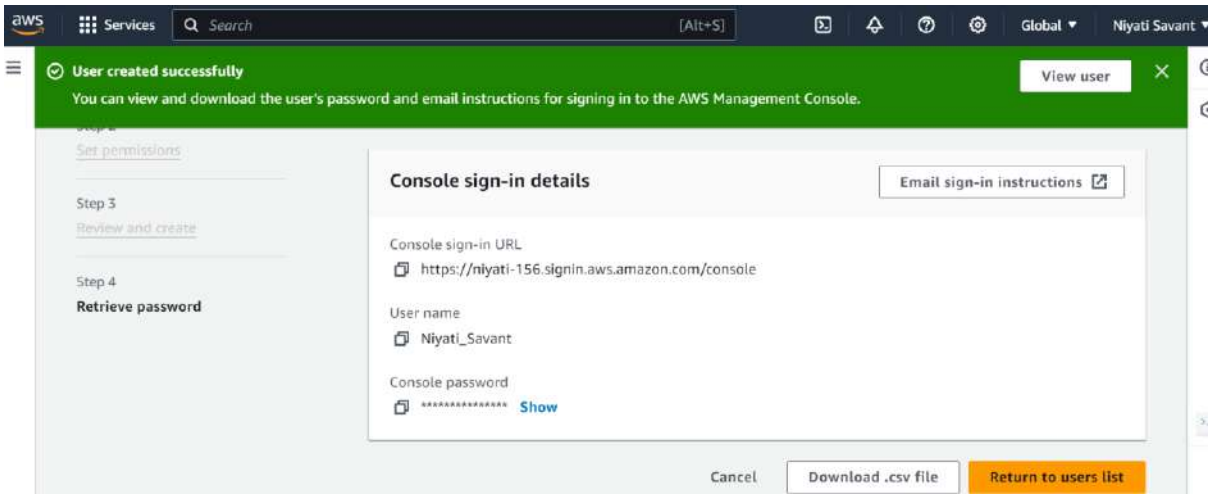


Add a tag if you want to just to keep track of your activities; then click on Next: Review

Niyati_Savant_TE_C31_2103156



Click on Create User Button



Open the URL in Incognito Mode

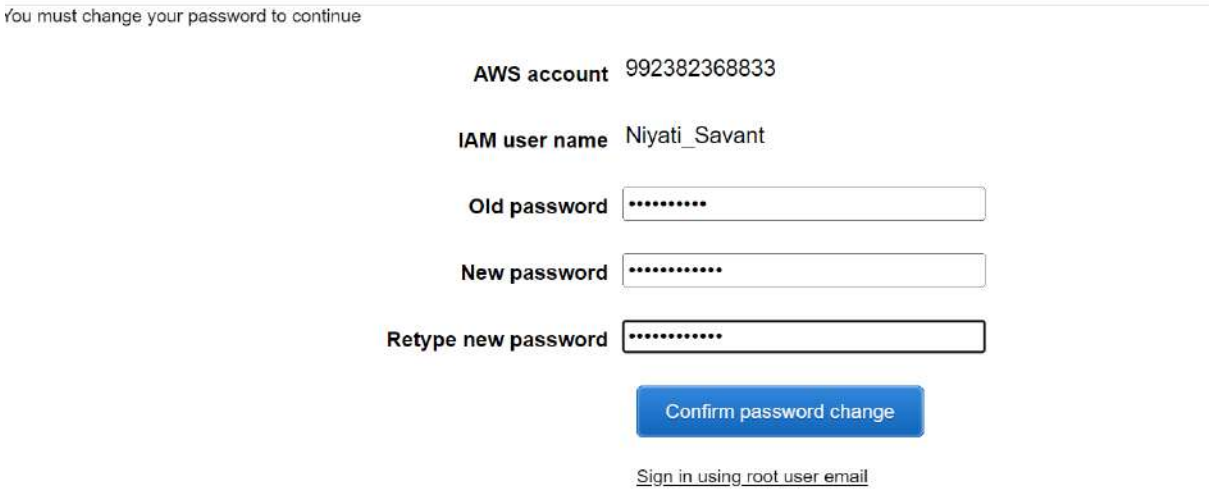
(Imc: https://niyati-156.signin.aws.amazon.com/console)

Note: Save the secret access ID & key in a notepad or download the csv

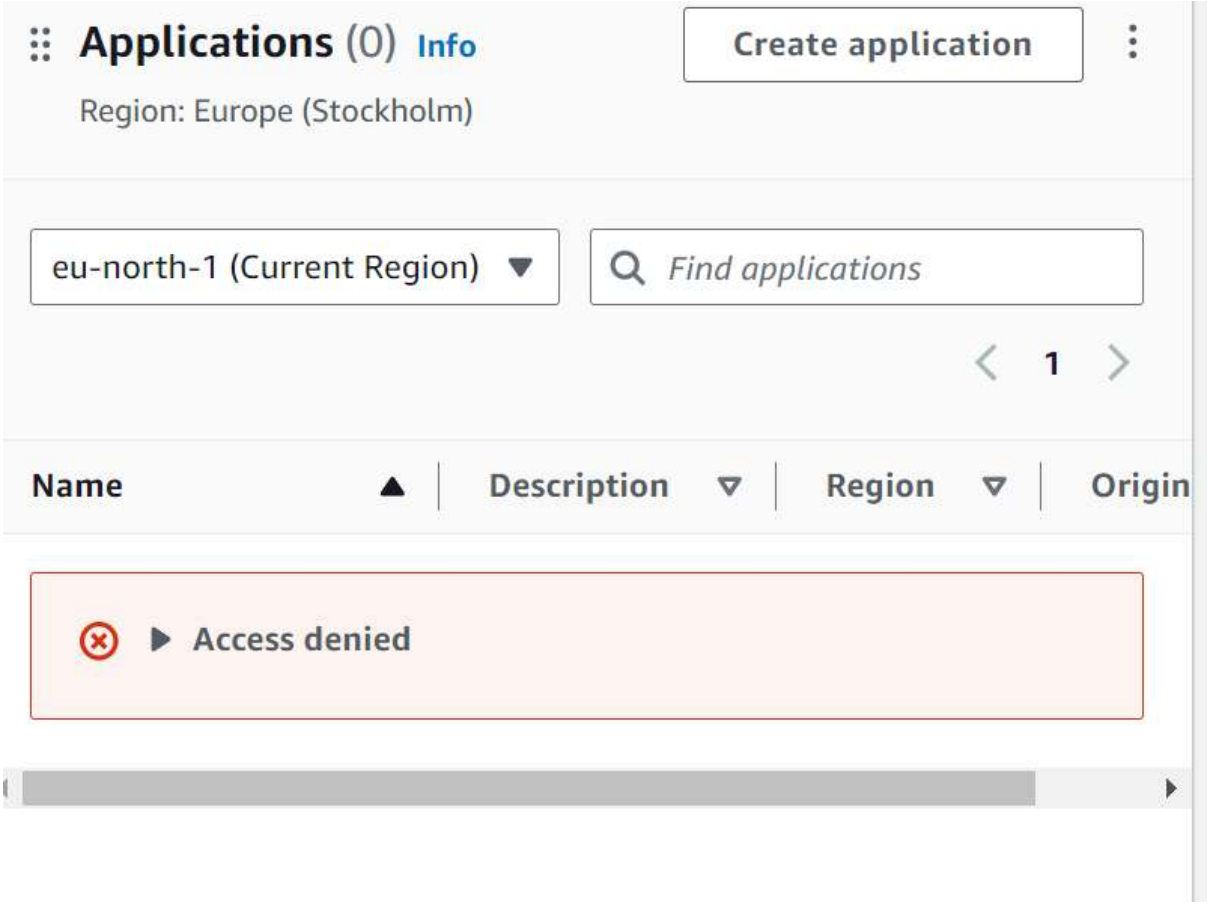
----- Logging in as the new User & Checking their permissions -----

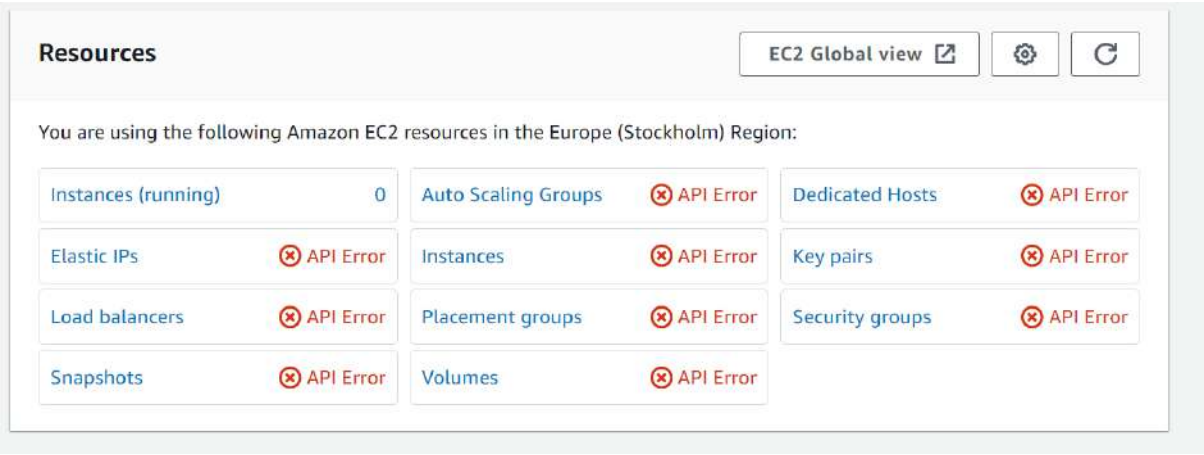
Enter the new user's name and psw saved earlier

Enter a new valid psw



After logging in, you will notice that you don't have permission to do anything yet

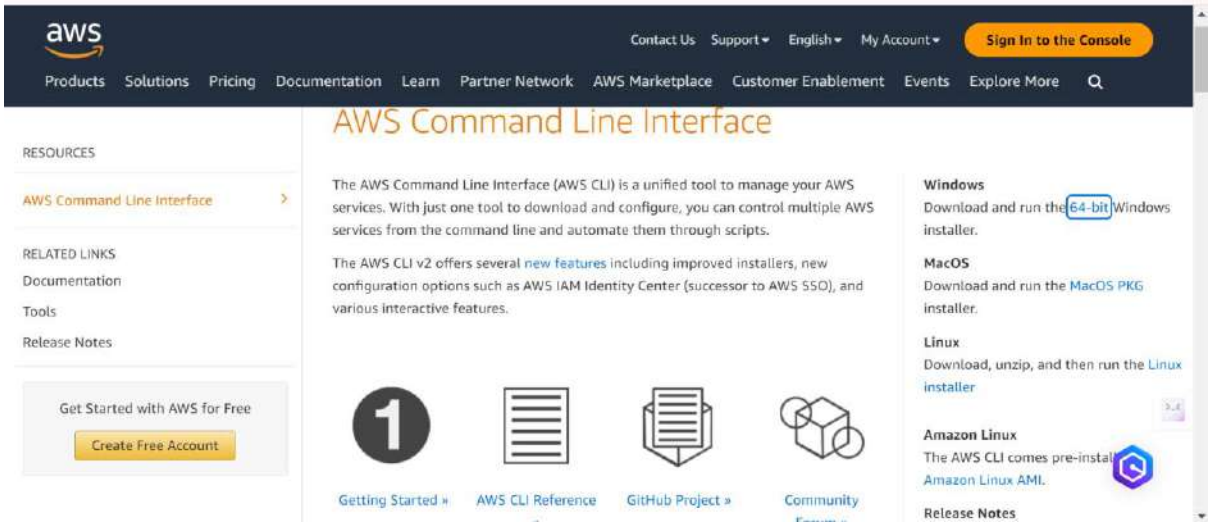


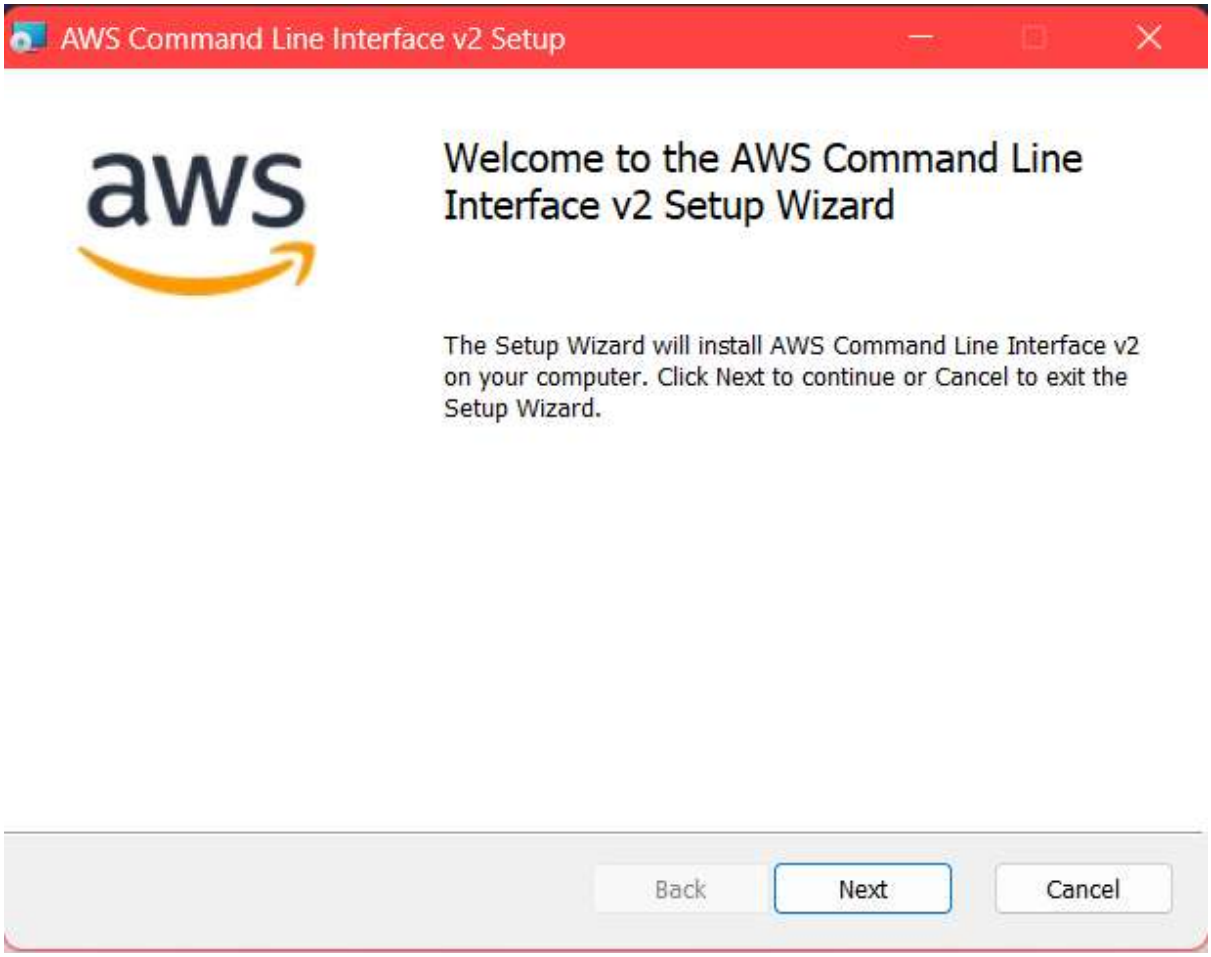


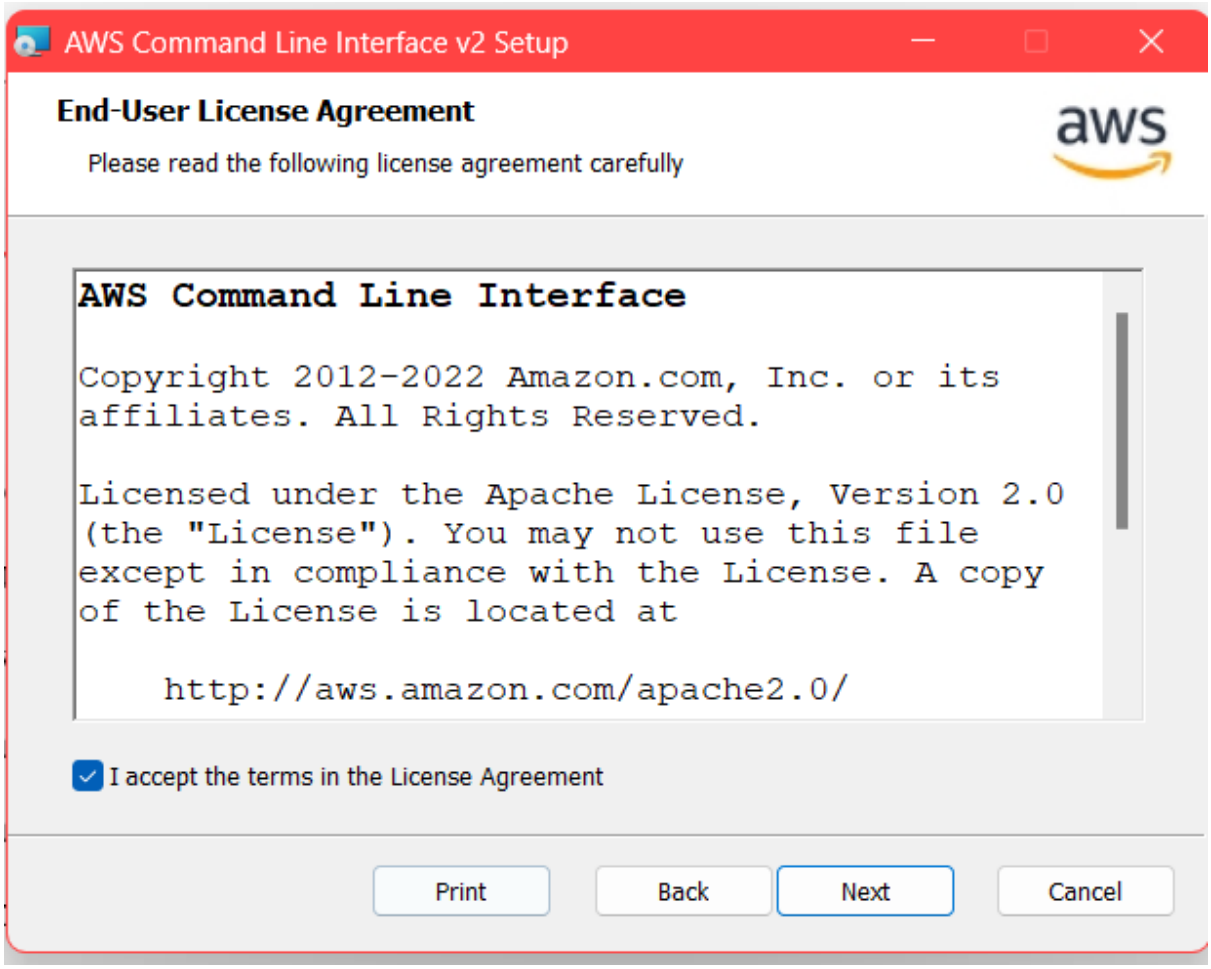
----- Adding MFA for the user via Root User -----

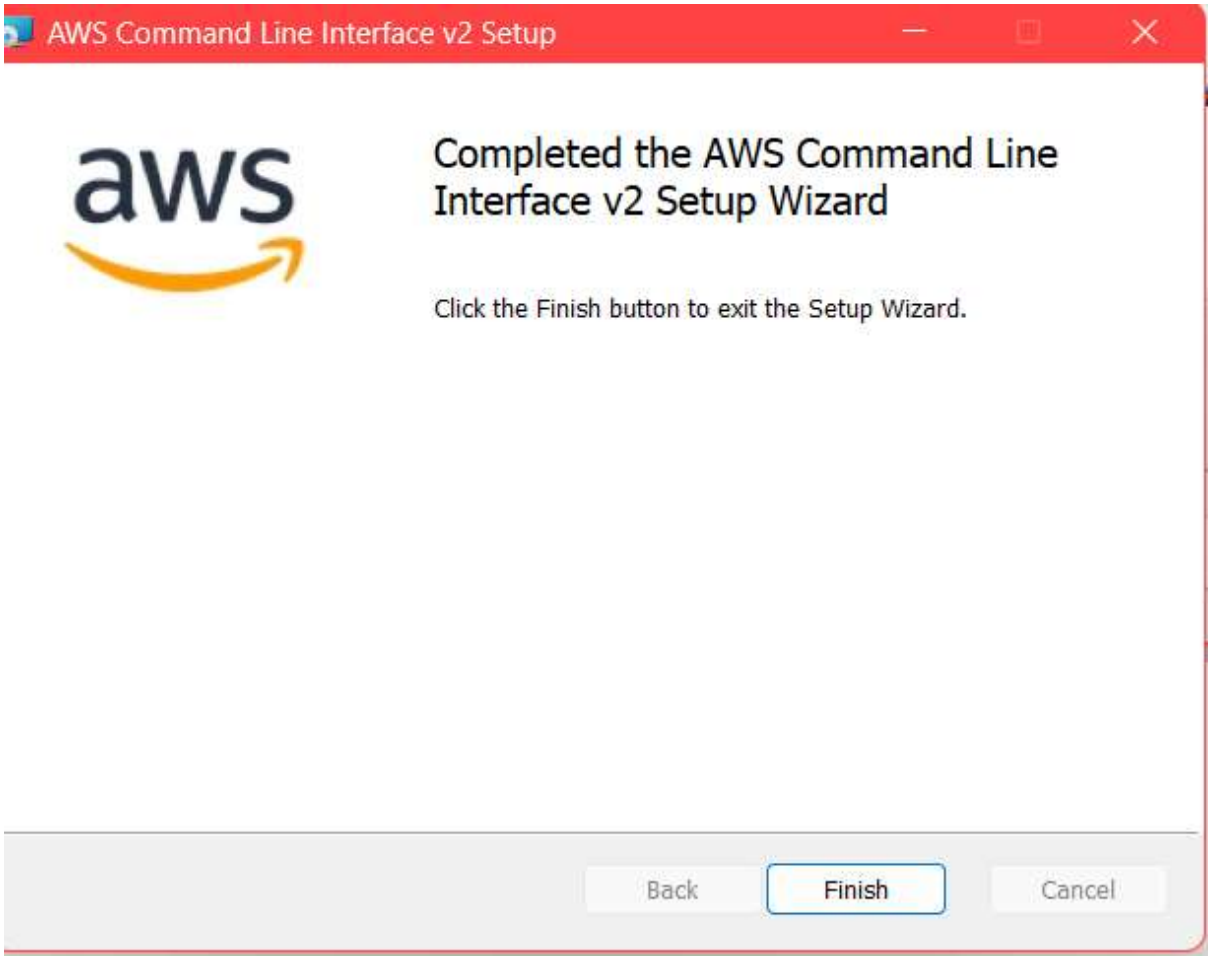
Type “AWS CLI” in a new window of any browser and go to it’s the main page of AWS regarding the same

Click on 64-bit hyperlink in the RHS column under the Windows section and download, install the AWS CLI

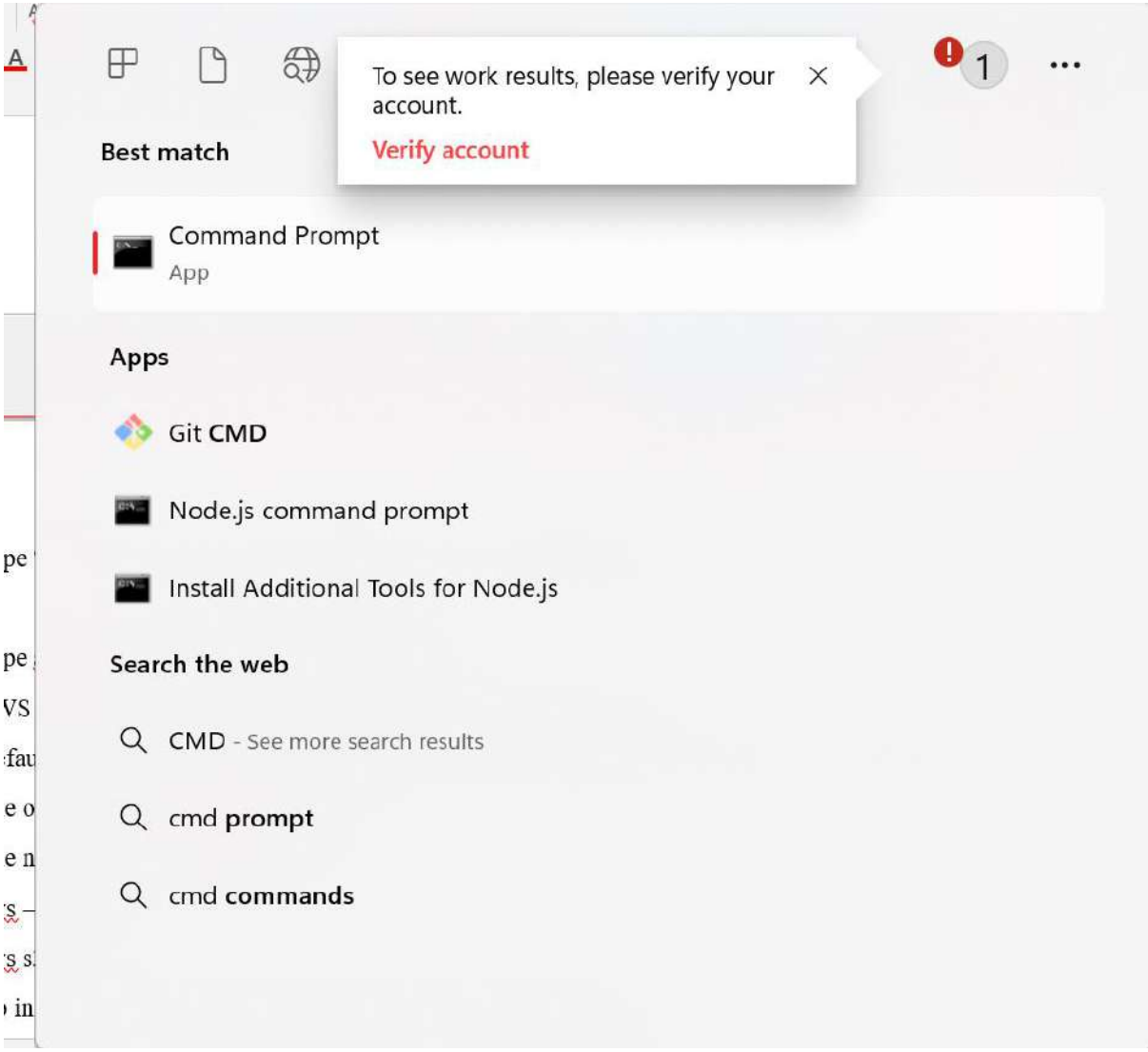



AWS Command Line Interface v2 Setup
aws
Welcome to the AWS Command Line Interface v2 Setup Wizard
The Setup Wizard will install AWS Command Line Interface v2 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel


AWS Command Line Interface v2 Setup
End-User License Agreement
Please read the following license agreement carefully
aws
AWS Command Line Interface
Copyright 2012-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved.
Licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at
http://aws.amazon.com/apache2.0/
I accept the terms in the License Agreement
Print
Back
Next
Cancel



Type “cmd” in the windows search bar and run it as an administrator



Type aws configure, it will ask for a few inputs;

AWS Access Key ID and Key are the ones which we saved earlier

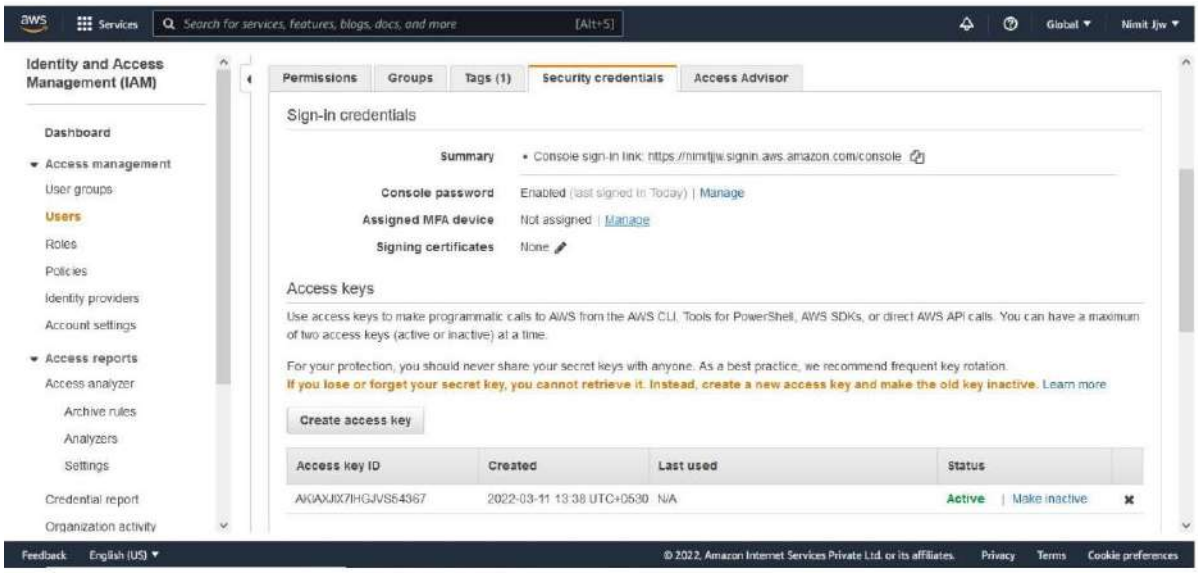
Default region name is whichever region AWS you are using; in case of Mumbai, its: ap□south-1

The output format is json in our case

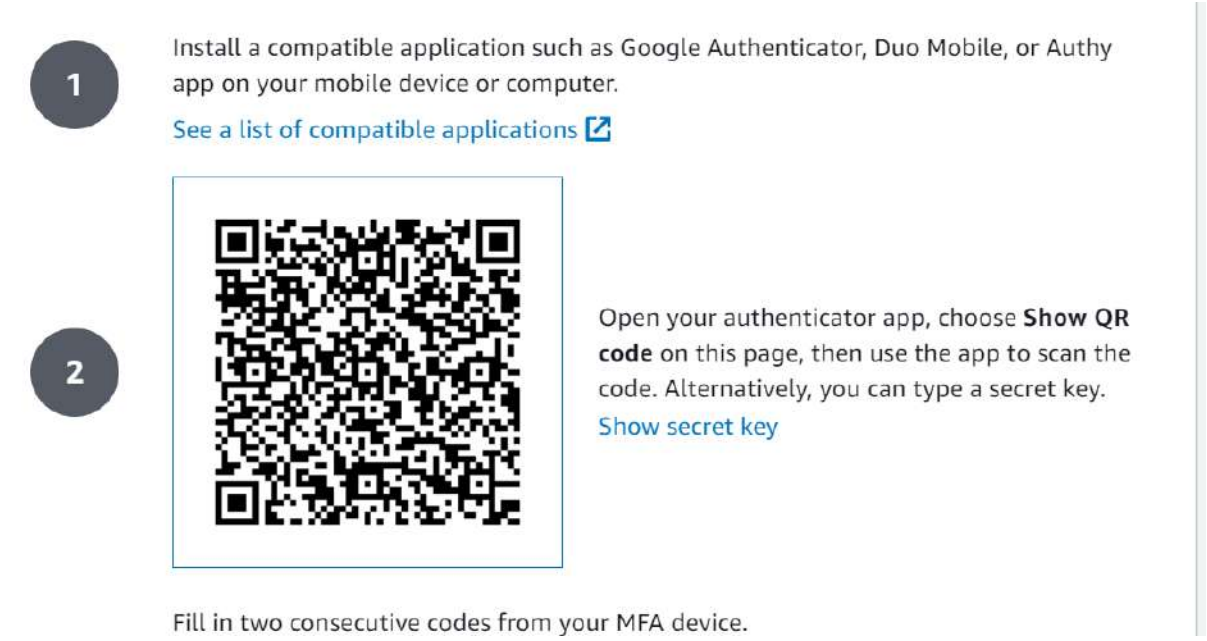
```
C:\Users\Niyati Savant>aws configure
AWS Access Key ID [None]: AKIA6ODUZHRA6C2UWBPT
AWS Secret Access Key [None]: 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
Default region name [None]: ap-south-1
Default output format [None]: json
```

Go in the security credentials tab under Users of IAM Dashboard

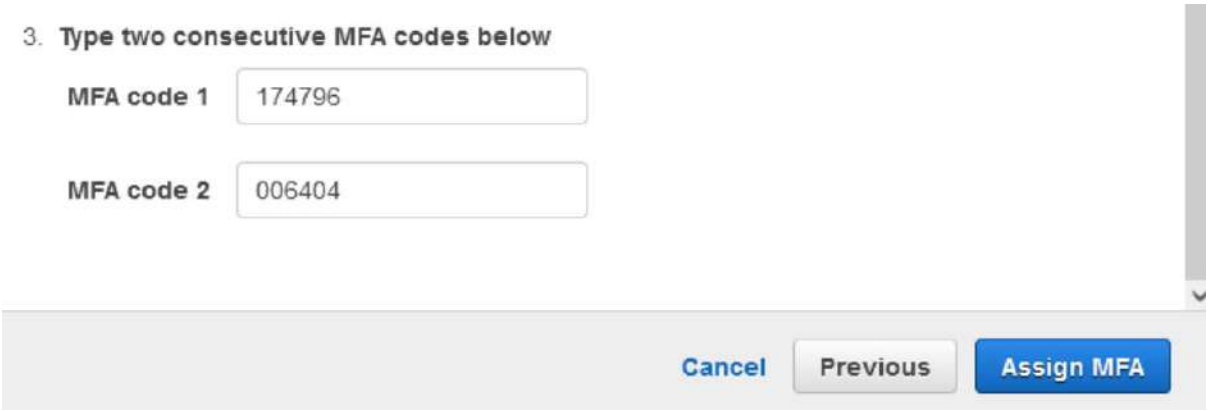
Click on the “Manage” Hyperlink

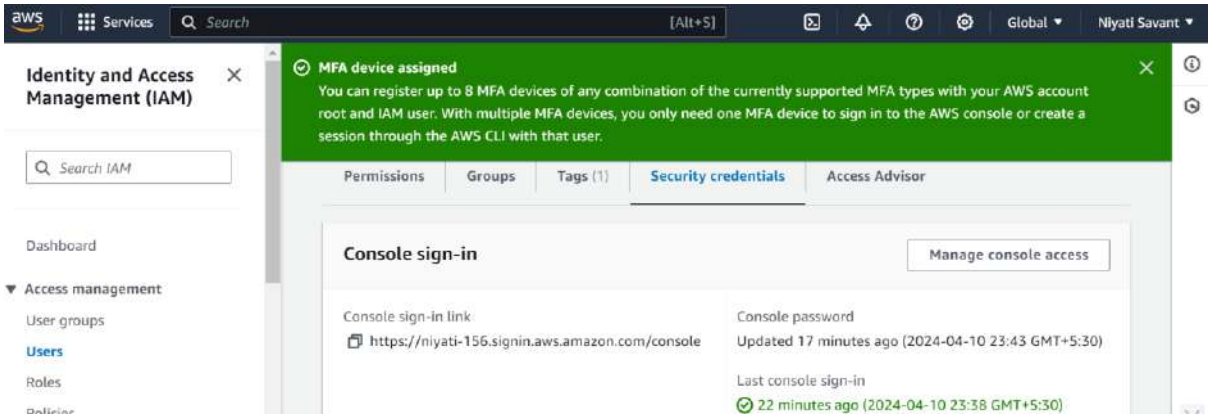


Use the Google Authenticator app downloaded earlier to scan the QR Code



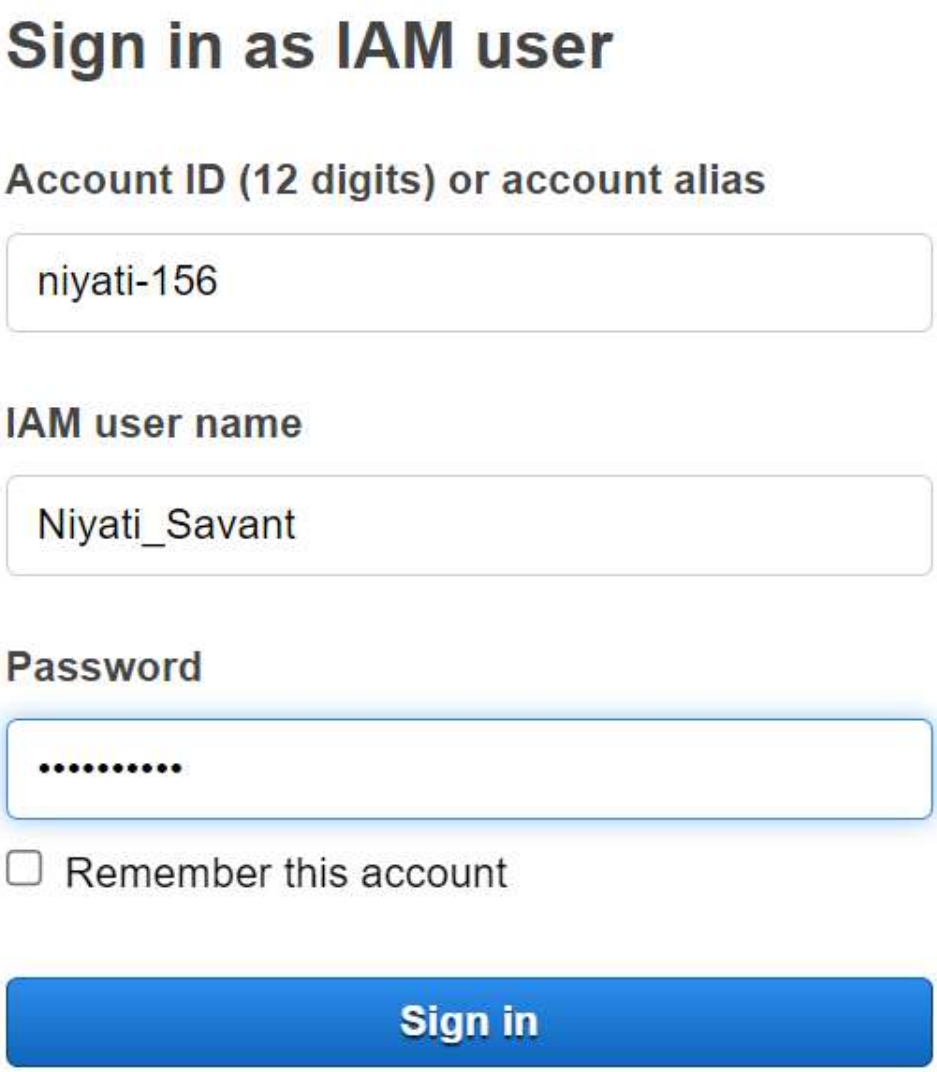
Enter two of the codes which are shown in the Google Authenticator App over a span of 30 secs each; click on Assign MFA Button



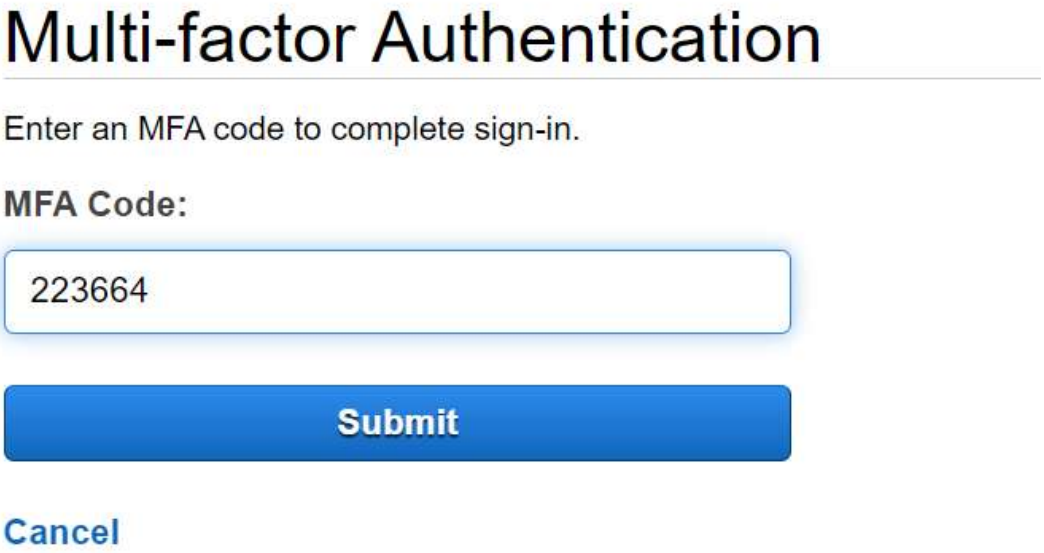


----- Logging in as the new user after MFA -----

Again try logging in via the new user created earlier; this time it will ask for MFA after you click on Sign In

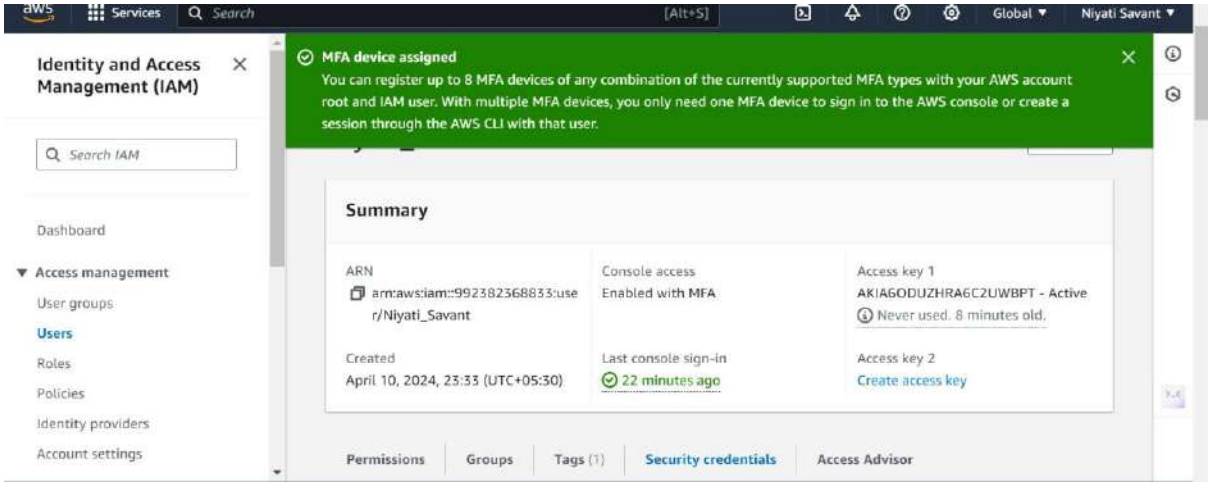


Use the code being shown in the Google Authenticator



Now, after opening the root user window again

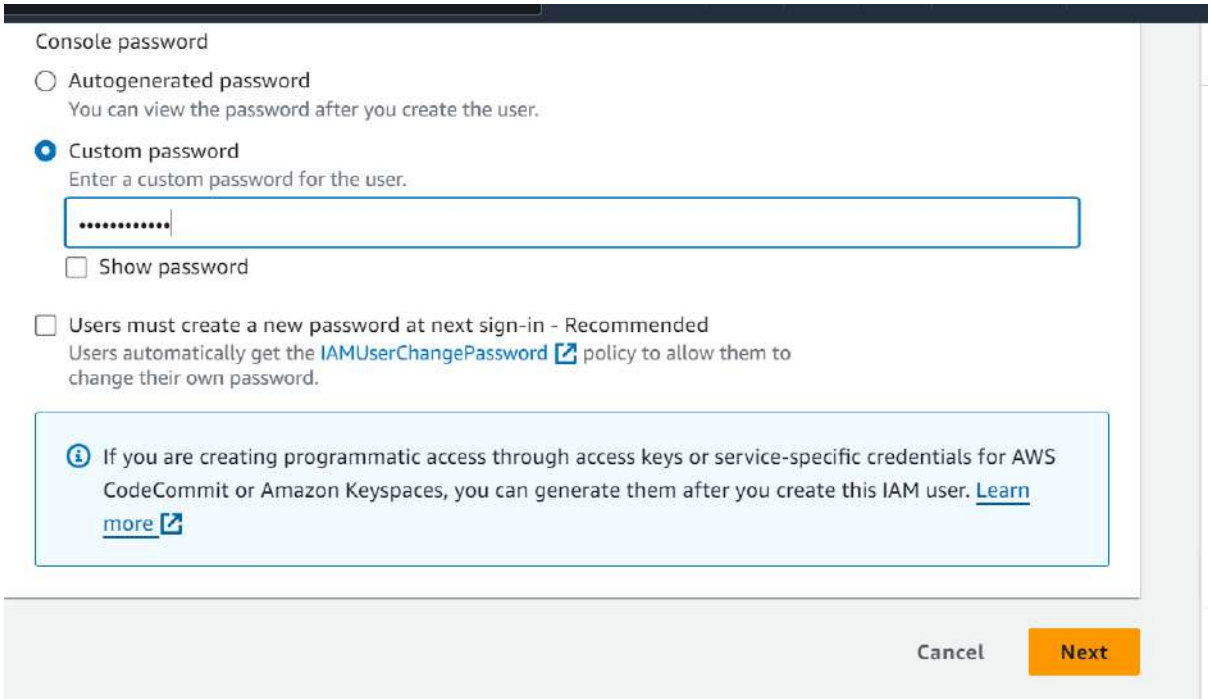
After going in the Users section of IAM Dashboard, we can see that MFA has been activated for the new user.



----- Adding 3 More Users and giving them permissions -----

Now, Adding 3 More Users

Not giving them an Access key and not checking the Psw Reset Checkbox; Click on the Next: Permissions

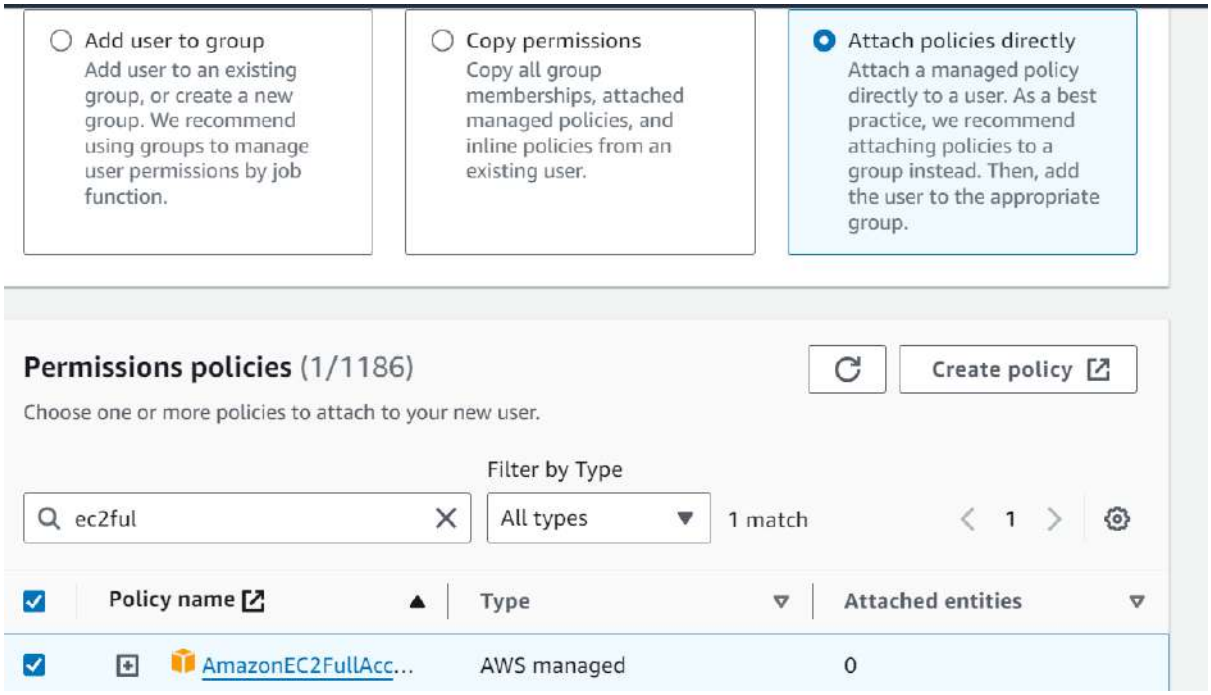


We will create a group later

We can see the previous user listed under the copy "permission from existing user" section (just for observation purpose)

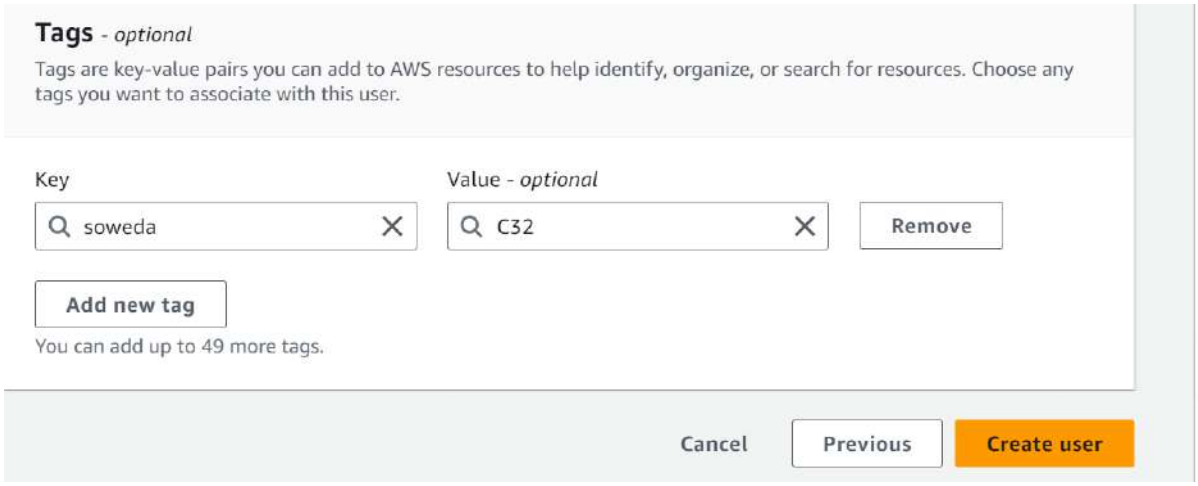
Click on the third section: Attach existing policies directly

Type in ec2fullaccess in the search box and click the check box for it; click on Next: Tags



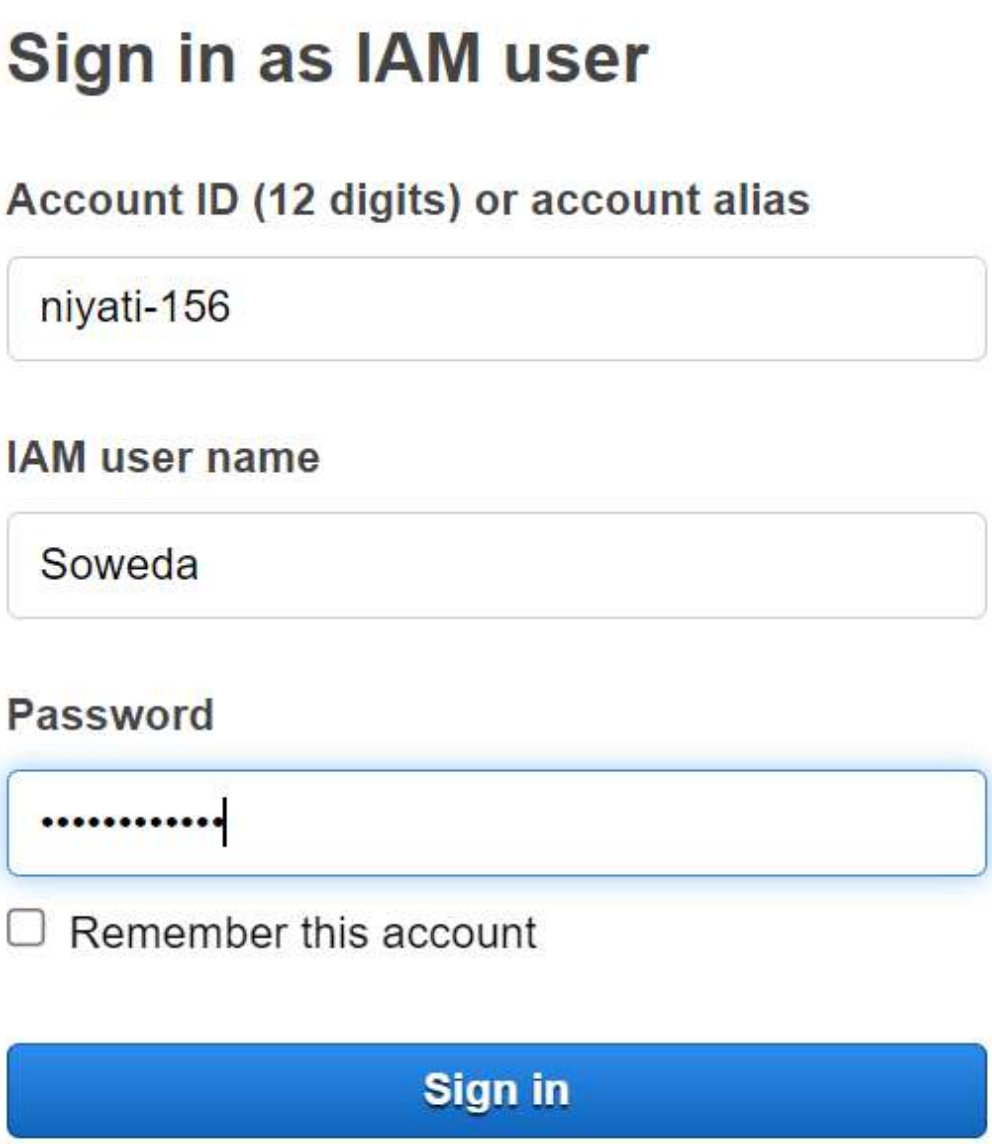
Input the Key and Value for the Tag to keep track of your activities; Click on Next: Review

Click on Create Users Button

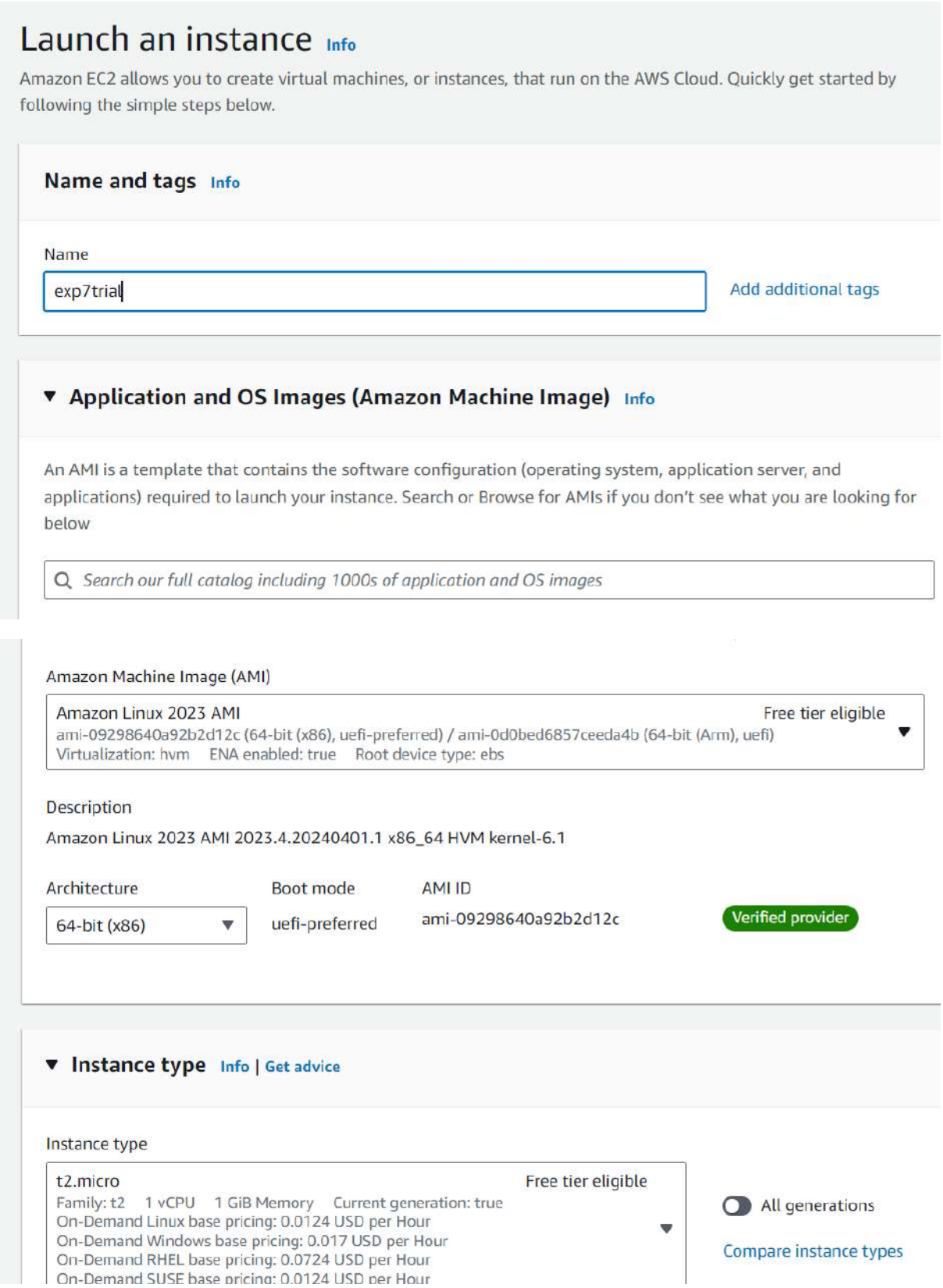


----- Logging in as one of the 3 new Users and checking their permissions -----

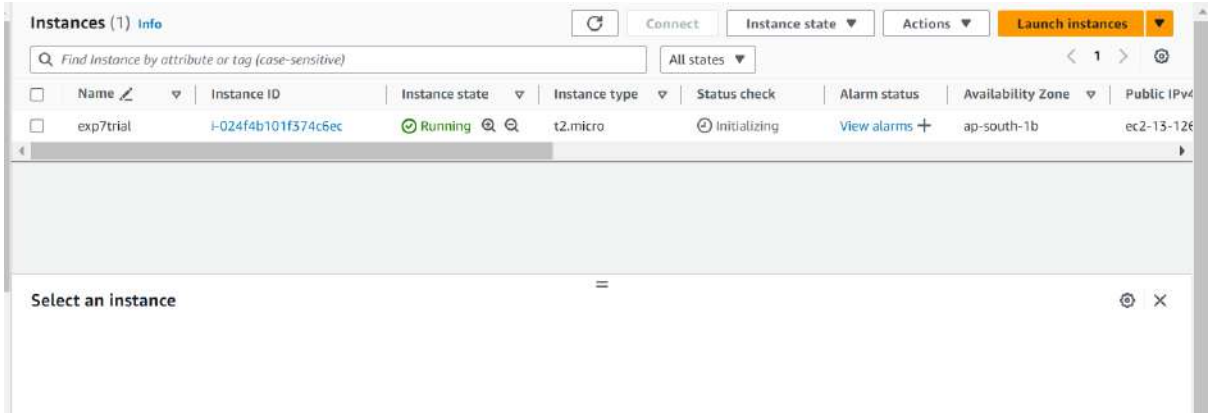
Try logging in as one of the 3 new users just created



Try launching an EC2 instance via the new user



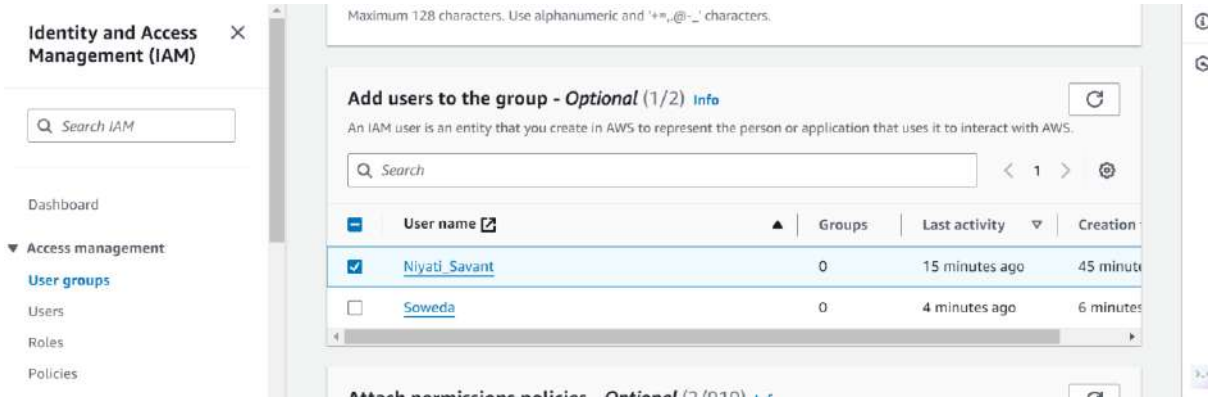
Hence, an instance has been created



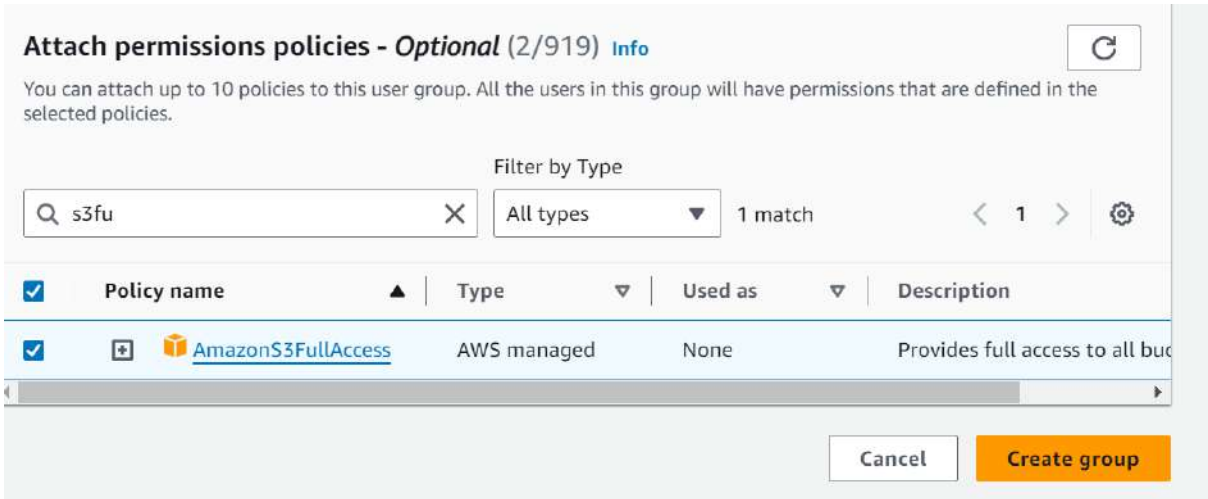
Delete the bucket when done with your work

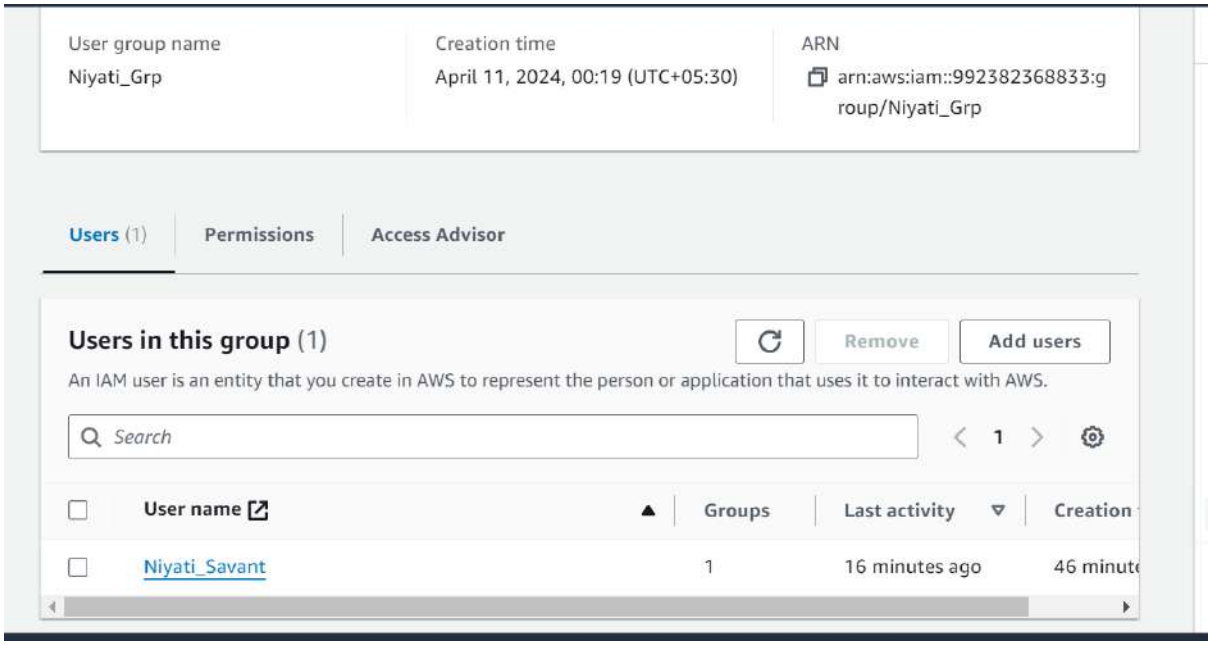
----- Creating a new Group and giving it permissions -----

Select the members to be present in the group (max 4 per group)



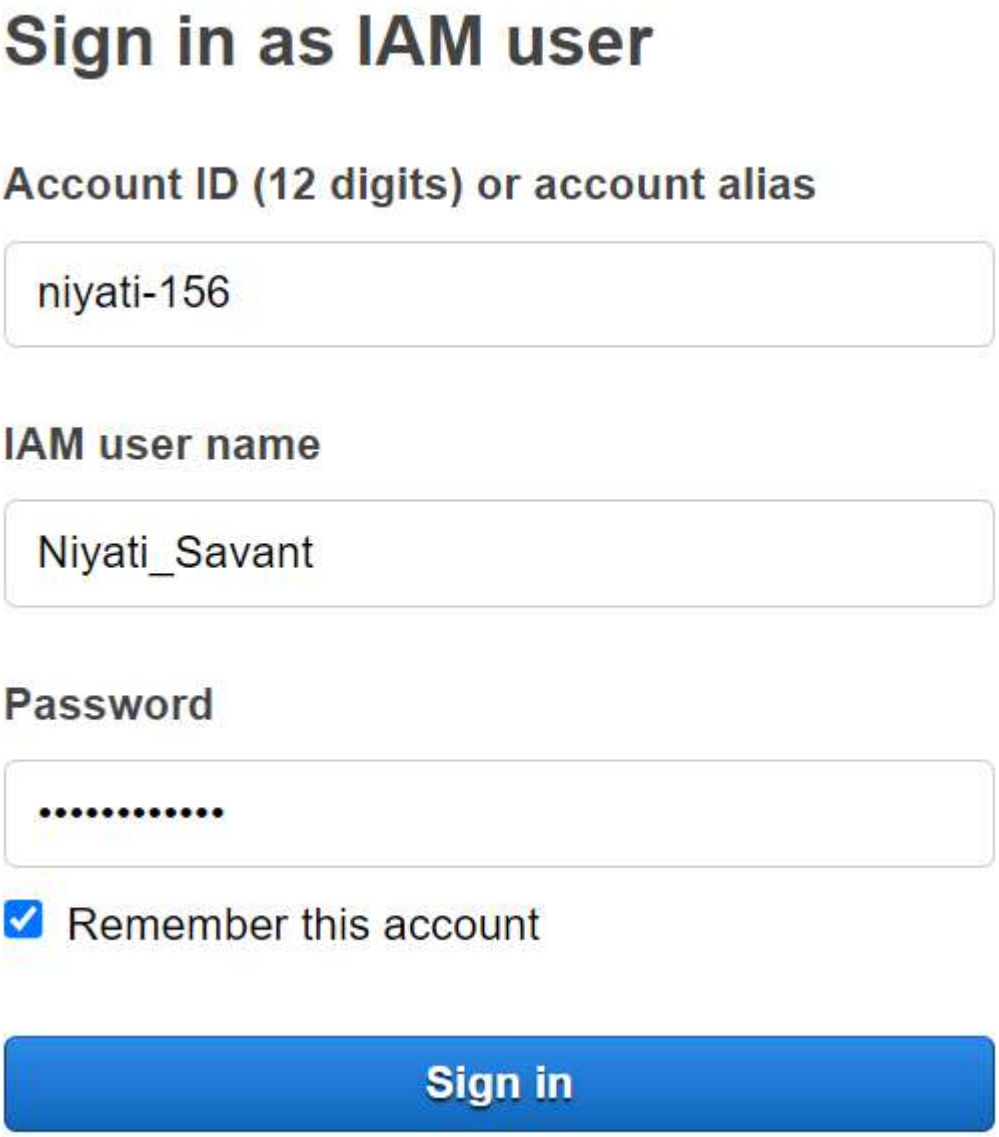
Giving this group ec2fullaccess and s3fullaccess



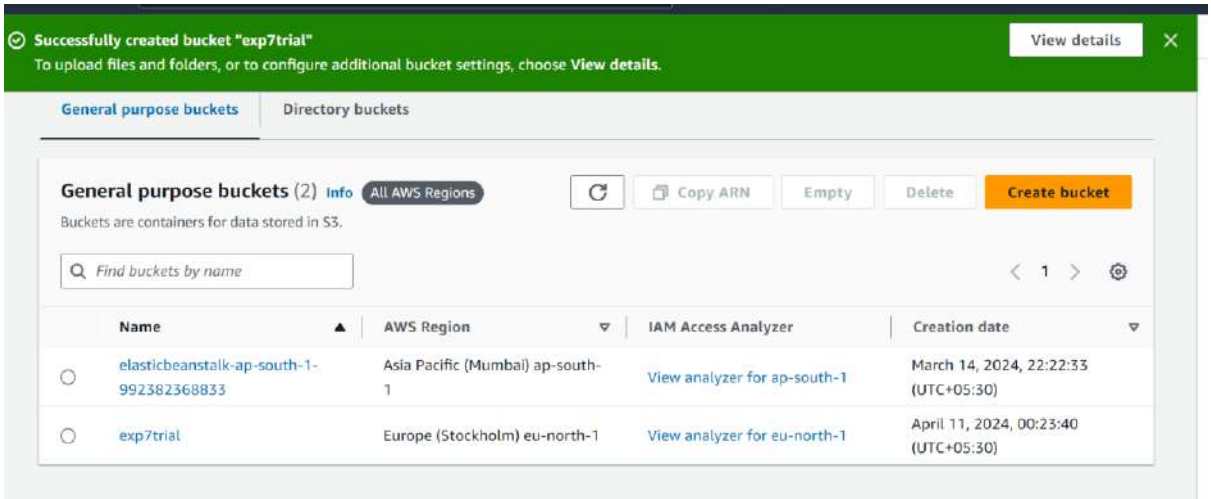


----- Logging in as a member of the Group & checking their permissions -----

Now, login as one of the users from the group and try creating a S3 bucket



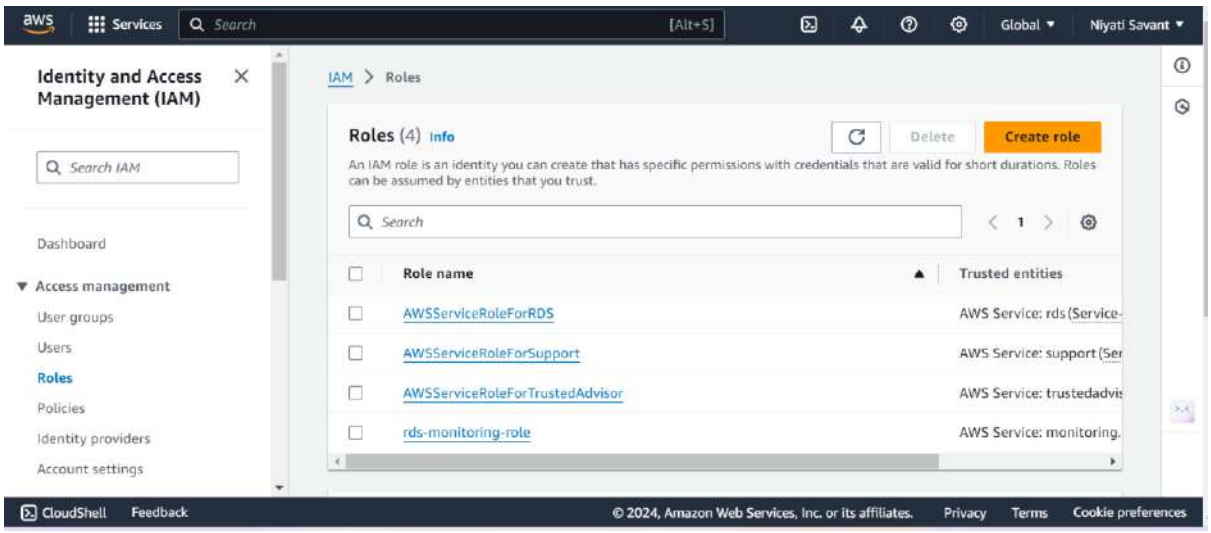
S3 bucket successfully created



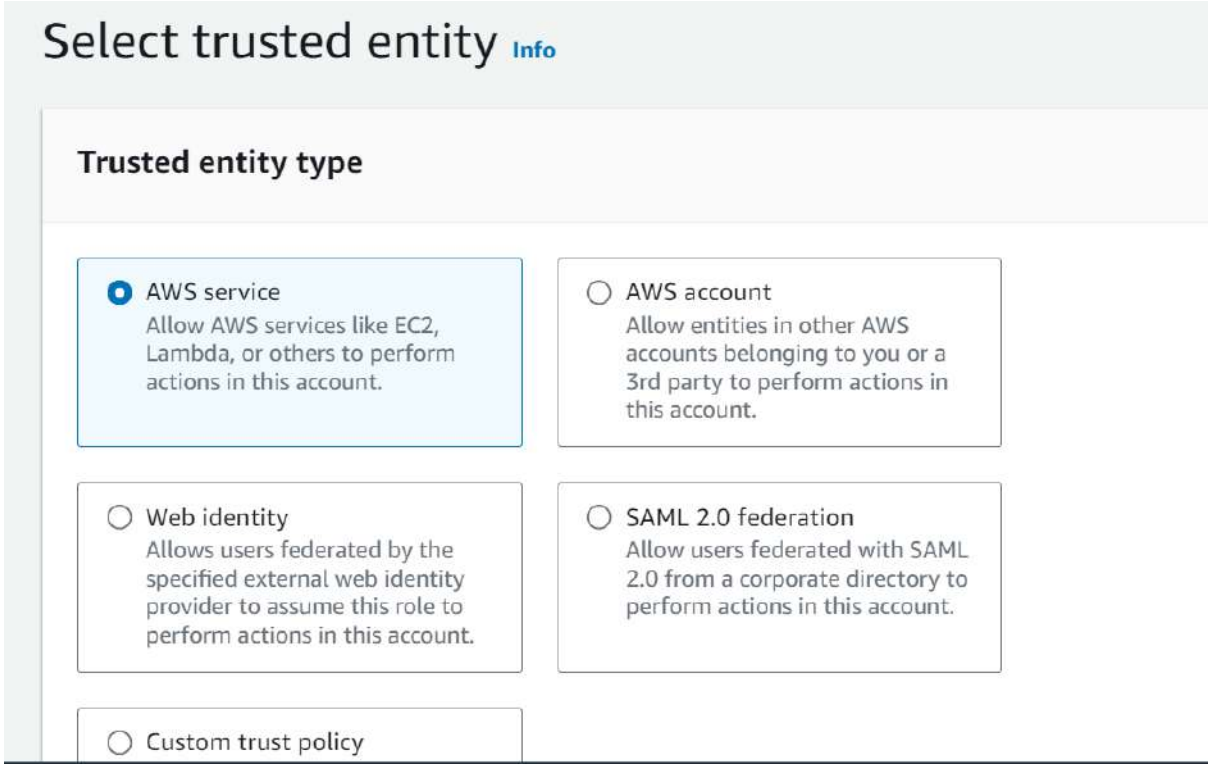
Delete the bucket when done with your work

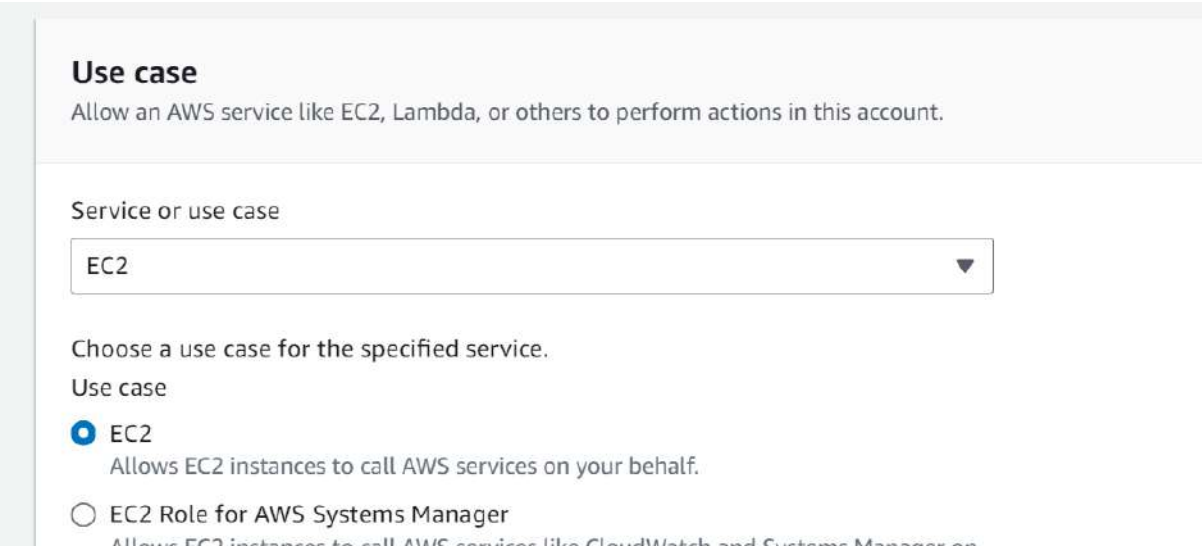
----- Creating a new Role -----

Go in the root user window and click on “create role” button in the “Roles” section of IAM Dashboard



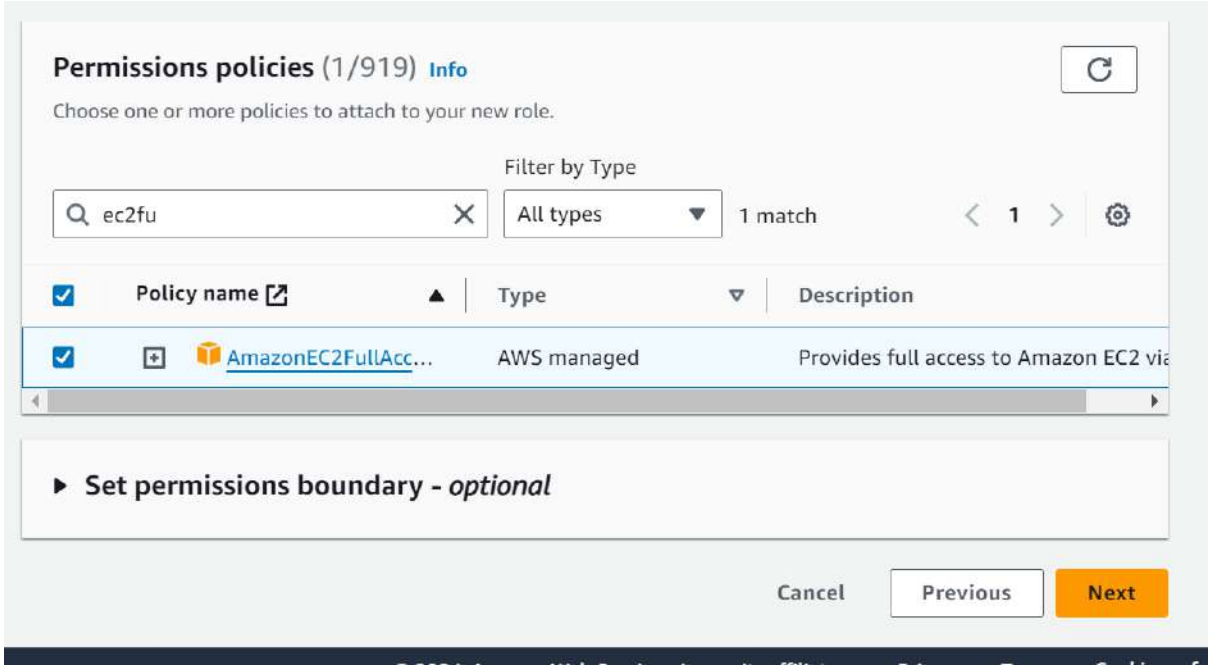
Let it be the default options (you can choose any use case you like)



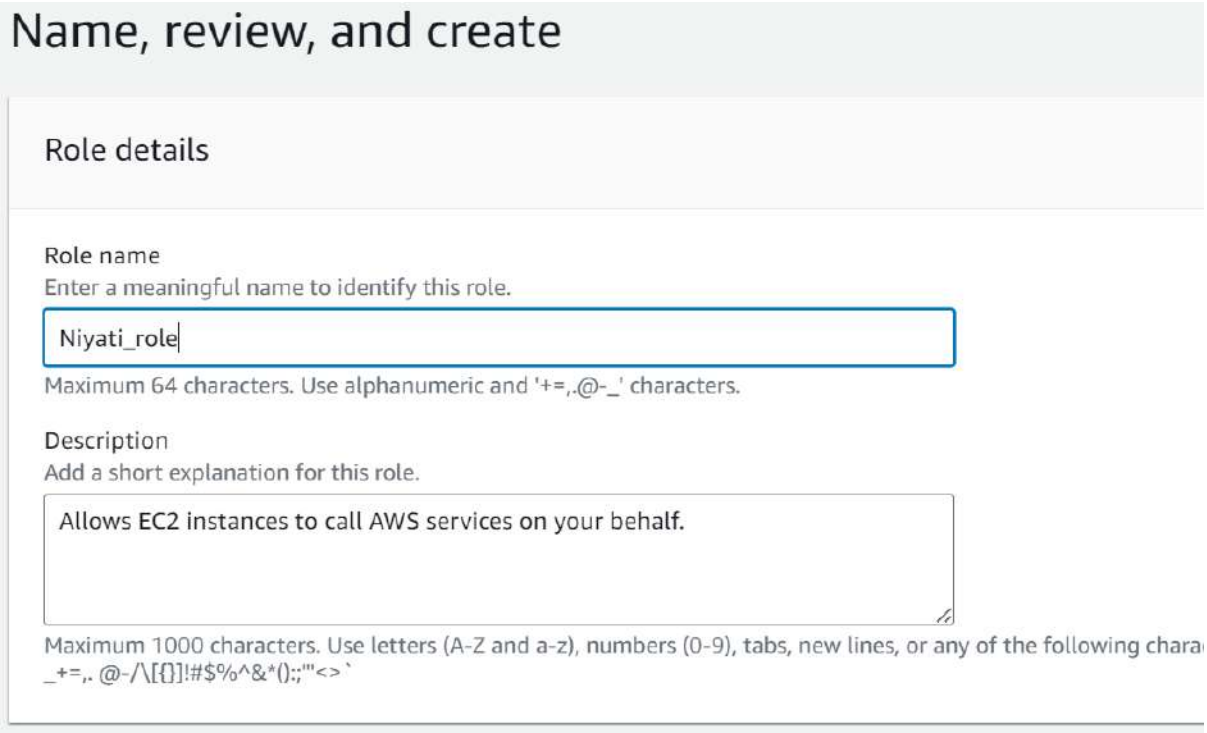


Click in Next button

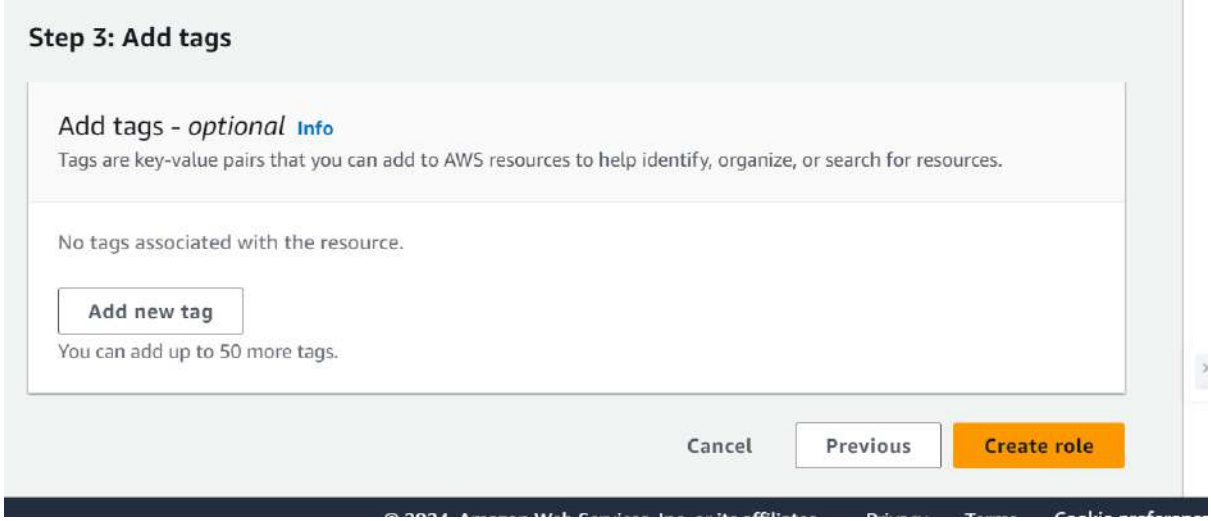
Give the permission suitable to the use case chosen



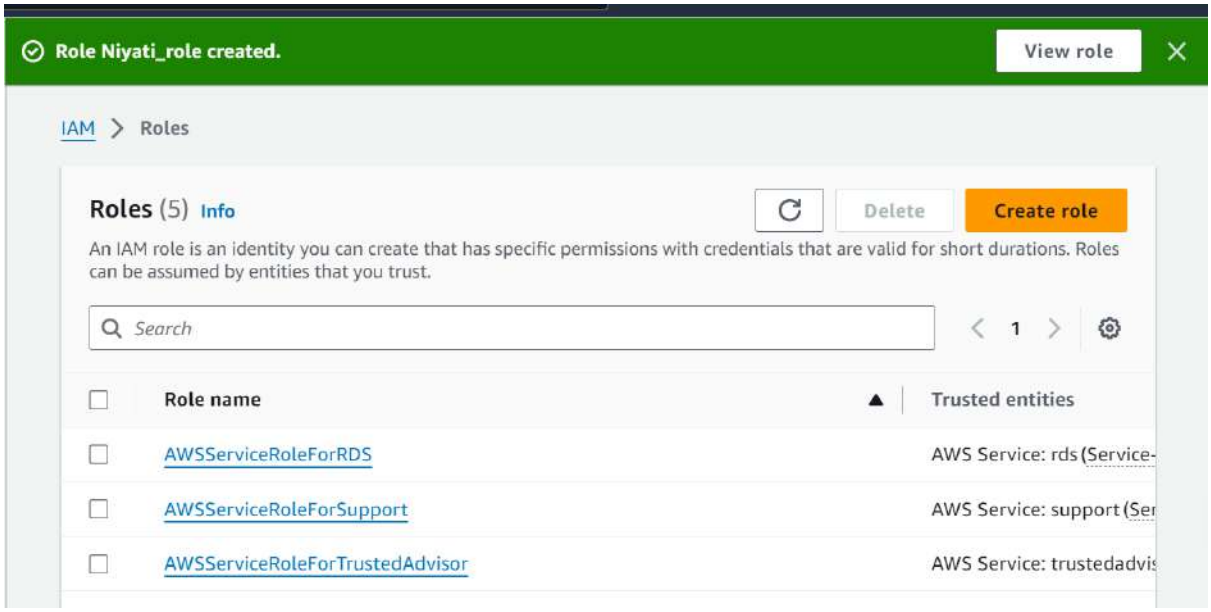
Give suitable Role name and description; rest would remain as default



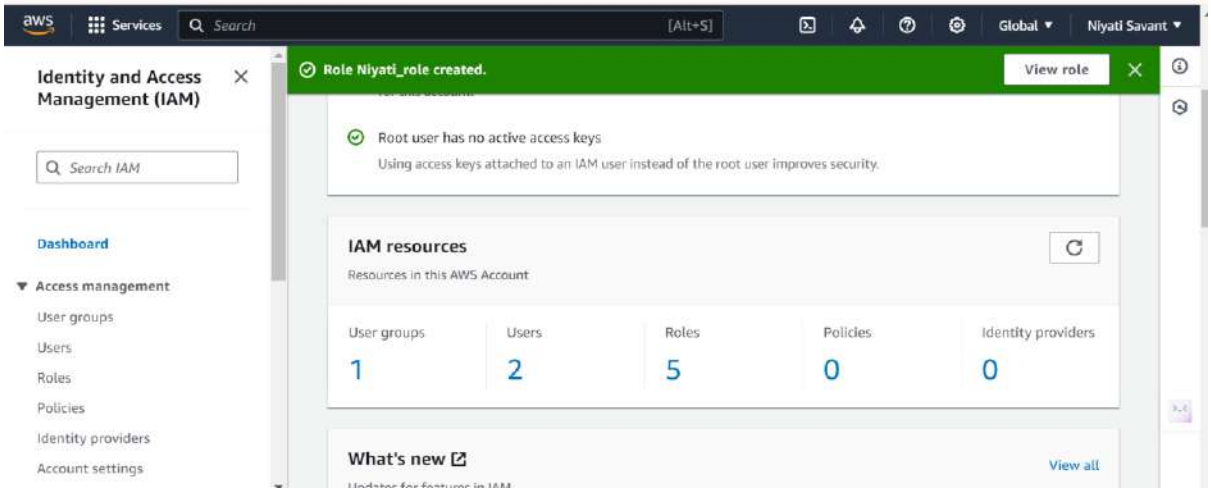
Add a tag if you want to; click on Create Role button



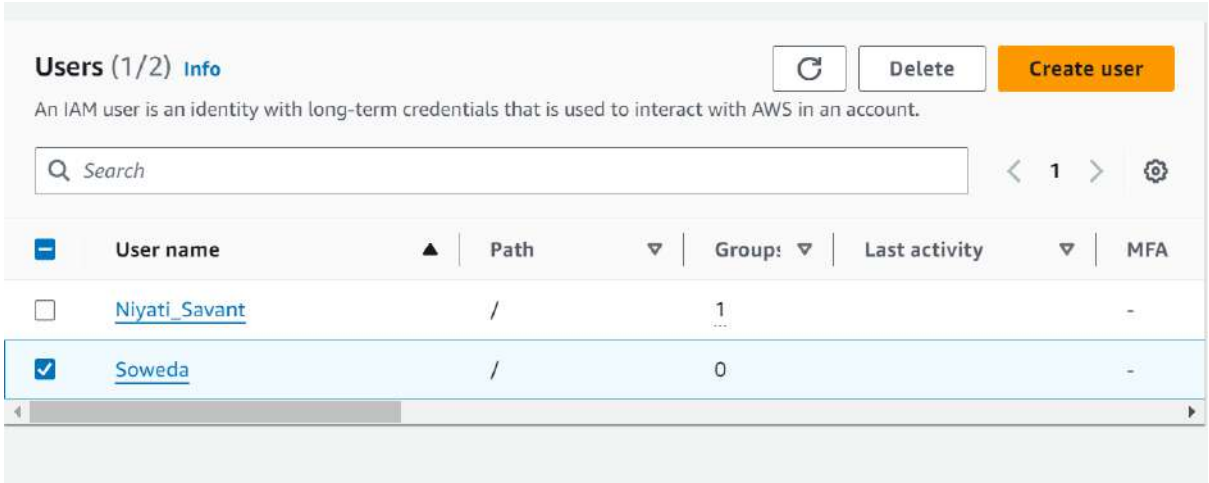
The role has been successfully created.

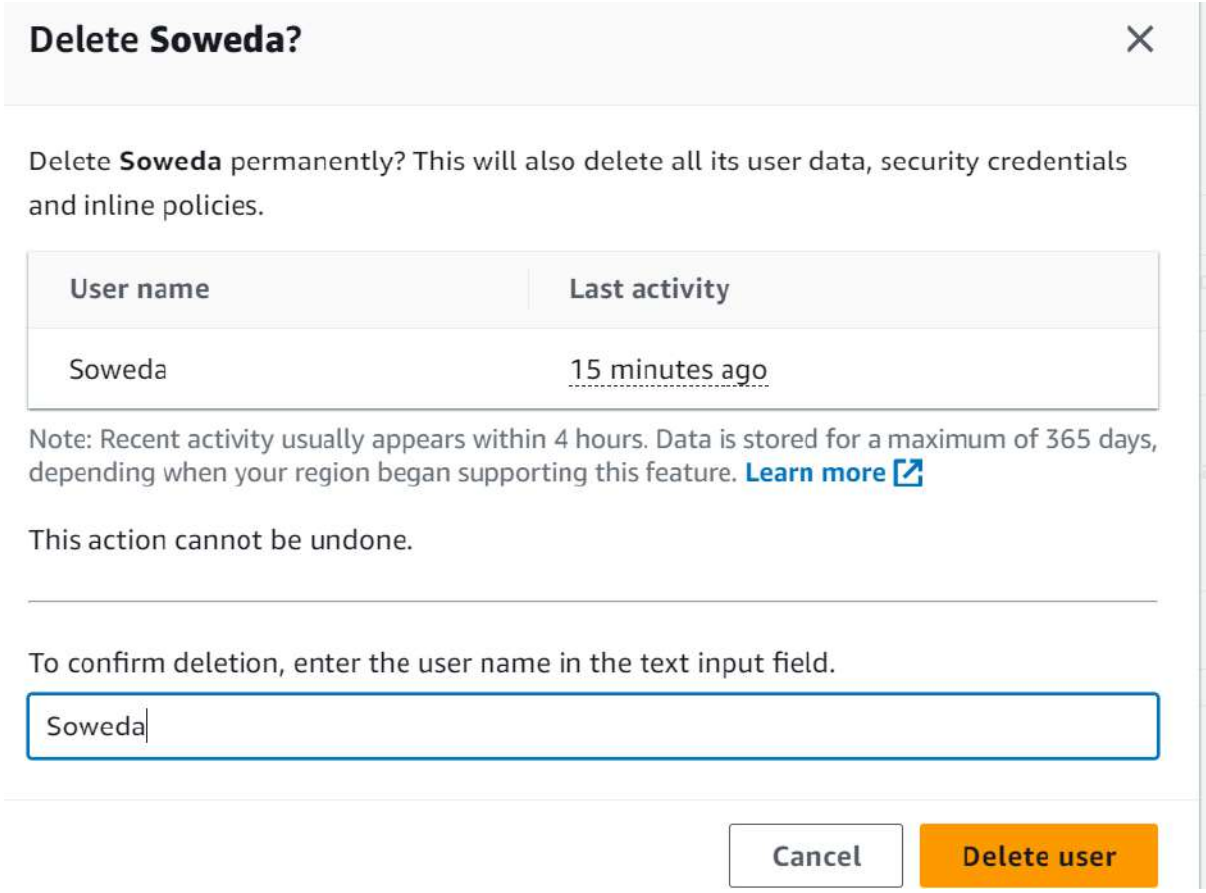


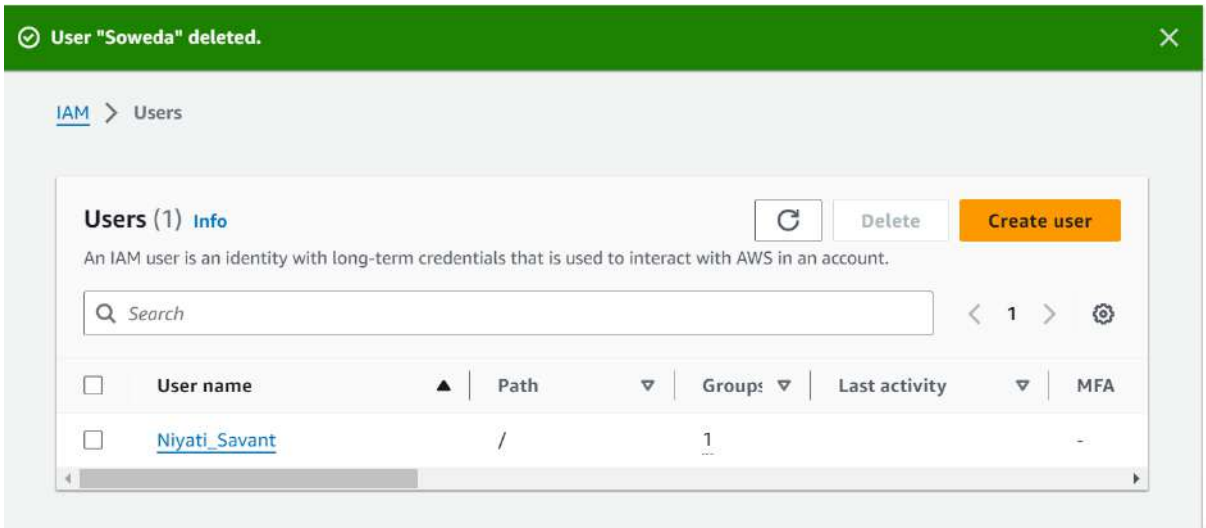
Just to check the overall users, groups and roles, you can check out the IAM Dashboard



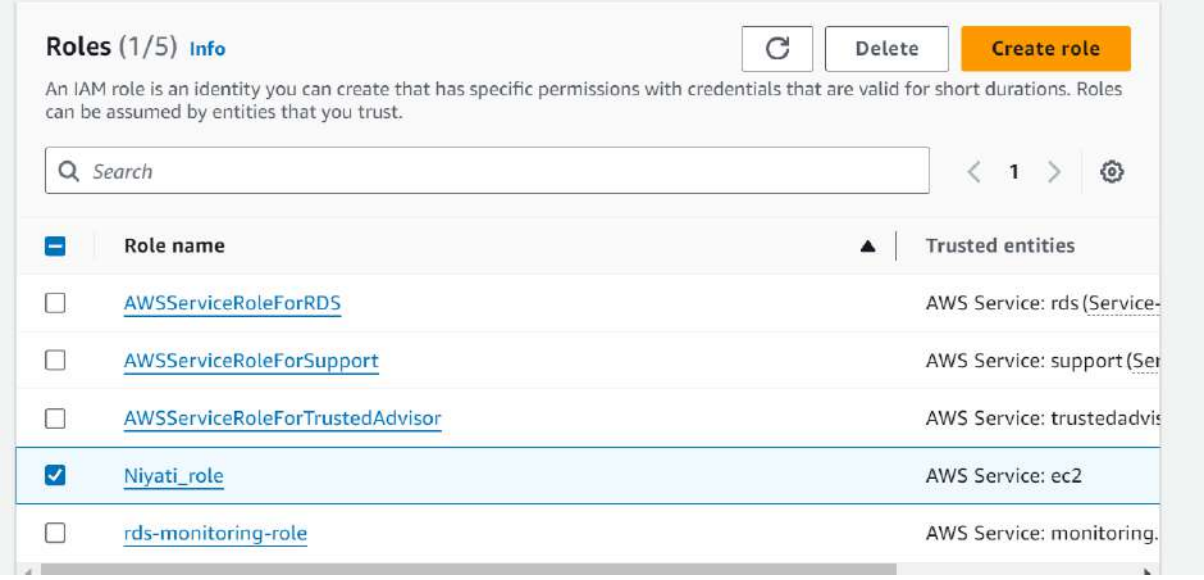
----- Deleting a User -----

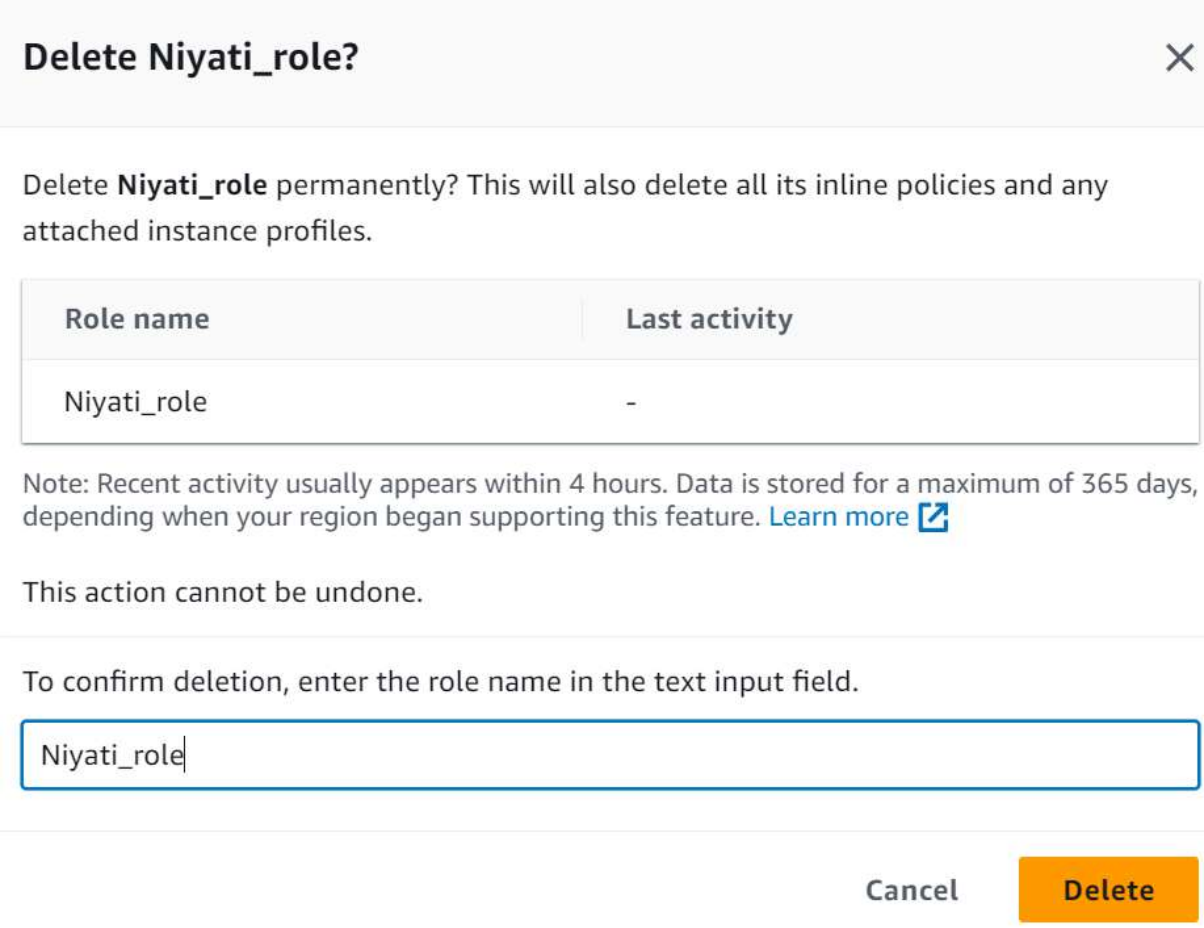




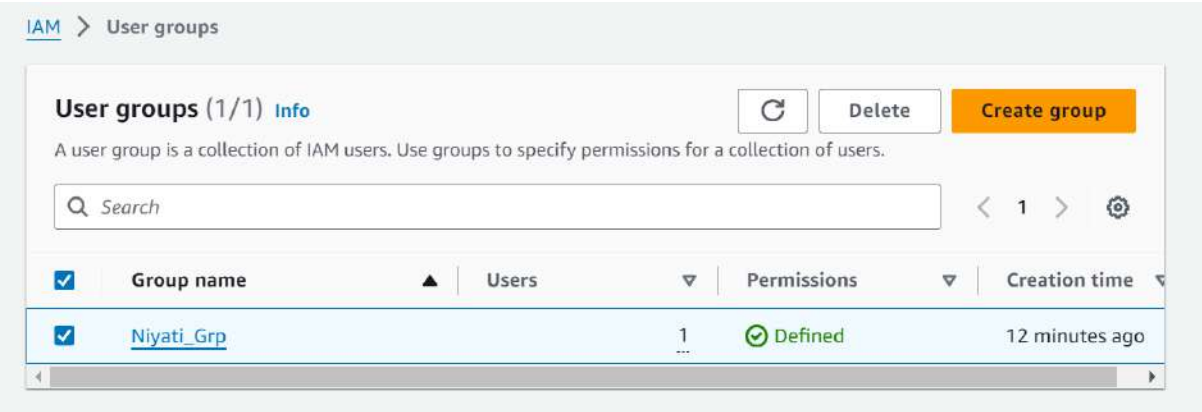


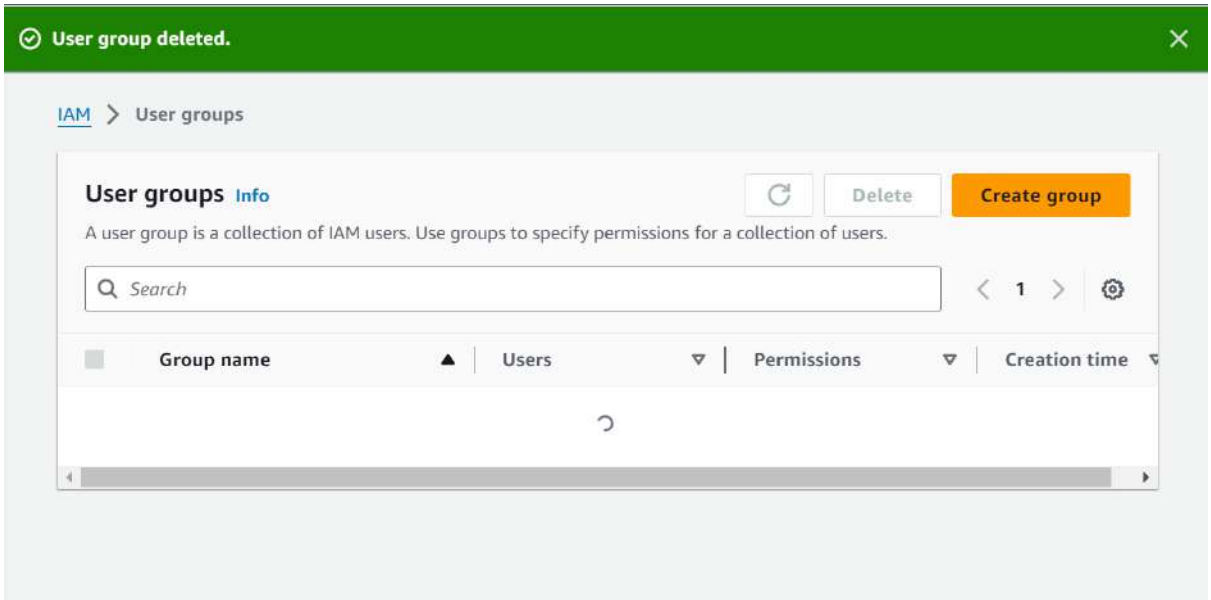
----- Deleting a Role -----



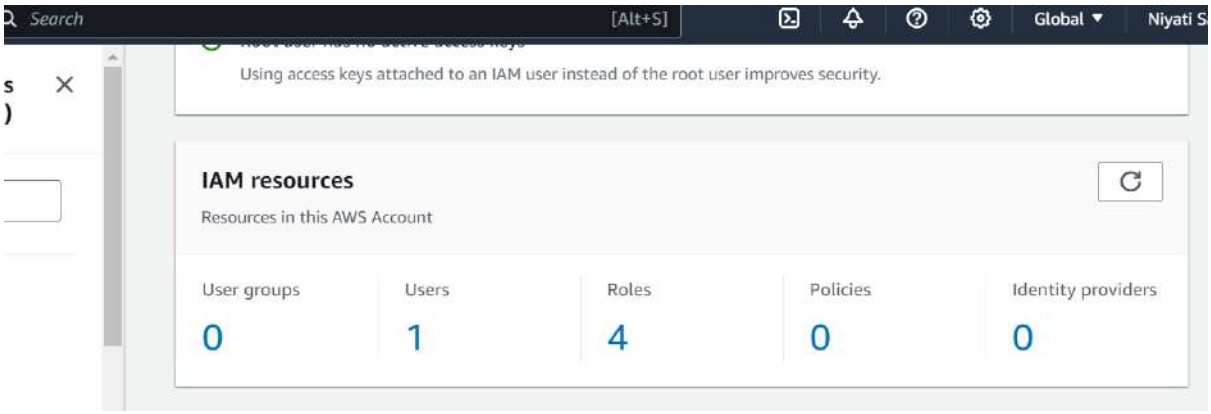


----- Deleting a Group -----

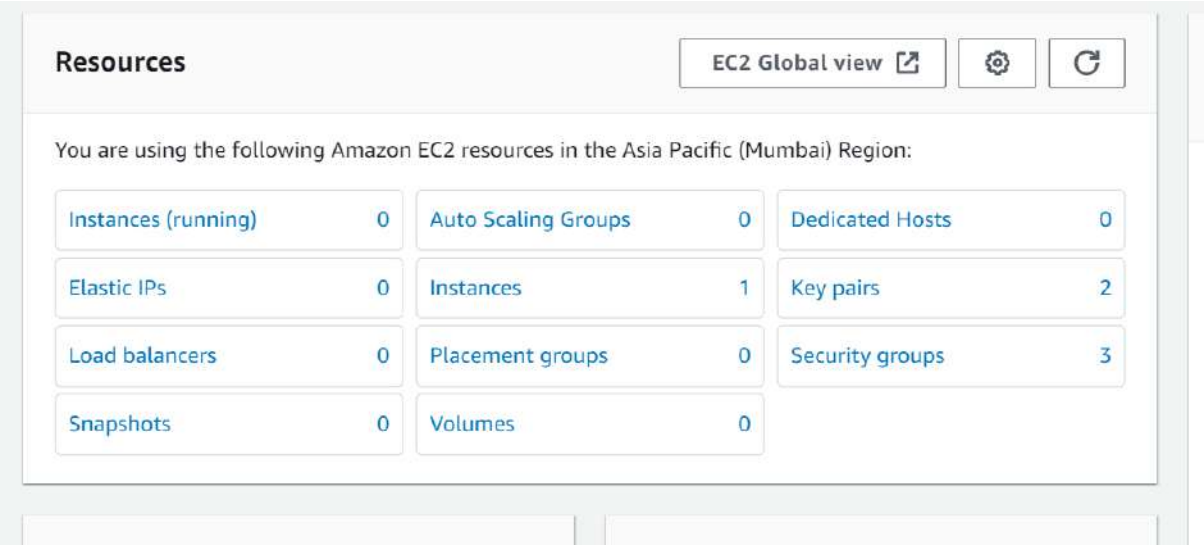




Check the IAM dashboard to see the results after deletion activities



Check the ec2 dashboard in case there are any running instances



Experiment - 8

Aim - To study and implement Database as a service on SQL databases using AWS RDS

Theory -

Database as a Service (DBaaS):
It is a cloud computing service model that provides users with ~~acto~~ access to database without the need for setting up, managing, and maintaining the underlying infrastructure. With DBaaS, users can easily deploy, scale, and manage databases using cloud resources, paying only for the resources they consume. This model offers benefits like reduced operational overhead, scalability, high availability, and automated backups

Some popular DBaaS providers include:
1. Amazon Web Services (AWS) - Amazon RDS, Amazon Aurora
2. Microsoft Azure - Azure SQL Database
3. Google Cloud Platform - Google Cloud SQL
4. IBM Cloud - IBM Db2 on Cloud, IBM Cloud Databases
5. Alibaba Cloud - ApsaraDB for RDS
6. Oracle Cloud - Oracle Autonomous Database
7. Heroku - Heroku Postgres
8. Firebase - Firestore
9. MariaDB - MariaDB SkySQL
10. Percona - Percona Cloud Database

SQL/NOSQL databases supported by AWS include.

1. Amazon RDS - MySQL, PostgreSQL, MariaDB, Oracle, Microsoft SQL Server
2. Amazon Aurora - MySQL and PostgreSQL-compatible
3. Amazon Redshift - Data warehouse
4. Amazon ~~Neput~~ Neptune - Graph database

For NoSQL databases AWS supports:

1. Amazon DynamoDB - Fully managed NoSQL database service
2. Amazon Document DB - MongoDB-compatible database
3. Amazon ElastiCache - In-memory caching service supporting Redis and

The following terms are frequently used in AWS DBaas -

1) Storage types : AWS provides different storage types like General Purpose SSD (gp2), Provisioned IOPS SSD (io1), Magnetic (standard), and Throughput Optimized HDD (st1) for different performance and cost requirements
2) Endpoint : It is a URL that uniquely identifies the entry point for a specific AWS service. In context of DBaas, it represents the network address to connect to the database instance.
3) Snapshot : A point-in-time backup of a database instance. It captures the entire state of the database at the time the snapshot was taken allowing users

restore database to exact state later

4) Read Replica - Copy of primary database instance that allows read-only access. It helps offload read queries from primary instance, improving overall performance and scalability.

5) Single AZ and Multi-AZ instances - Single AZ instances run in single Availability Zone, while MultiAZ instances are deployed across multiple Availability Zones for high availability and fault tolerance. In such deployments, AWS automatically replicates data to a standby instance in a different AZ.

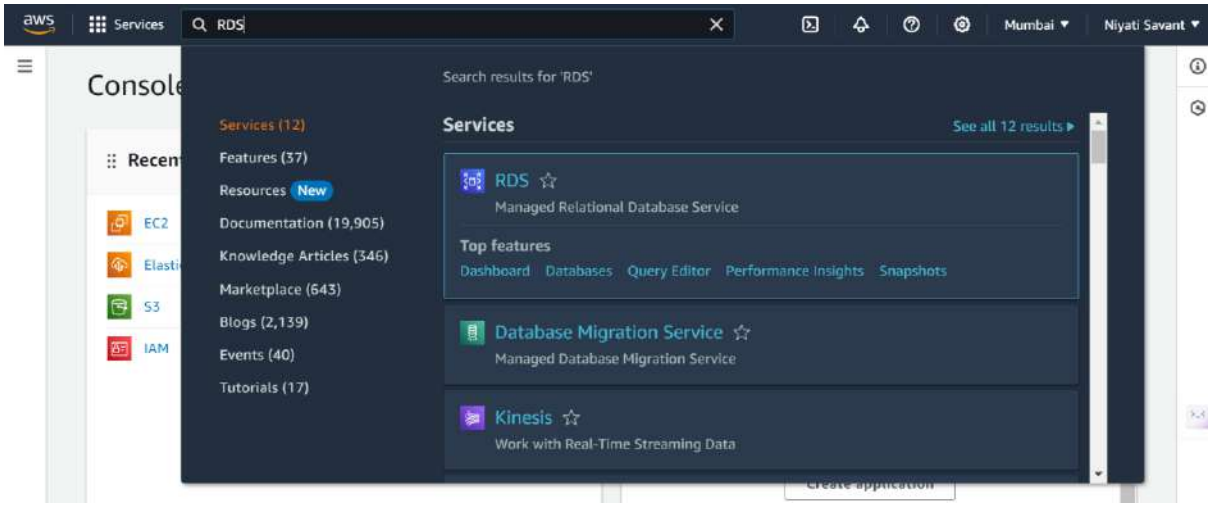
Difference between RDS and Aurora

Features	Aurora	RDS
Database Engine support	MySQL and PostgreSQL	MySQL, PostgreSQL, MariaDB.
Design	Reliable and fault tolerant	For reliability, must manually enable it
Durable	Better due to multiple copies	Less durable
Replication service	15 replicas	5 replicas
Fail Over to read replica	Serve query immediatly Automatic	Slower Manual
Storage	Automatic scaling (inc of 10GB)	Auto scaling (Max 16TiB)
Pricing	Exhensive	Lesser

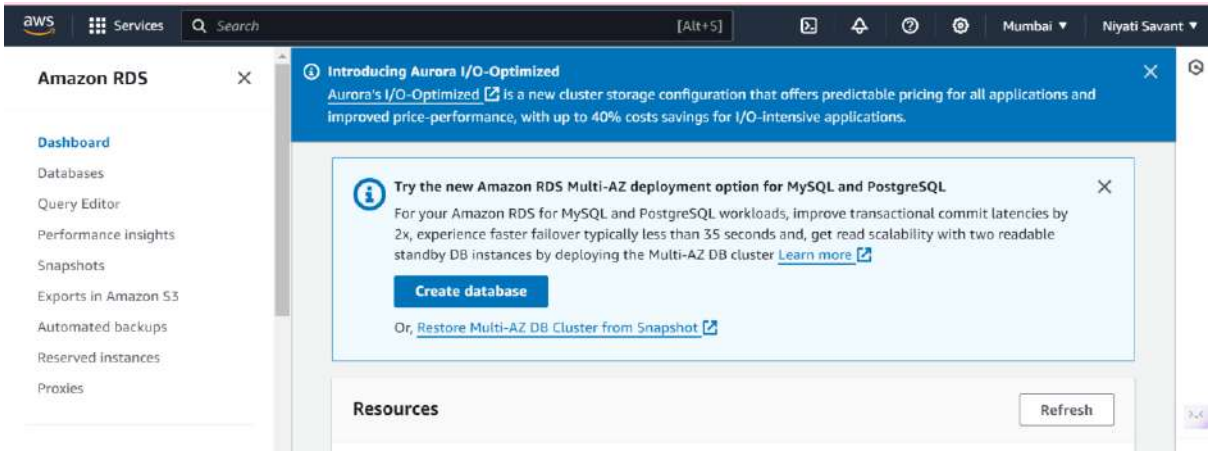
18/3/24 (18)

EXPERIMENT 8 - Implement Database as a Service on SQL databases Using AWS RDS.

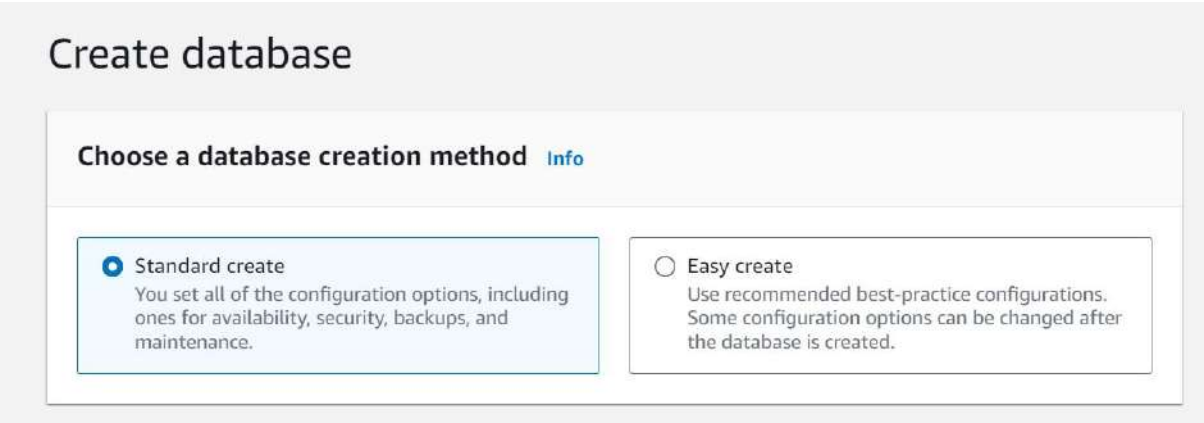
Step1 : Login to AWS console and search RDS



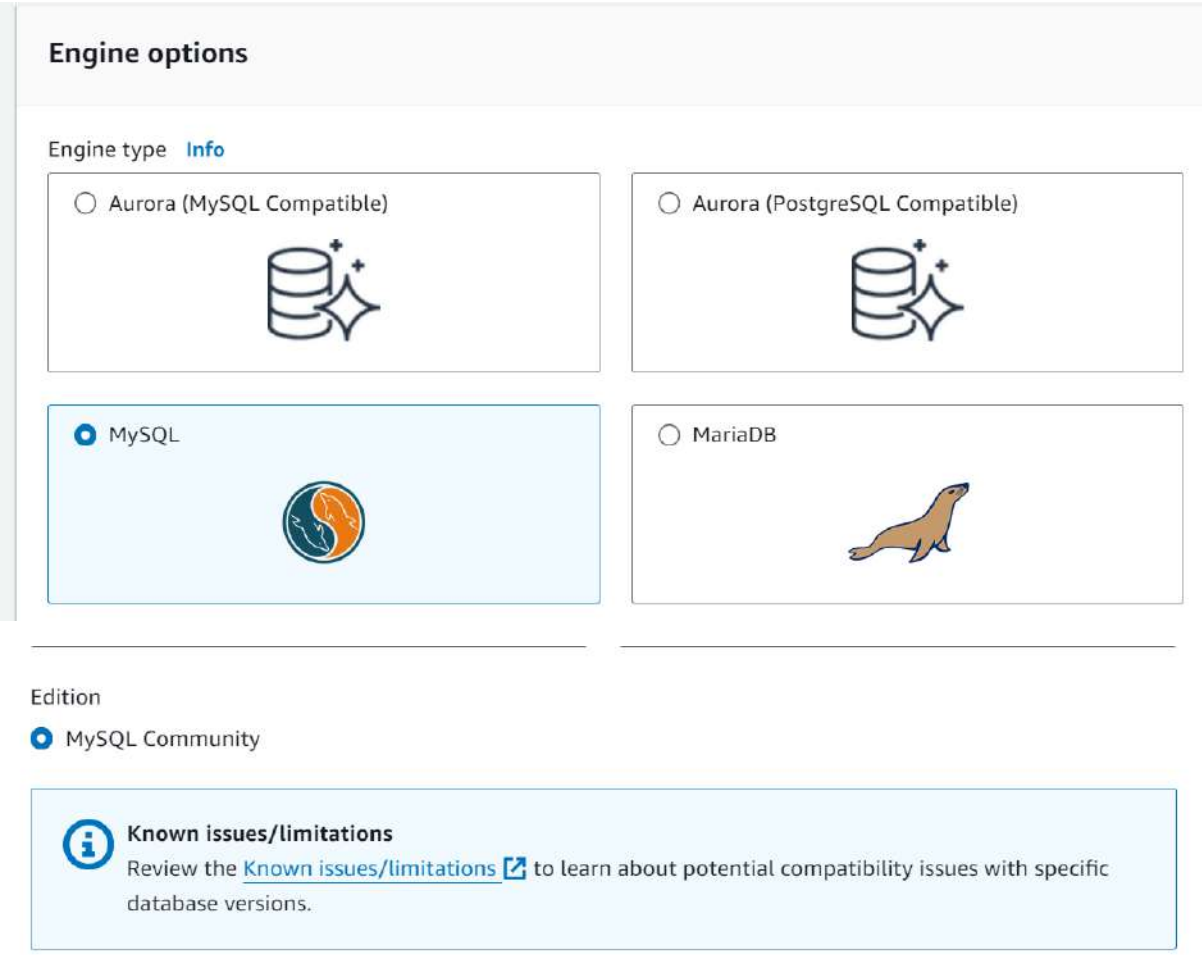
Step2: Click on to RDS and create database



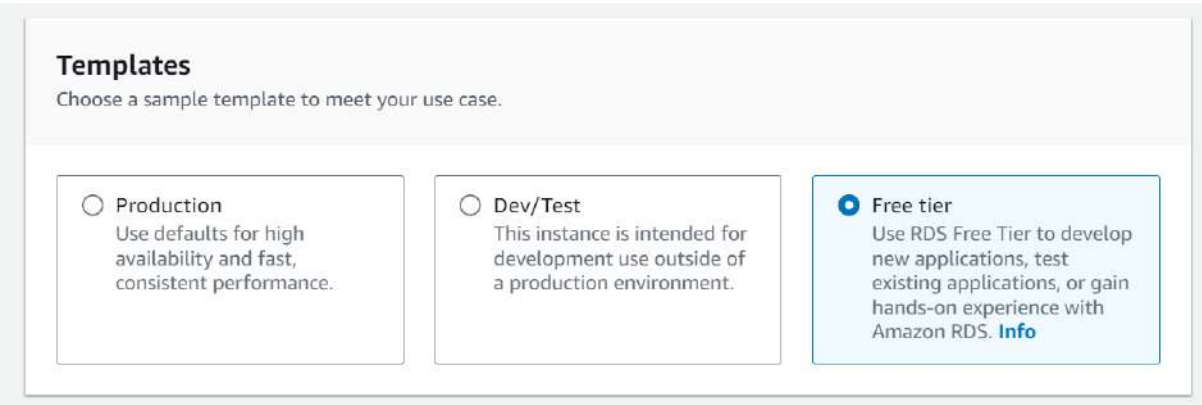
Step 3: Select standard database



Step 4: Select MySQL and MySQL Community edition



Step 5: In Templates select Free tier



Step 6: Mention database name (default is database1) and username and password

DB instance identifier Info

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

databaseNiyati

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username Info

Type a login ID for the master user of your DB instance.

admin

1 to 16 alphanumeric characters. The first character must be a letter.

☐ Manage master credentials in AWS Secrets Manager

Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

Step 7: Instance is t2.micro

Services Search [Alt+S]

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class Info

▼ Hide filters

Show instance classes that support Amazon RDS Optimized Writes Info

Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Include previous generation classes

Standard classes (includes m classes)

Memory optimized classes (includes r and x classes)

Burstable classes (includes t classes)

db.t2.micro

1 vCPUs 1 GiB RAM Not EBS Optimized

• Rest of things keep default

Storage

Storage type Info
Provisioned IOPS SSD (io2) storage volumes are now available.

General Purpose SSD (gp2)
Baseline performance determined by volume size

Allocated storage Info

20 GiB

The minimum value is 20 GiB and the maximum value is 6,144 GiB

After you modify the storage for a DB instance, the status of the DB instance will be in storage-optimization. Your instance will remain available as the storage-optimization operation completes.
Learn more

▶ Storage autoscaling

▼ Storage autoscaling

Storage autoscaling Info

Provides dynamic scaling support for your database's storage based on your application's needs.

☑ Enable storage autoscaling
Enabling this feature will allow the storage to increase after the specified threshold is exceeded.

Maximum storage threshold Info
Charges will apply when your database autoscales to the specified threshold

1000 GiB

The minimum value is 22 GiB and the maximum value is 6,144 GiB

Connectivity Info

Compute resource

Choose whether to set up a connection to a compute resource for this database. Setting up a connection will automatically change connectivity settings so that the compute resource can connect to this database.

- (●) **Don't connect to an EC2 compute resource**
  Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.
- ( ) **Connect to an EC2 compute resource**
  Set up a connection to an EC2 compute resource for this database.

Virtual private cloud (VPC) Info

Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Default VPC (vpc-021cf3b0158ac2b4b)
3 Subnets, 3 Availability Zones

Only VPCs with a corresponding DB subnet group are listed.

ⓘ After a database is created, you can't change its VPC.

Step 8: Select Public Access -Yes

Public access Info

- (●) **Yes**
  RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.
- ( ) **No**
  RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) Info

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

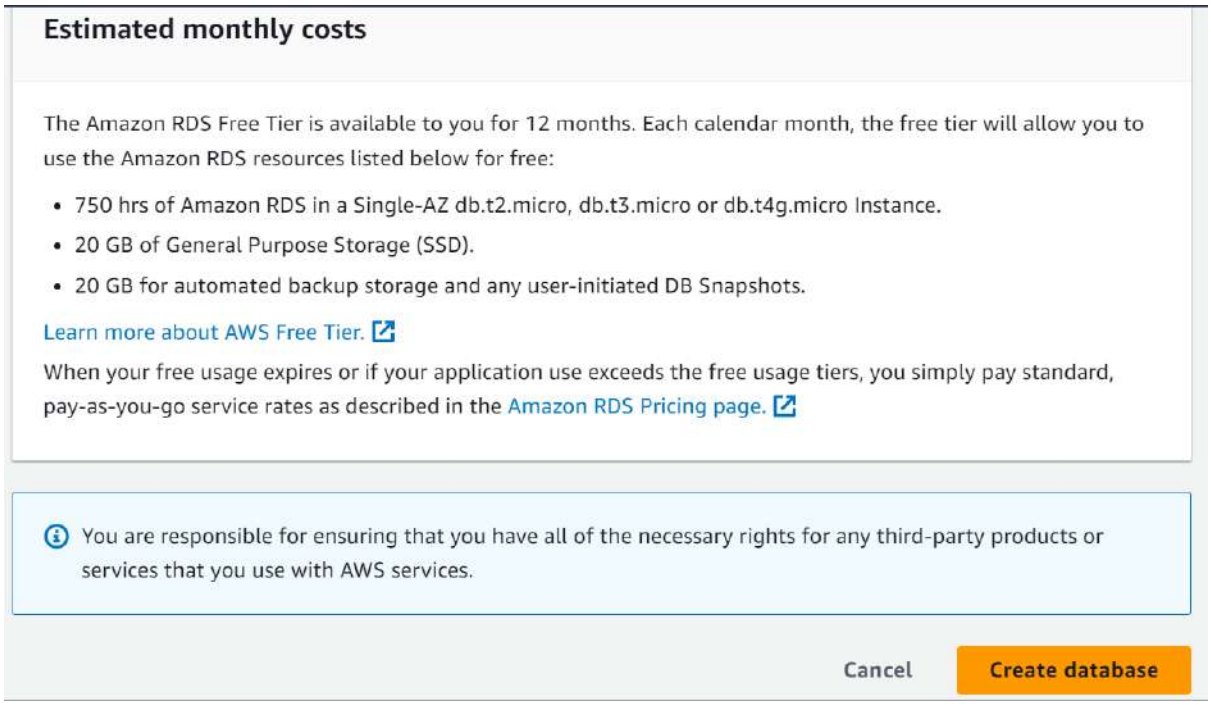
- (●) **Choose existing**
  Choose existing VPC security groups
- ( ) **Create new**
  Create new VPC security group

Database authentication

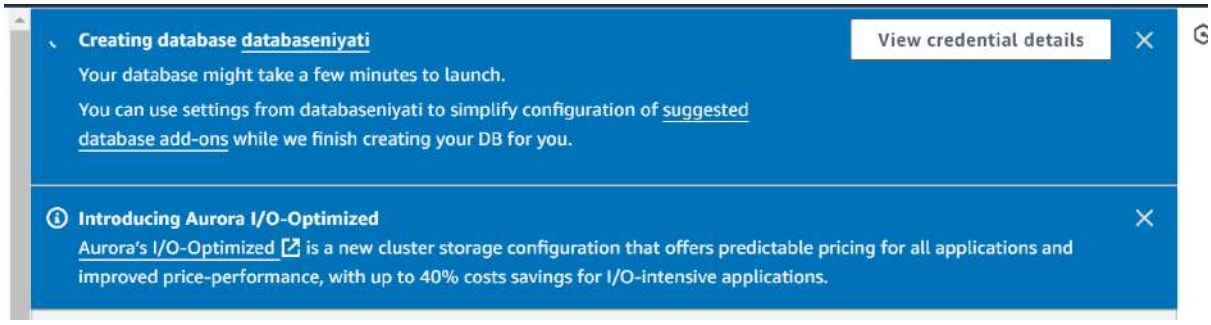
Database authentication options Info

- (●) **Password authentication**
  Authenticates using database passwords.
- ( ) **Password and IAM database authentication**
  Authenticates using the database password and user credentials through AWS IAM users and roles.
- ( ) **Password and Kerberos authentication**
  Choose a directory in which you want to allow authorized users to authenticate with this DB instance using Kerberos Authentication.

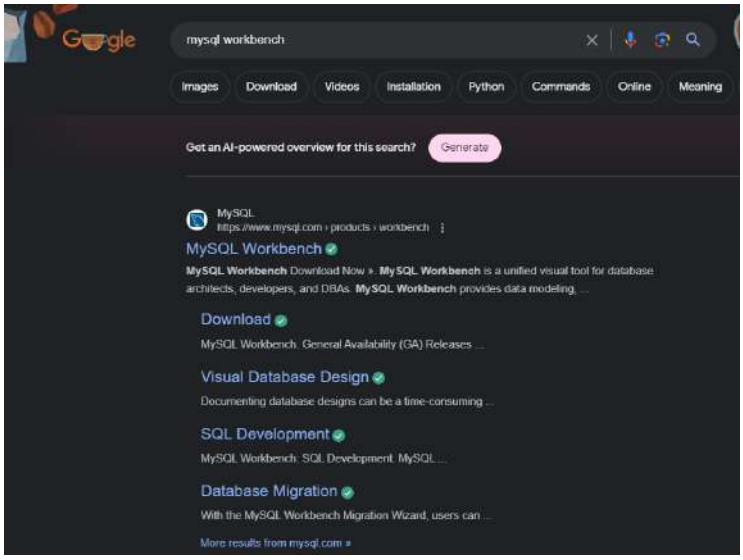
Step 9: Click on to create Database



Step 10: It will take some time

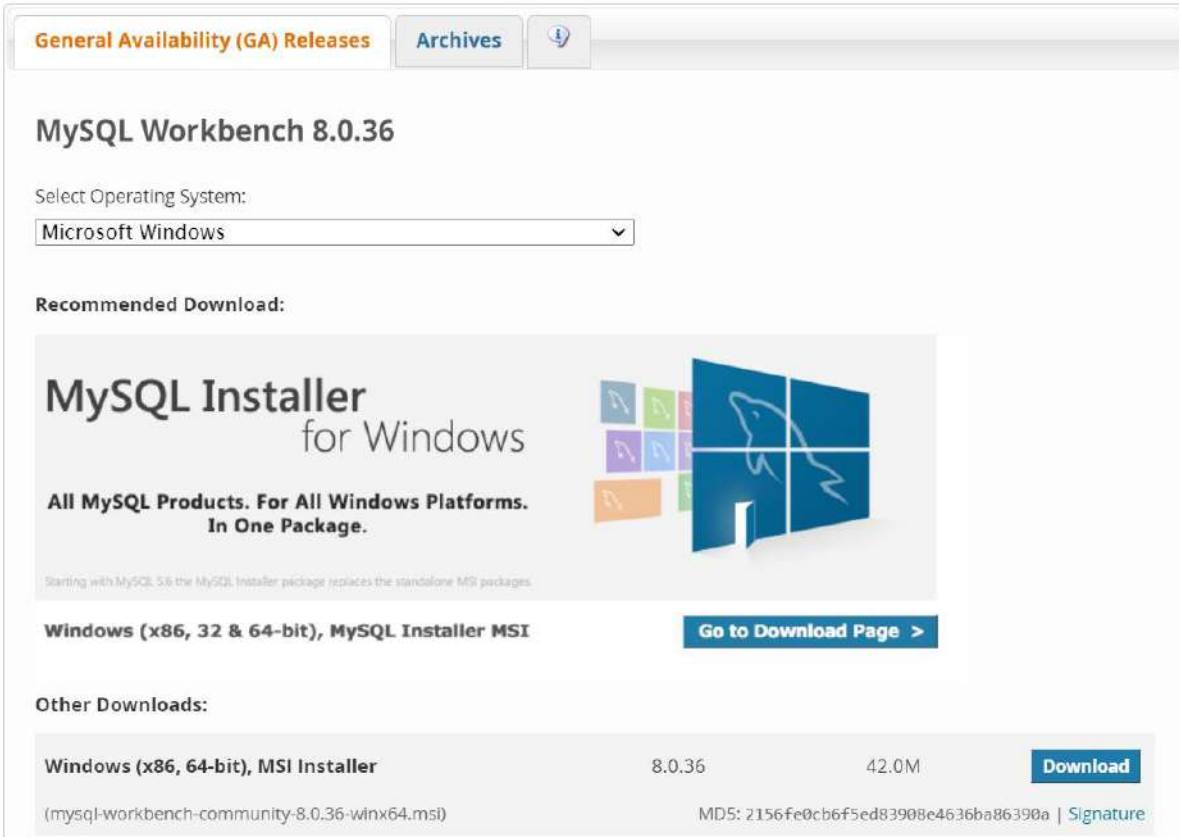


Step 11: Go to Google and type 'mysql workbench'

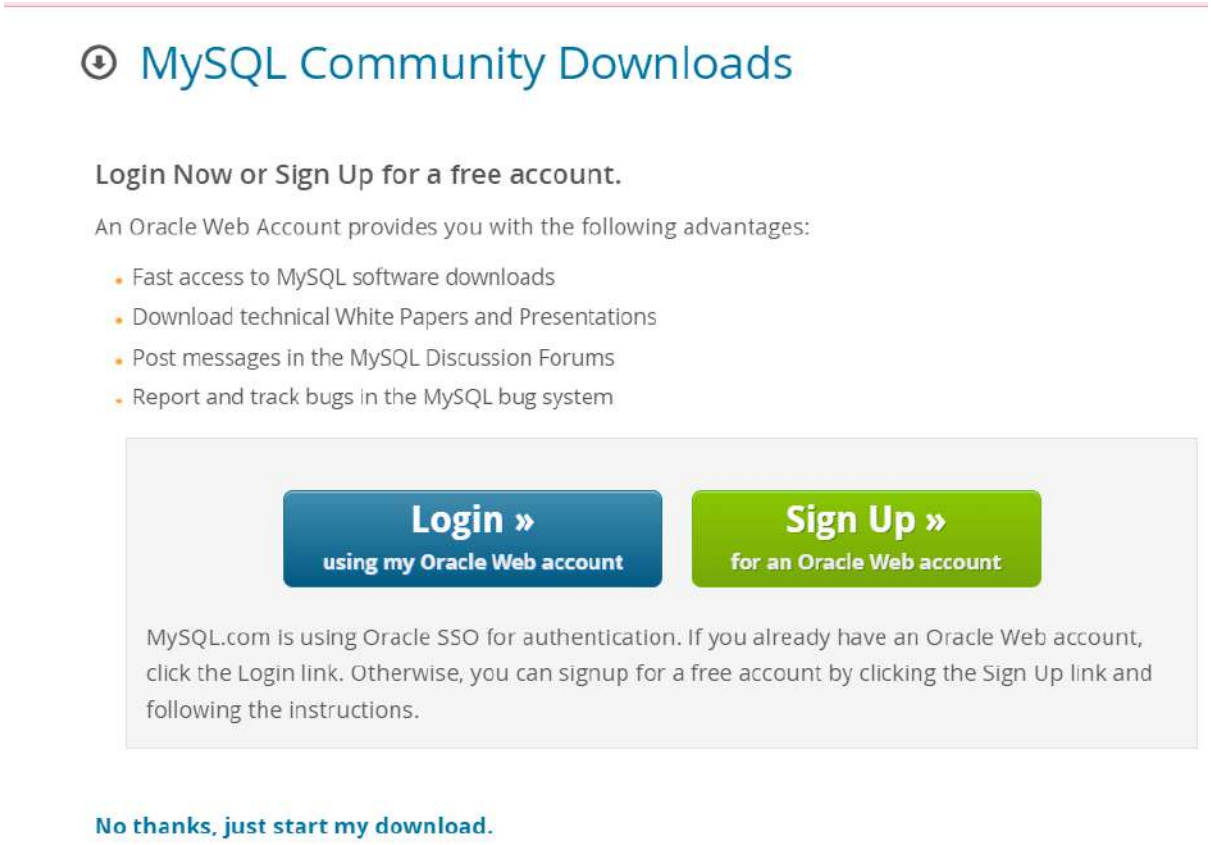


Step 12: Click on to download

Step 12: MySQL community download – Microsoft Windows



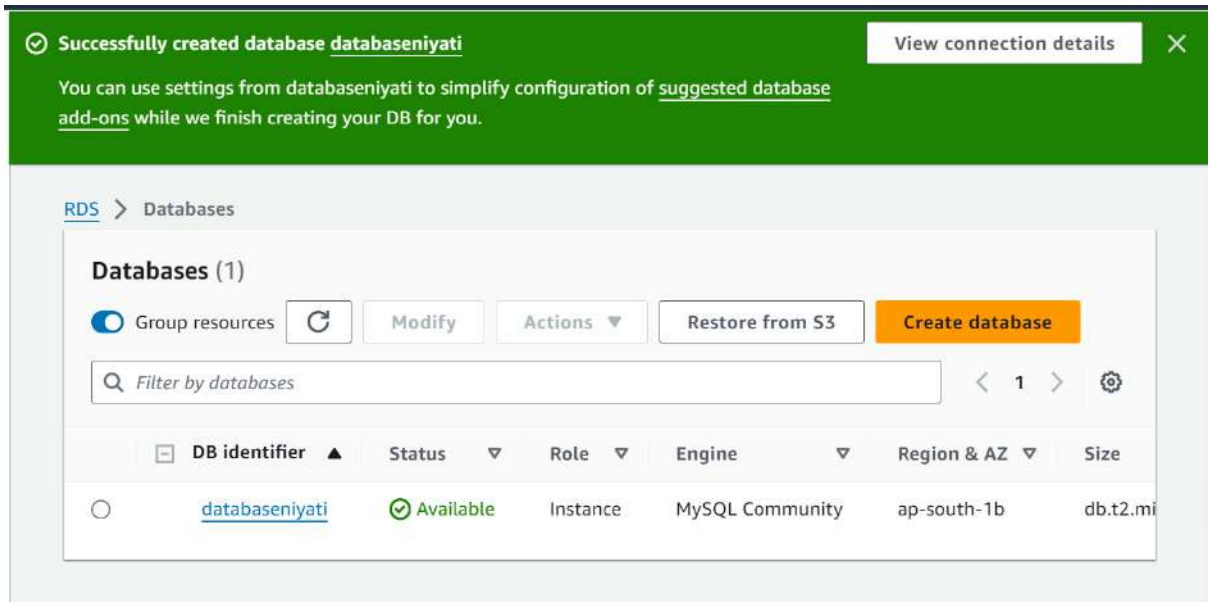
Step 13: Click on to – No thanks , just download



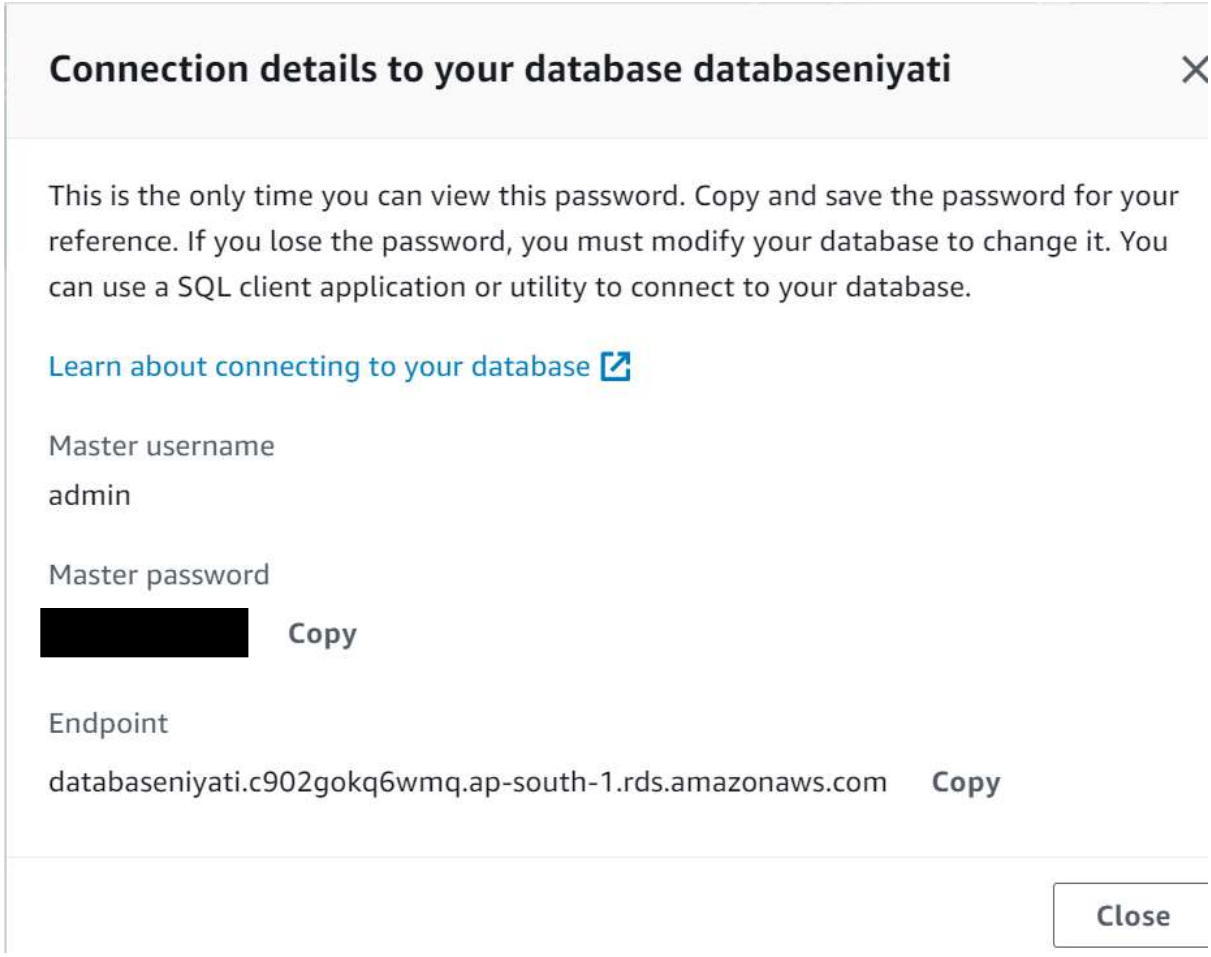
Step 14: Go to downloads of your machine and install it with default settings

Step 15: Check your database is created and status is available. Click on to view credential

Step 16: Click on to database

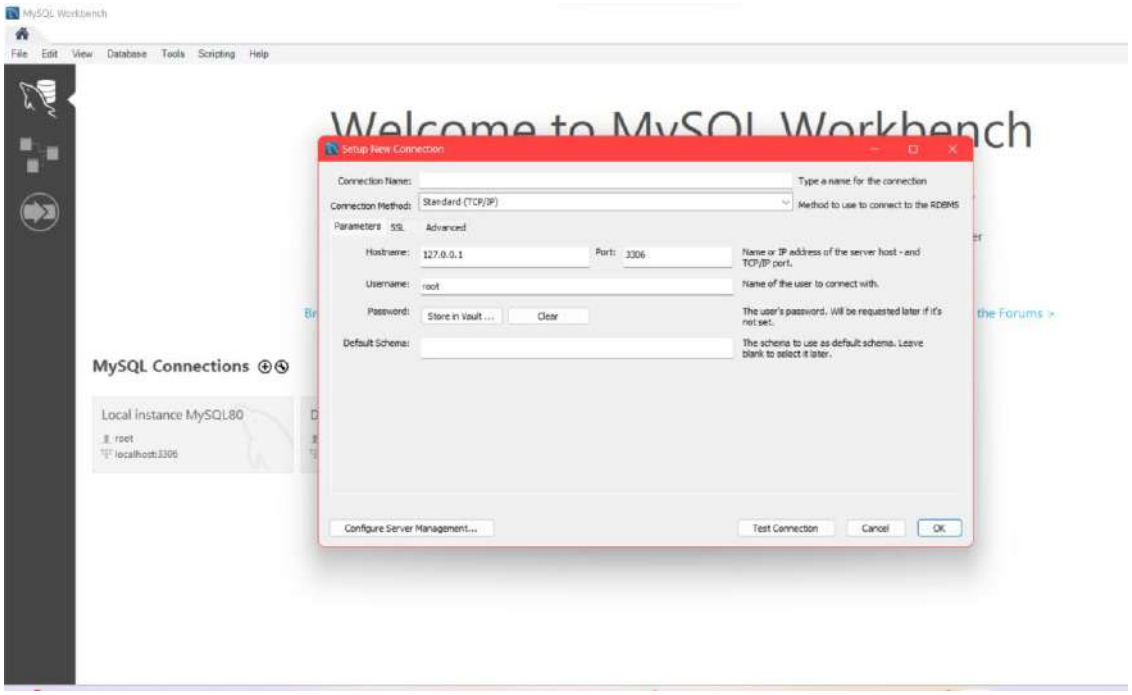


Step 17: Copy Endpoint



Step 18: Go back to workbench.

Step 20: Click on to mysql connection



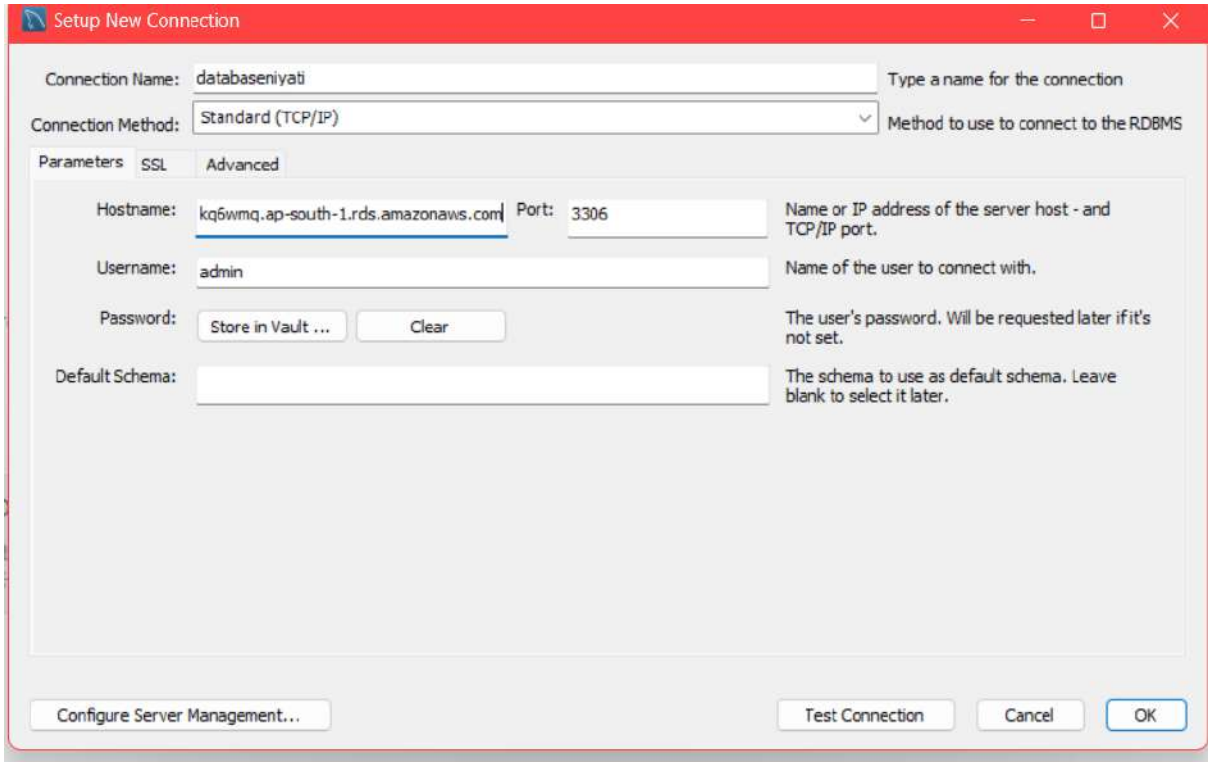
Step 21:

Paste the copied endpoint in Hostname .

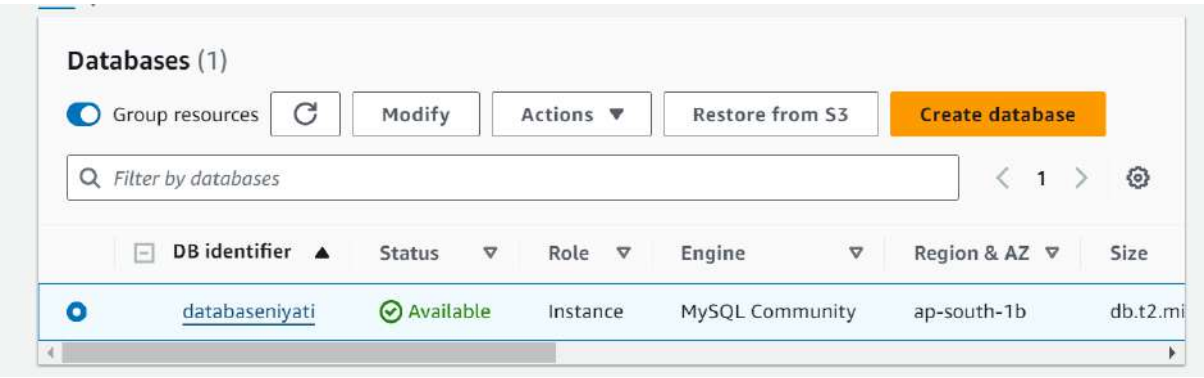
Connection Name : databaseniyati

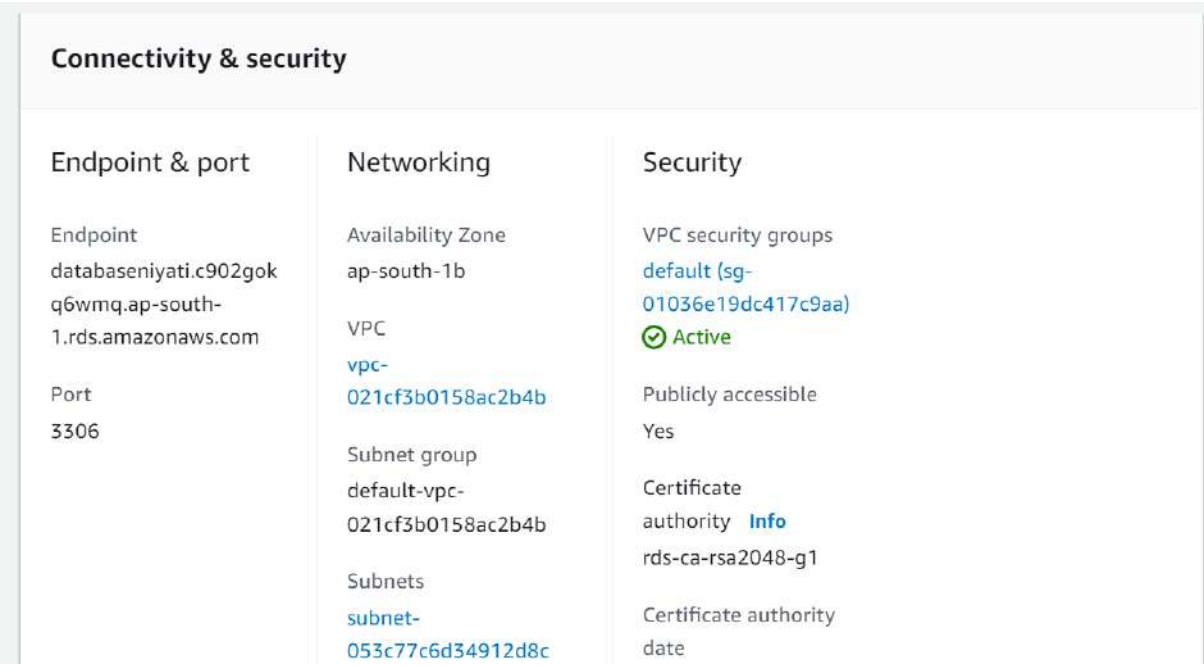
Username : admin

Click on to Test Connection

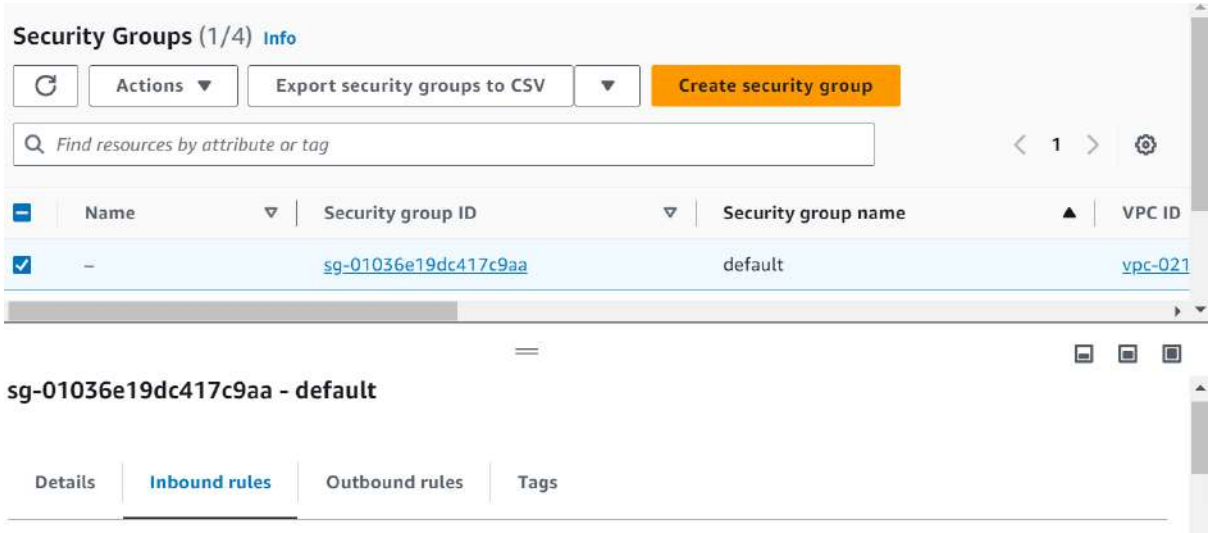


Step 22: Go to vpc security group

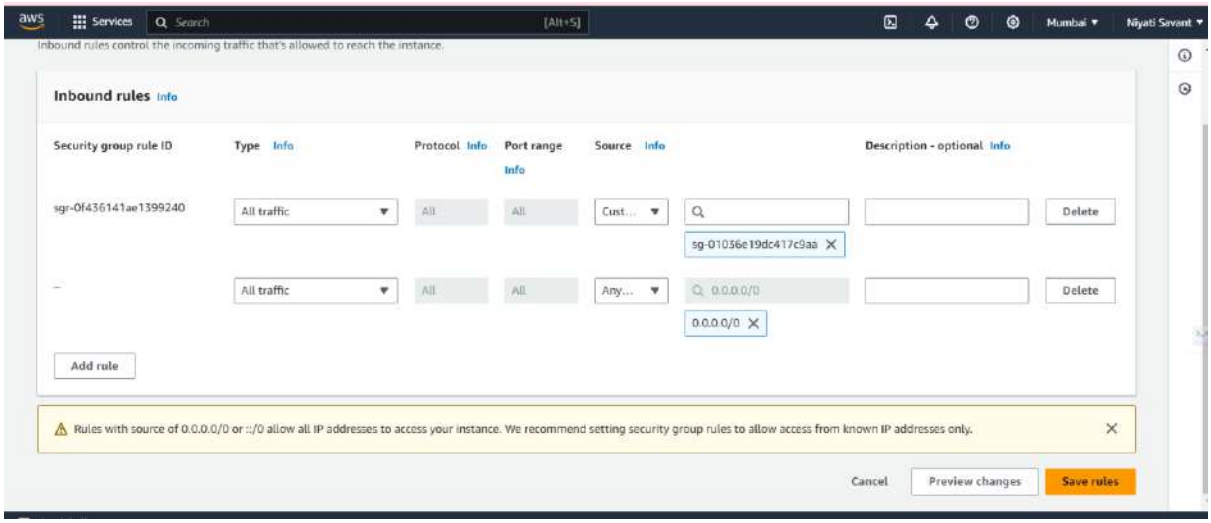




Step 23: Click on to inbound rules



Step 24: First select Click on to Edit inbound rule add rule select ipv4 --all traffic (add 0.0.0.0./0) and save Rules.



Step 22: Goto workbench (after giving details click on to Test Connection)

Click on Ok button

Go to workbench double click on connection(databaseniyati)

It will get opened

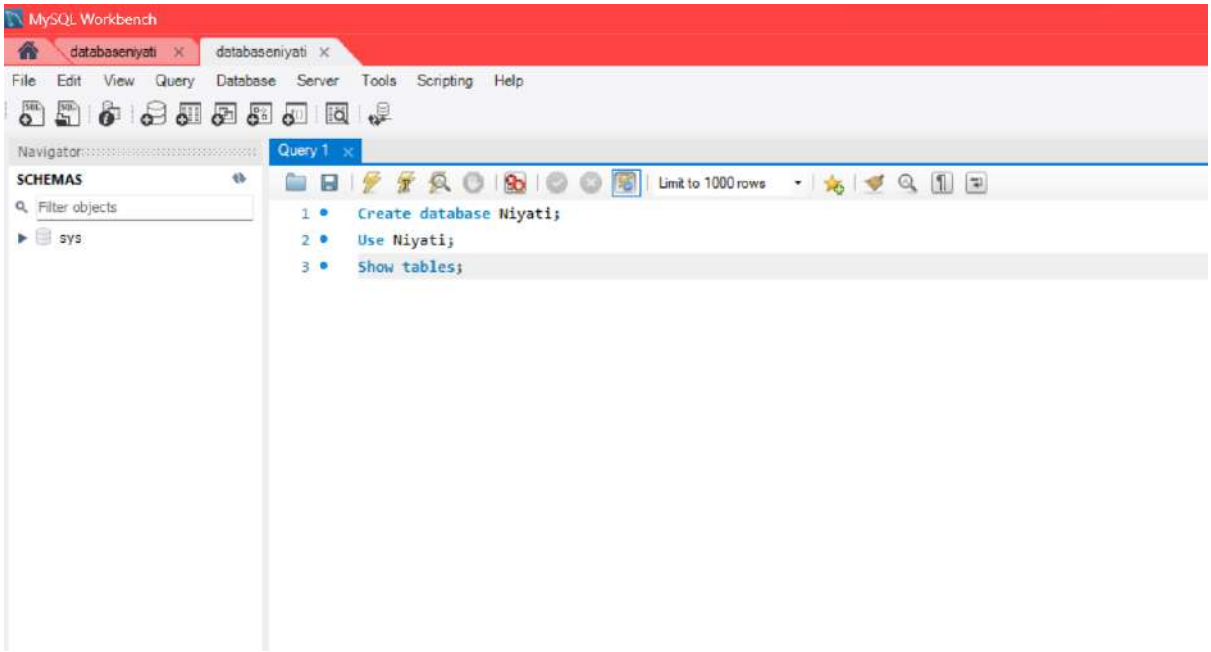
Step 23:

Write query and execute

Create database Niyati;

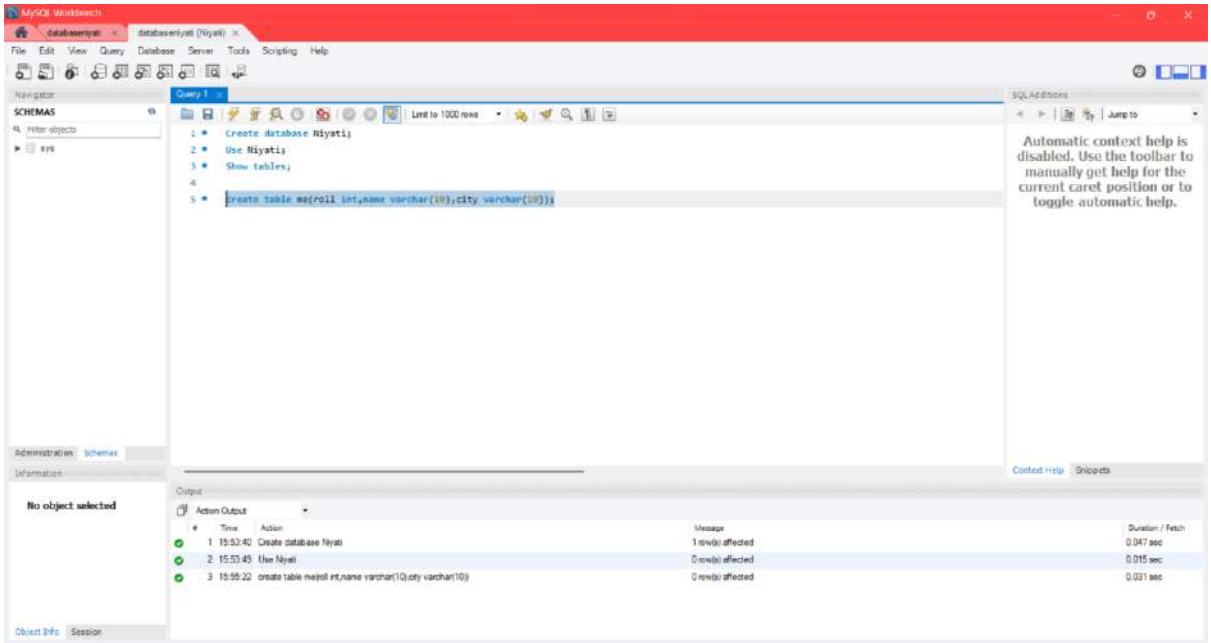
Use Niyati;

Show tables;



Create table for eg:

create table me(roll int, name varchar(10), city varchar(10));



Describe me;

Niyati_Savant_TE_C31_2103156

```
create table me(roll int,name varchar(10),city varchar(10));
desc me
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
roll	int	YES		NULL	
name	varchar(10)	YES		NULL	
city	varchar(10)	YES		NULL	

Result 1

Output

Action Output

#	Time	Action	Message
1	15:53:40	Create database Niyati	1 row(s) affected
2	15:53:49	Use Niyati	0 row(s) affected
3	15:55:22	create table me(roll int,name varchar(10),city varchar(10))	0 row(s) affected
4	15:56:04	desc me	3 row(s) returned

insert into student values(156,'Niyati','Kharghar');

```
insert into me values(156,"Niyati","Kharghar");
select * from me;
```

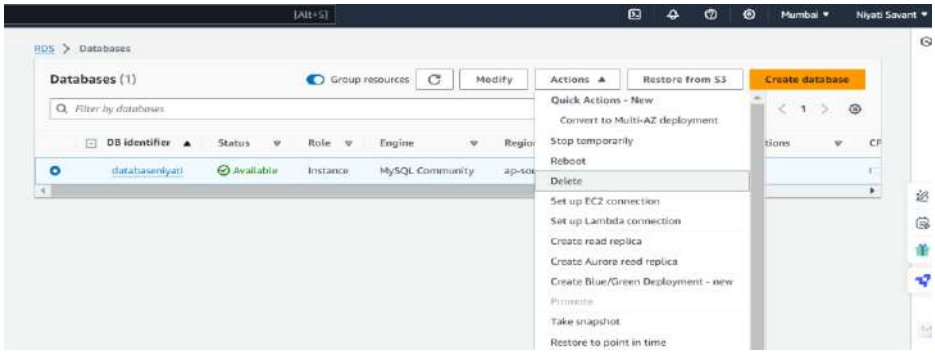
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

roll	name	city
156	Niyati	Kharghar

me 2

Step 24:

Now delete the instance. Select instance go to action stop instance and then delete instance



Uncheck create final shapshot

Permanently delete **databaseniyati** DB instance. You can't undo this action.

⚠ Proceeding with this action will delete the instance with all its content and can affect related resources. Learn more

☐ Create final snapshot
Determines whether a final DB Snapshot is created before the DB instance is deleted.

☐ Retain automated backups
Determines whether retaining automated backups for 1 day after deletion

☑ I acknowledge that upon instance deletion, automated backups, including system snapshots and point-in-time recovery, will no longer be available.

To avoid accidental deletion provide additional written consent.

To confirm deletion, type *delete me* into the field.

delete me

⚠ We strongly recommend taking a final snapshot before instance deletion since after your instance is deleted, automated backups will no longer be available.

Cancel Delete

Experiment - 9

Aim - To study and implement Containerization using Docker.

Theory -

Docker and its architecture

Docker makes use of a client-server architecture. The Docker client talks with the docker daemon which helps in building, running, and distributing docker containers. The Docker client runs with daemon remotely - with REST API over a UNIX socket or a network, the docker client and daemon interact with each other.

The architecture ~~has~~ is client-server based

The Docker client talks with the docker daemon which helps in building, running, and distributing docker containers.

The client runs with the daemon on same system or we can connect Docker client with daemon remotely with REST API over a UNIX socket or a network the client and daemon interact with each other.

Daemon manages all services by communication with other daemon. It manages docker objects like images, containers, networks and volumes with API requests.

Benefits of Containerization

Containerzation offers several advantages -

1) Portability - Containers encapsulates application and their dependencies, making them portable across different environments, from the development to production
2) Isolation - Containers provide isolated environments for applications, ensuring that changes or dependencies or container do not affect others
3) Scalability - Can be easily scaled up or down as per demand, allowing for efficient resource utilization

Explain the following w.r.t Docker

1) Containers - A Container is a lightweight, standalone, executable package that includes everything needed to run a piece of software including code, runtime, system tools, libraries and setting.
2) Images - Images are read only templates used to create containers. They contain application code, dependencies and other necessary files and metadata required to run the app.
3) Docker File - A text file that contains instruction for building a Docker image. It specifies the base image, environment variables commands to during the image build process and other configuration setting necessary to create the image.

Compare the following

1) Container & Virtual Machine

Virtual Machine	Container
1) Application running on VM/hypervisor can run on different OS.	1) While application is running in container environment share a single OS.
2) VM virtualizes the computer system ie the hardware	2) Container virtualizes the OS or software only
3) VM size is very large, generally in GB	3) While size of container is very light.
4) Eg - Type 1 hypervisor like KVM, Xen etc.	4) Eg : Photon OS, Rancher's OS etc.

2) Image and container

Image	Container
1) A blue print of the container	1) Instance of the image
2) Logical Entity	2) Real world entity
3) Created only once	3) Created any no. of times using an image
4) Sharing docker image is possible	4) Sharing container is not exactly possible

28/3/24 (A)

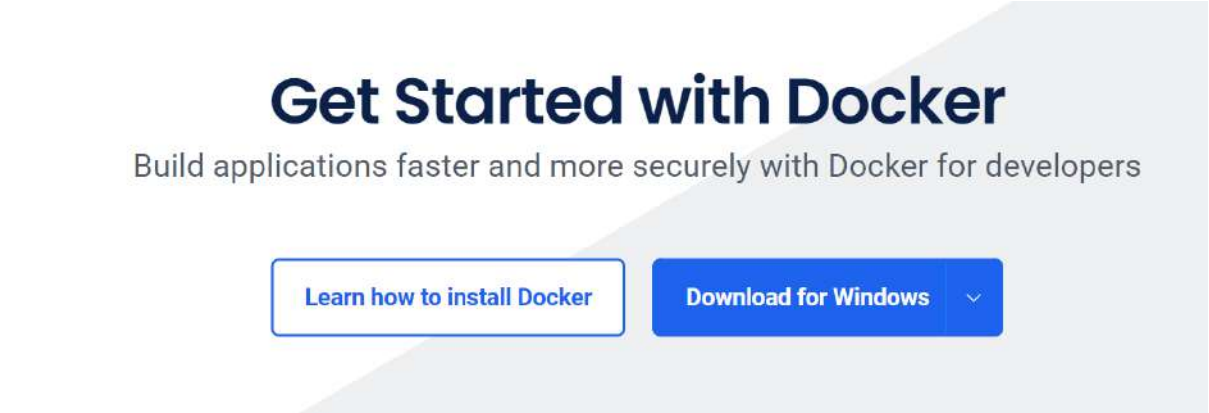
EXPERIMENT 9

To Study and implement containerization using DOCKER.

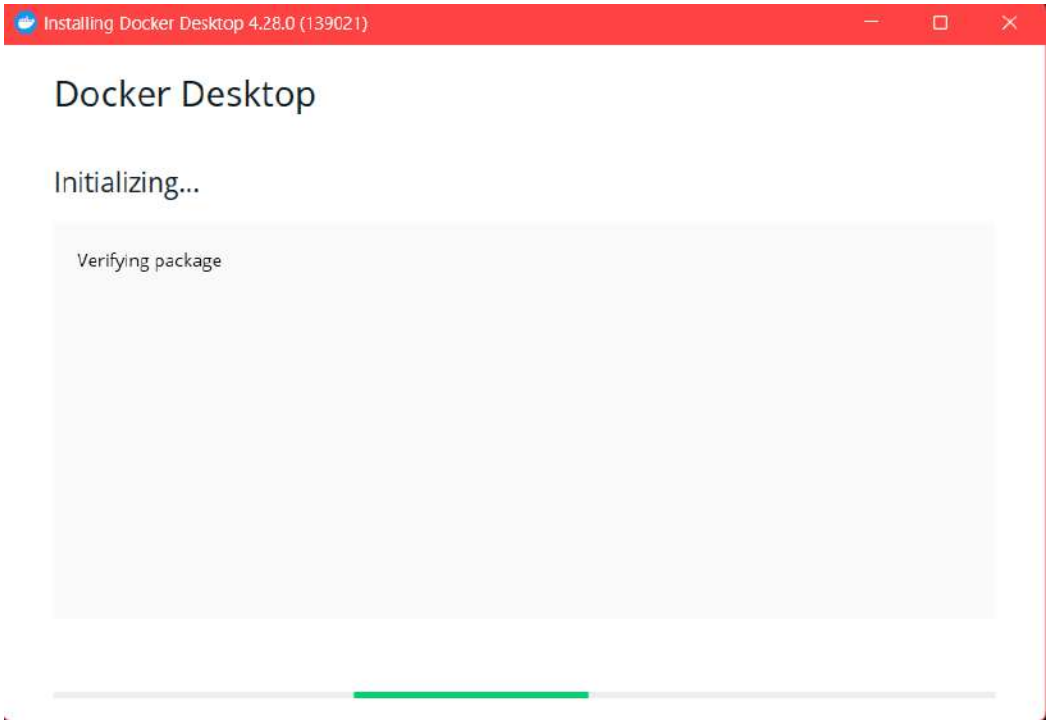
Step 01: Open docker.com

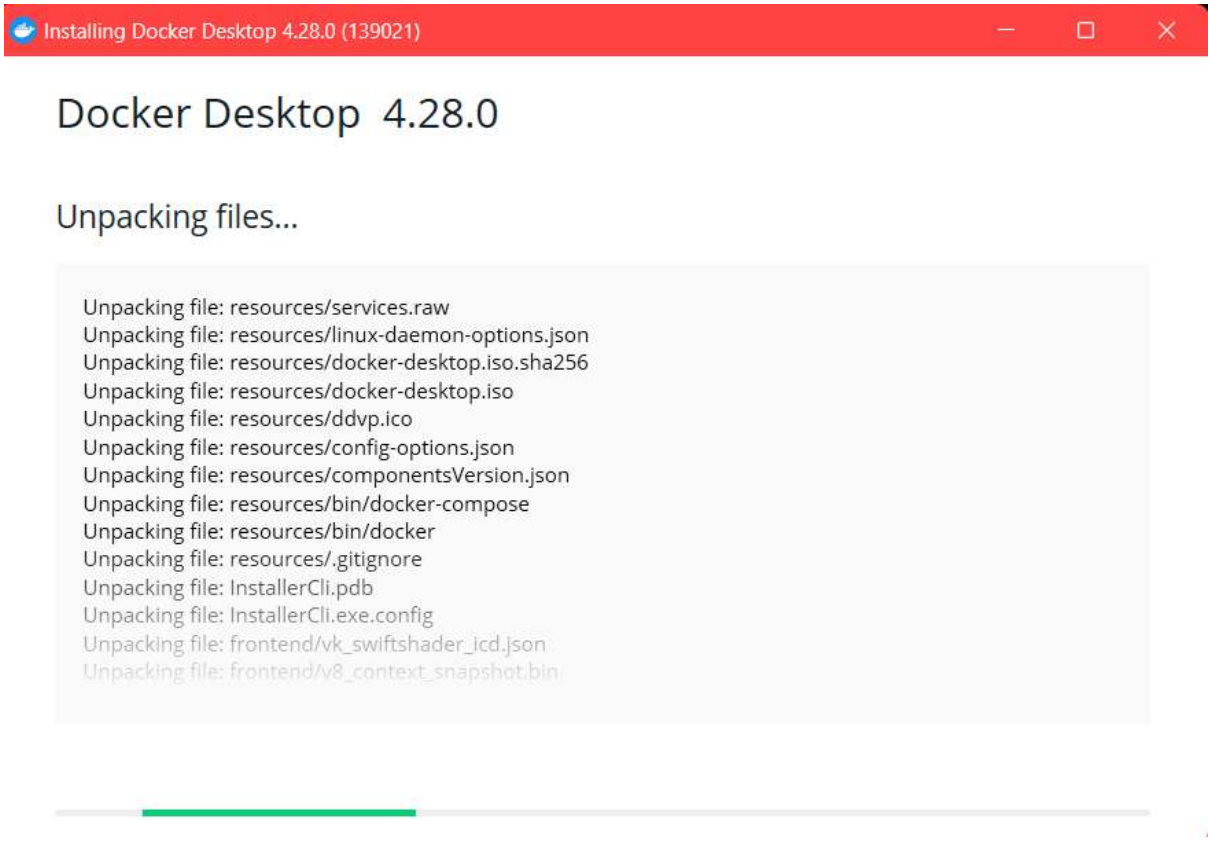
Scroll down, Click on 'Get started for free' tab.

Step 02: Click on Docker Desktop, Download it.

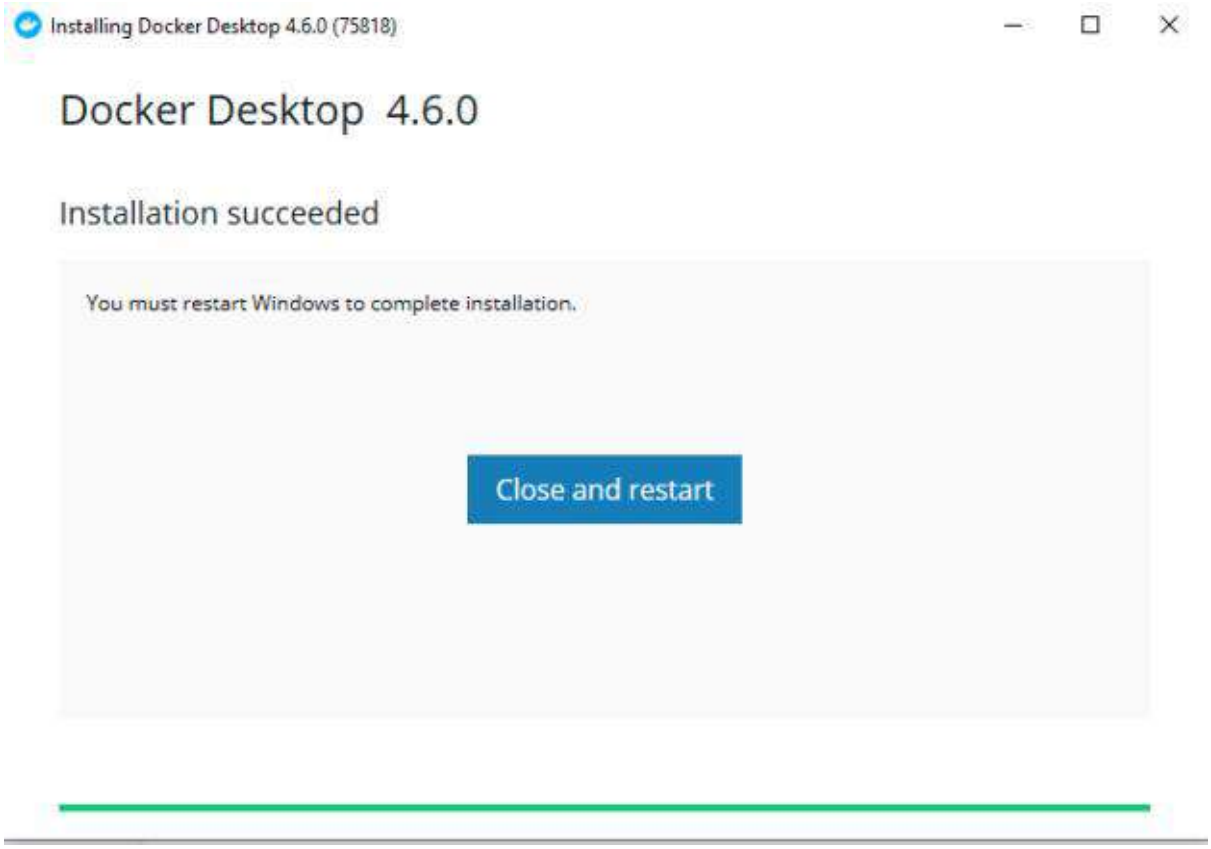


Step 03: After downloading, Open 'Docker Desktop Installer' & start installation.

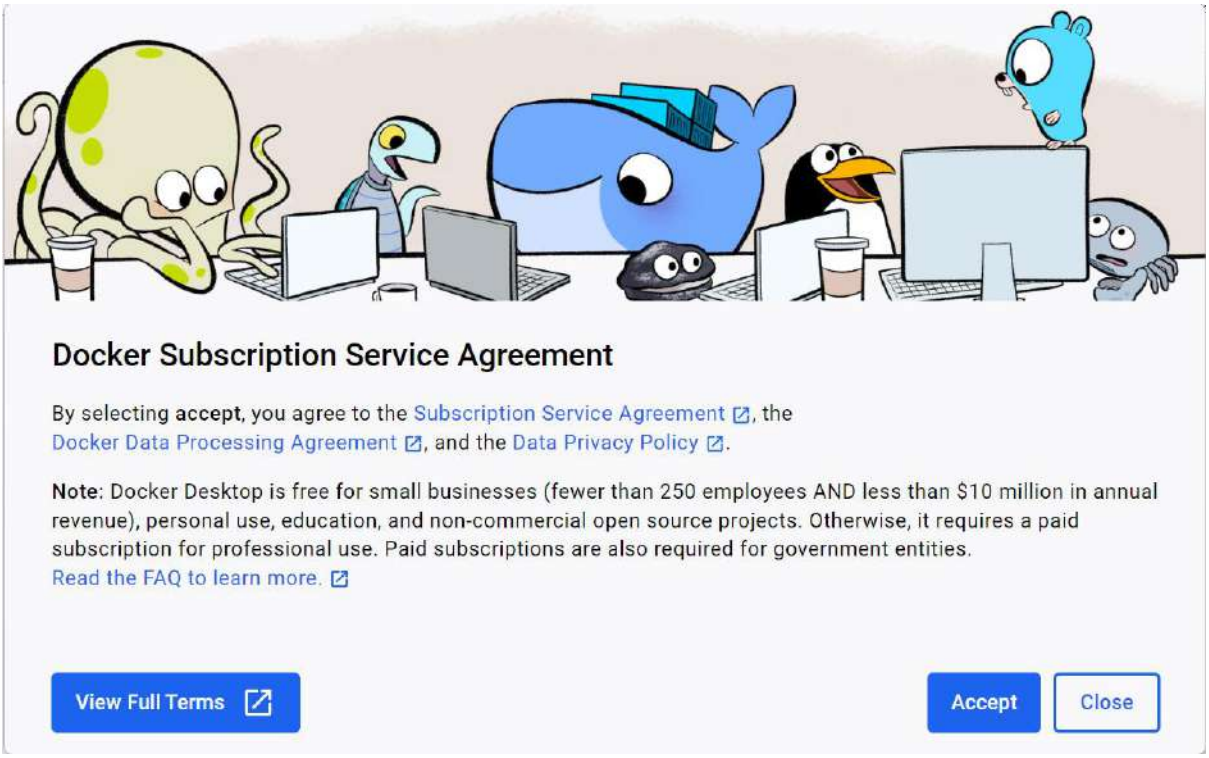




Step 04: After Installation, Restart your device.



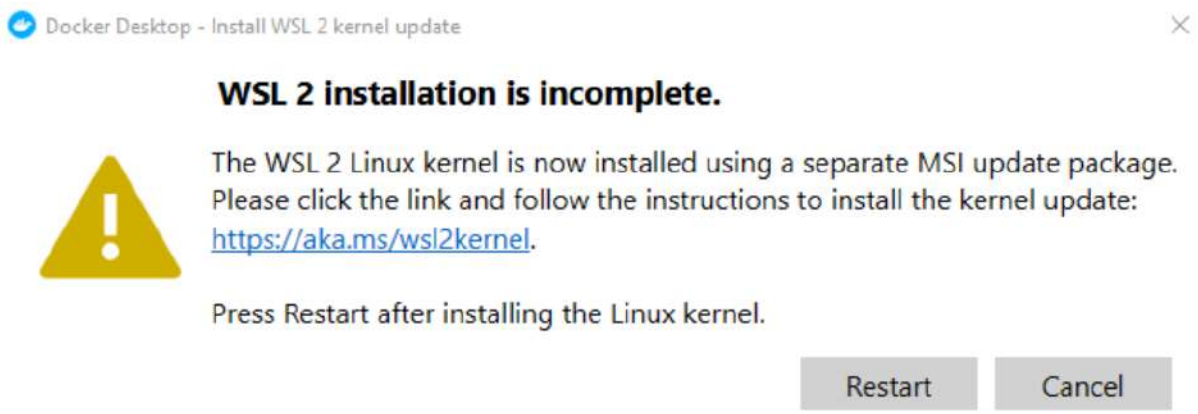
Step 05: Accept the terms and conditions, Click on Accept.



The following window should pop up.

Click on the link - https://aka.ms/wsl2kernel.

(Do not close this window).



Download the WSL2 Linux kernel update package for x64 machines.

Step 4 - Download the Linux kernel update package

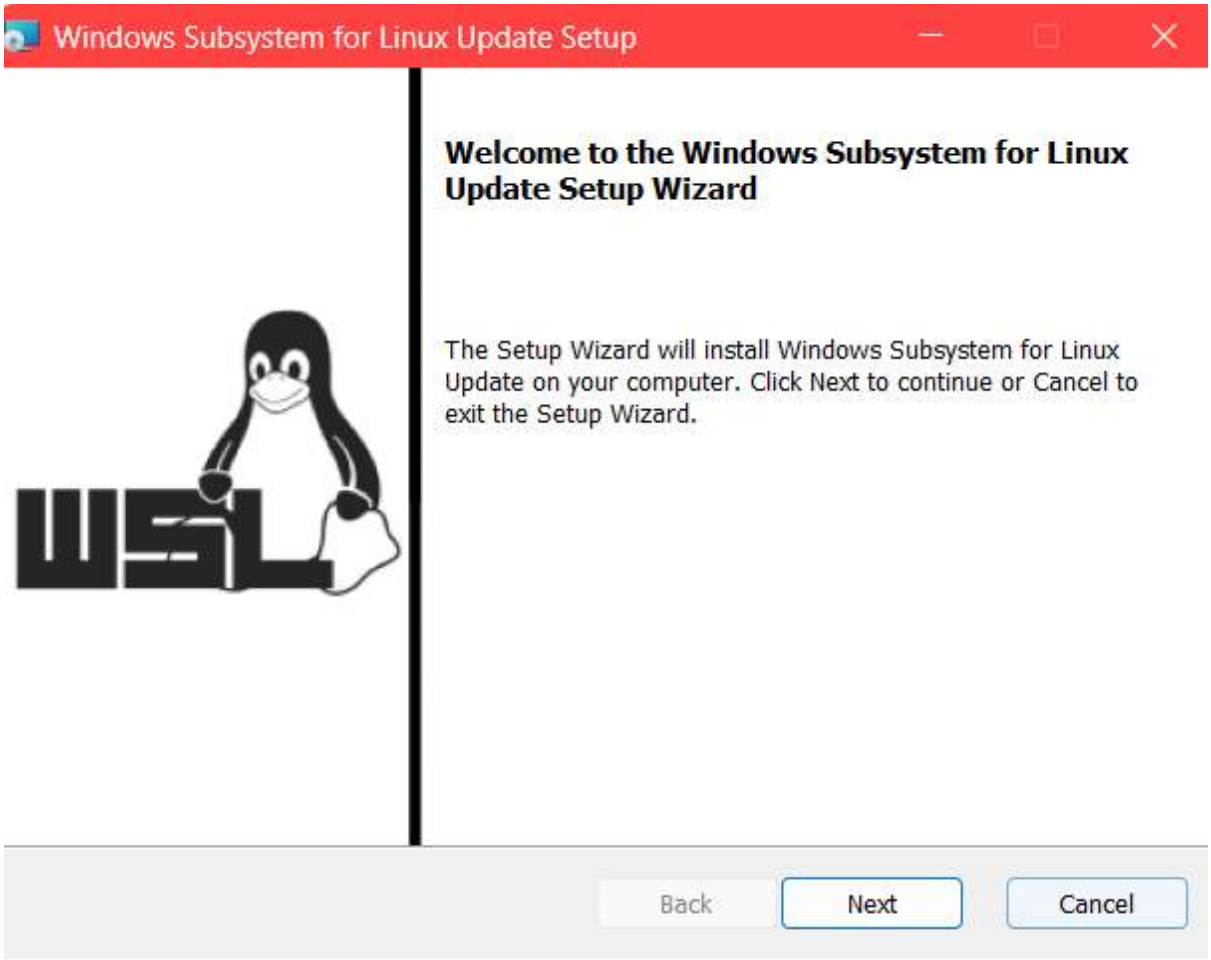
The Linux kernel update package installs the most recent version of the WSL 2 Linux kernel for running WSL inside the Windows operating system image. (To run WSL from the Microsoft Store, with more frequently pushed updates, use `wsl.exe --install` or `wsl.exe --update`.).

1. Download the latest package:
   - WSL2 Linux kernel update package for x64 machines

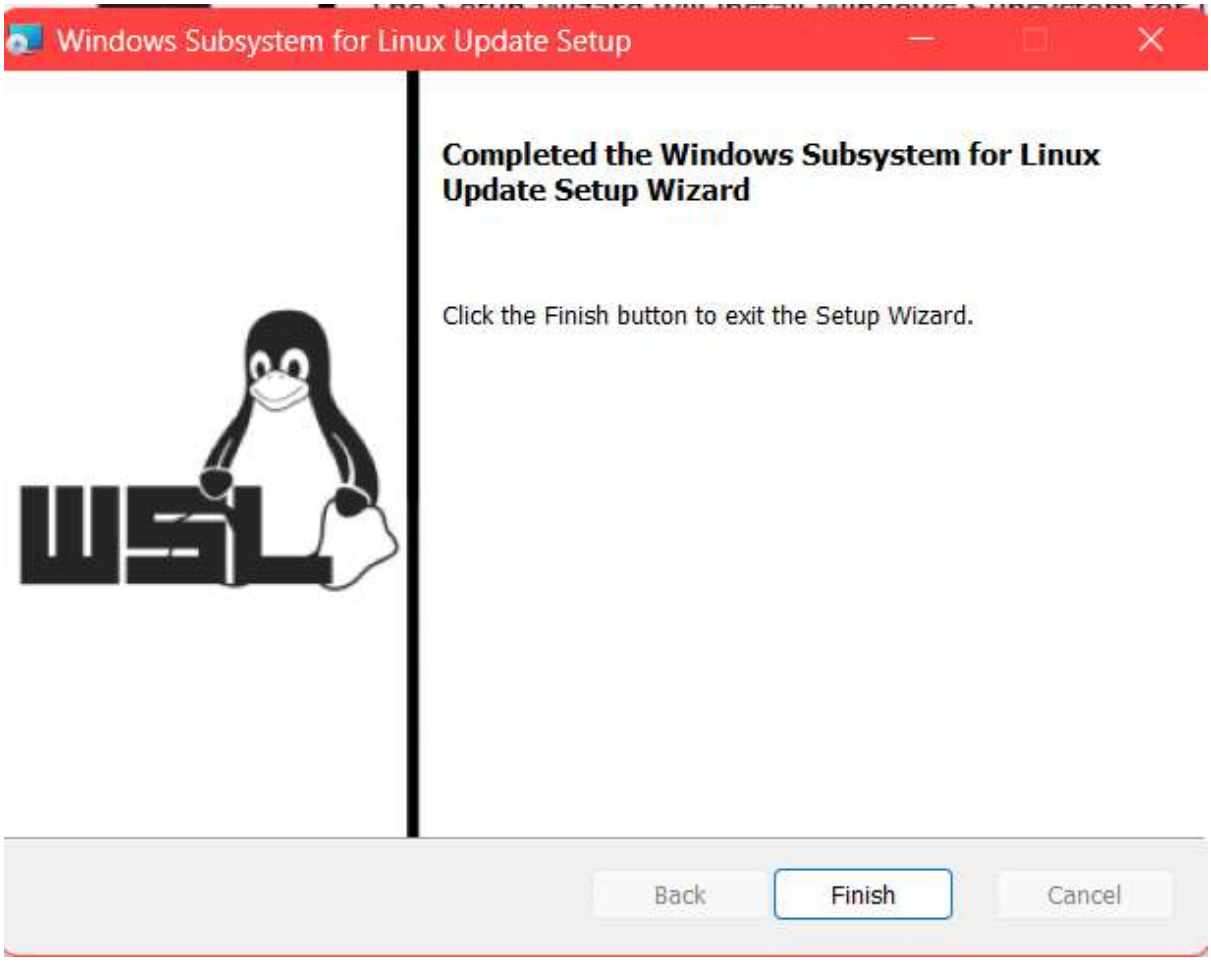
> **Note**
>
> If you're using an ARM64 machine, please download the ARM64 package instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: `systeminfo | find "System Type"`. **Caveat:** On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German `systeminfo | find '"Systemtyp"'`.

2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)

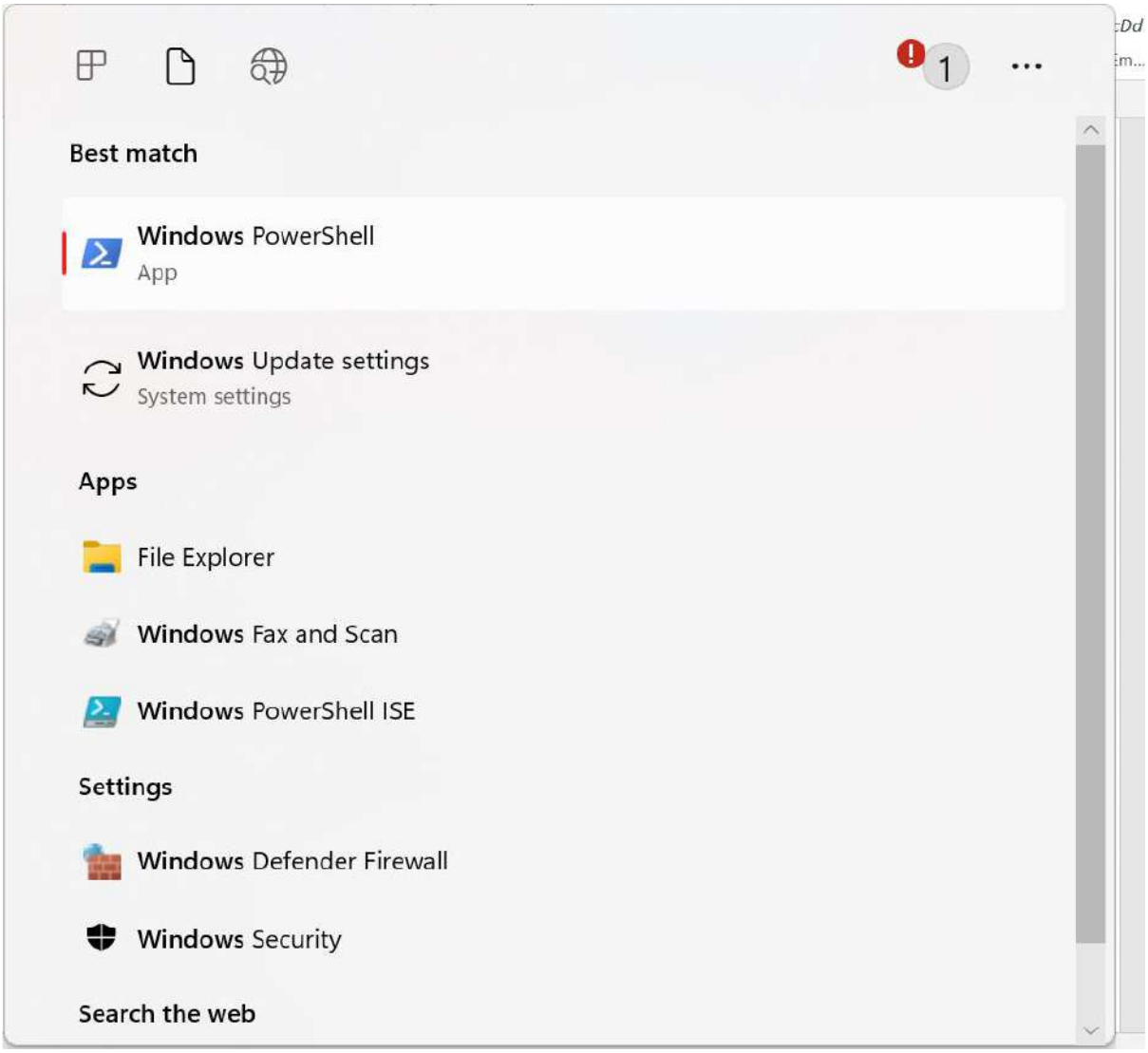
After Download is complete, Run the .msi package. Click on next.



After, the setup is complete, Click on finish.

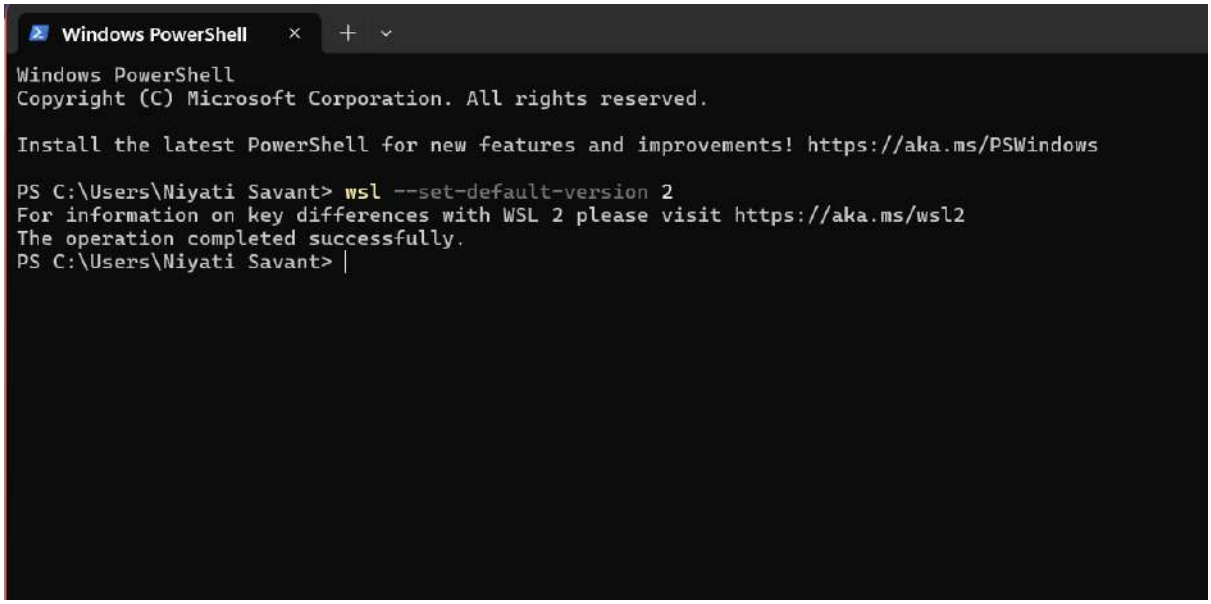


Open Powershell as an Administrator.



Run the following Command:

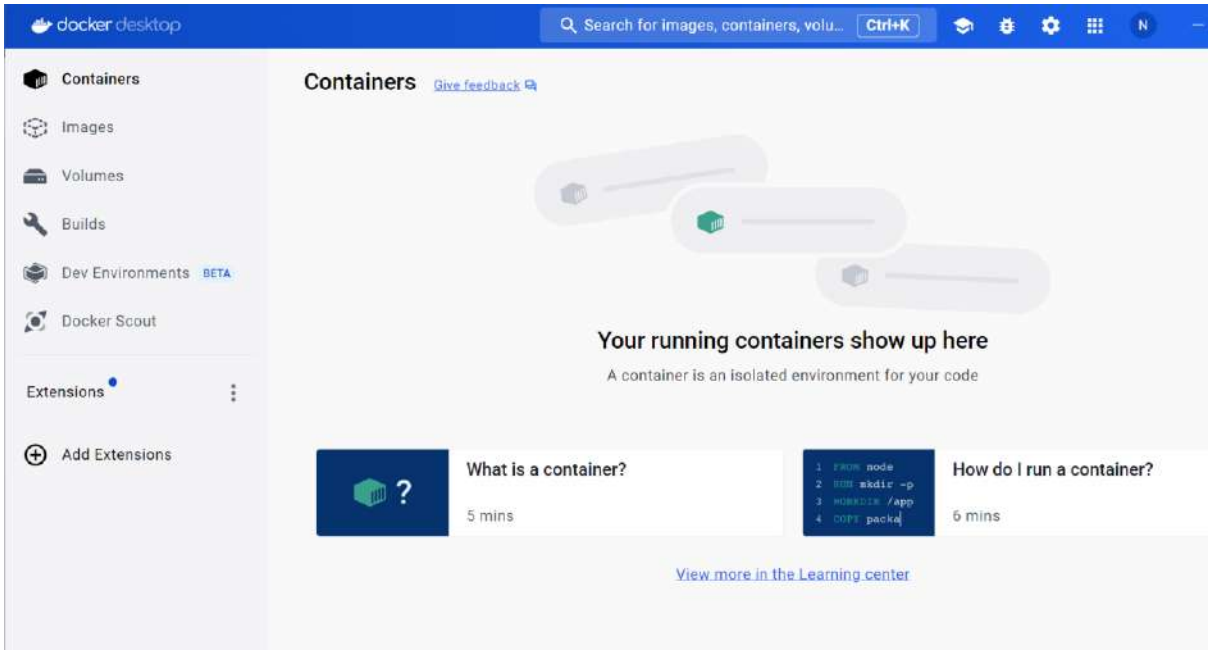
wsl --set-default-version 2



Now, Click on Restart. Docker should now restart.

Click on Start. The following window should pop up.

This means the installation is now complete.



Open Command Prompt, run the following commands:

1) To check the version of Docker:

docker --version

2) To install image of ubuntu

docker pull ubuntu

```
C:\Users\Niyati Savant>docker --version
Docker version 25.0.3, build 4debf41

C:\Users\Niyati Savant>docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
bccd10f490ab: Pull complete
Digest: sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
```

3) Check downloaded images,

docker images

```
C:\Users\Niyati Savant>docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    ca2b0f26964c   6 weeks ago   77.9MB
```

4) Run ubuntu OS

docker run -it ubuntu /bin/bash

```
root@878cb17dc19d:/#
```

5) Open another Command Prompt and follow the steps shown below.

-docker ps

```
PS C:\Users\Niyati Savant> docker ps
CONTAINER ID   IMAGE     COMMAND       CREATED              STATUS              PO
RTS     NAMES
878cb17dc19d   ubuntu    "/bin/bash"   About a minute ago   Up About a minute
        boring_allen
```

-docker container ls –a

```
PS C:\Users\Niyati Savant> docker container ls -a
CONTAINER ID   IMAGE     COMMAND       CREATED         STATUS         PORTS     NA
MES
878cb17dc19d   ubuntu    "/bin/bash"   3 minutes ago   Up 3 minutes
    boring_allen
```

-docker container rm b71e3e6b1118 //copy docker id for remove but first

(Use your container ID in the above command)

```
PS C:\Users\Niyati Savant> docker container rm 878cb17dc19d
Error response from daemon: cannot remove container "/boring_allen": container is
running: stop the container before removing or force remove
```

stop your docker

- docker container stop b71e3e6b1118

```
PS C:\Users\Niyati Savant> docker container stop 878cb17dc19d
878cb17dc19d
```

- docker container rm b71e3e6b1118

```
PS C:\Users\Niyati Savant> docker container rm 878cb17dc19d
878cb17dc19d
```

- docker ps

```
PS C:\Users\Niyati Savant> docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

- docker //list all docker commands

```
PS C:\Users\Niyati Savant> docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run         Create and run a new container from an image
  exec        Execute a command in a running container
  ps          List containers
  build       Build an image from a Dockerfile
  pull        Download an image from a registry
  push        Upload an image to a registry
  images      List images
  login       Log in to a registry
  logout      Log out from a registry
  search      Search Docker Hub for images
  version     Show the Docker version information
  info        Display system-wide information

Management Commands:
  builder     Manage builds
  buildx*     Docker Buildx (Docker Inc., v0.12.1-desktop.4)
  compose*    Docker Compose (Docker Inc., v2.24.6-desktop.1)
  container   Manage containers
  context     Manage contexts
  debug*      Get a shell into any image or container. (Docker Inc., 0.0.24)
  dev*        Docker Dev Environments (Docker Inc., v0.1.0)
  extension*  Manages Docker extensions (Docker Inc., v0.2.22)
  feedback*   Provide feedback, right in your terminal! (Docker Inc., v1.0.4)
  image       Manage images
  init*       Creates Docker-related starter files for your project (Docker Inc.,
v1.0.1)
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an ima
ge (Anchore Inc., 0.6.0)
  scout*      Docker Scout (Docker Inc., v1.5.0)
  system      Manage Docker
```

- docker images

```
PS C:\Users\Niyati Savant> docker images
REPOSITORY   TAG       IMAGE ID       CREATED       SIZE
ubuntu       latest    ca2b0f26964c   6 weeks ago   77.9MB
PS C:\Users\Niyati Savant>
```

- docker image rm ff0fea8310f3 // copy image id from previous output

(Use your image ID in the above command)

```
PS C:\Users\Niyati Savant> docker image rm ca2b0f26964c
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44
eff17e
Deleted: sha256:ca2b0f26964cf2e80ba3e084d5983dab293fdb87485dc6445f3f7bbfc89d7459
Deleted: sha256:5498e8c22f6996f25ef193ee58617d5b37e2a96decf22e72de13c3b34e147591
PS C:\Users\Niyati Savant>
```

- docker run -it ubuntu /bin/bash //check output

```
PS C:\Users\Niyati Savant> docker run -it ubuntu /bin/bash
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
Digest: sha256:77906da86b60585ce12215807090eb327e7386c8fafb5402369e421f44eff17e
Status: Downloaded newer image for ubuntu:latest
```

s

Experiment-10

Aim - To study and implement Container orchestration using Kubernetes.

Theory -

1. Explain the need for a Container orchestration tool:

As organizations adopt containerized application for their software development and deployment needs, they face challenges in managing these containers at scale. Container orchestration tools are essential to automate the deployment, scaling and management of containerized applications across a cluster of hosts. These tools help in ensuring high availabelety, efficient resource utilization, load balancing, and seamless scaling of containerized applications, which are crucial for modern cloud-native architectures.

2. What is Kubernets? Describe its features:

Kubernetes is an open-source container orchestration platform originally developed by Google and now maintained by the Cloud Native Computing Foundation (CNCF). Its features include:

1) Automated container deployment and scaling

Kubernetes allows users to easily deploy containerized applicating and scale

2) Service discovery and load balancing

Kubernetes provides built-in mechanisms for service discovery and load balancing, enabling seamless communication

3) Self-healing
Automatically restarts, replaces, kills container and doesn't advertise them to clients until they are ready to serve

4) Storage orchestration
Kubernetes enables the automatic mounting of storage systems of choice, whether from local storage, public cloud providers.

5) Configuration management
It allows for flexible and dynamic configuration of containerized applications through Config Maps and Secrets.

3. Explain Kubernetes Components, its working
Kubernetes architecture consists of several key components:

- Master Node - It comprises various components responsible for managing the cluster:
  - API Server - Exposes Kubernetes API which is used by internal components and external clients to interact with the cluster
  - Scheduler - Assigns workloads to nodes based on resource requirements and constraints
  - Controller Manager - Monitors the cluster state and handles cluster-wide tasks such as node management, replication and endpoints.
  - etcd - Distributed key-value store that stores cluster configuration and state information.

- Worker Node - Each worker node hosts the following components:
  - Kubelet - Agent responsible for communicating with API server and managing containers on the node
  - Kube-proxy - Network proxy that reflects Kubernets networking services on each node.
  - Container runtime - software responsible for running containers like Docker or Containerd.

Kubernets works by receiving commands via its API server, which interacts with the etcd database to store desired state of cluster. The scheduler then assigns workloads to appropriate nodes and kubelet on each node ensures desired state is maintained.

4. Difference between POD and node

A pod is the smallest deployable unit in Kubernetes, representing one or more containers that are tightly coupled and share resources (like networking and storage). Containers within a pod share same IP address and port space and can communicate with localhost. They are scheduled onto nodes in the cluster, & each pod is unique and isolated.

A node is a worker machine responsible for running containers and providing runtime environment. Each node has necessary services to run pods, including container runtime, Kubelet, kube-proxy. Nodes are where actual workloads are executed, and they form underlying infrastructure of cluster.

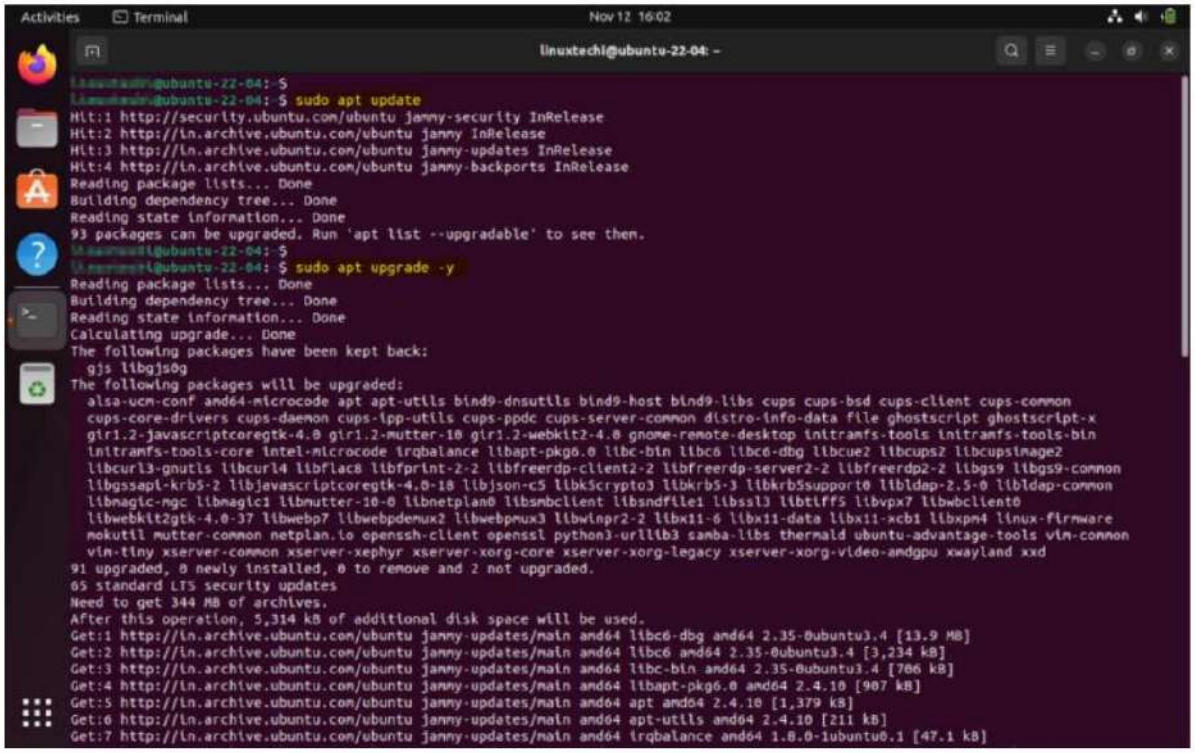
5. Compare Kubernetes and Docker swarm

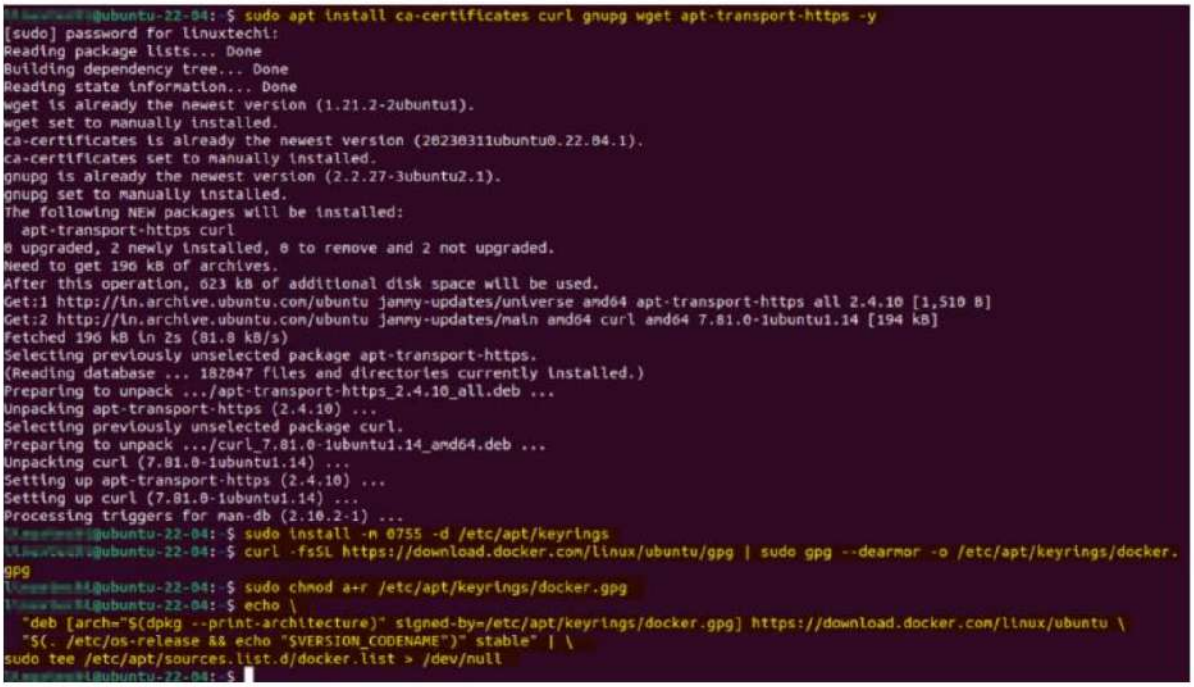
Parameter	Kubernetes	Docker Swarm
Orchestration	Mature and feature-rich	simple to set and use - lack flexibility
Scalability	Ability to manage large clusters with thousands of nodes and containers	May face limitations in extremely large deployments
Community	Larger, more active extensive documentation resources, integration	Smaller community and ecosystem in comparison
Flexibility	More flexibility and customization options	Suitable for smaller, straightforward deployments
Tooling	Rich set of tooling support like monitoring, logging, CI/CD integration	Fewer tools and integrations available
Ease of Use	Steeper learning curve but more robust capabilities	Easier to set up and use.

Sher 28/3/24 (A)

EXPERIMENT 10

To study and implement container orchestration using Kubernetes.





```
[redacted]@ubuntu-22-04:~$ sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  docker-ce-rootless-extras git git-man liberror-perl libslirp0 pigz slirp4netns
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs
  git-mediawiki git-svn
The following NEW packages will be installed:
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git git-man
  liberror-perl libslirp0 pigz slirp4netns
0 upgraded, 12 newly installed, 0 to remove and 2 not upgraded.
Need to get 118 MB of archives.
After this operation, 430 MB of additional disk space will be used.
Get:1 https://download.docker.com/linux/ubuntu jammy/stable amd64 containerd.io amd64 1.6.24-1 [28.6 MB]
Get:2 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git-man all 1:2.34.1-1ubuntu1.10 [954 kB]
Get:5 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-buildx-plugin amd64 0.11.2-1~ubuntu.22.04~jammy [28.2 MB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 git amd64 1:2.34.1-1ubuntu1.10 [3,166 kB]
Get:7 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-cli amd64 5:24.0.7-1~ubuntu.22.04~jammy [13.3 MB]
Get:8 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce amd64 5:24.0.7-1~ubuntu.22.04~jammy [22.6 MB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 libslirp0 amd64 4.6.1-1build1 [61.5 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy/universe amd64 slirp4netns amd64 1.0.1-2 [28.2 kB]
Get:11 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-ce-rootless-extras amd64 5:24.0.7-1~ubuntu.22.04~jammy [9,
030 kB]
Get:12 https://download.docker.com/linux/ubuntu jammy/stable amd64 docker-compose-plugin amd64 2.21.0-1~ubuntu.22.04~jammy [11.9 MB]
Fetched 118 MB in 8s (14.4 MB/s)
```

```
[redacted]@ubuntu-22-04: $ sudo usermod -aG docker $USER
[redacted]@ubuntu-22-04: $ newgrp docker
[redacted]@ubuntu-22-04: $ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
     Active: active (running) since Sun 2023-11-12 16:20:53 IST; 1min 54s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 3114 (dockerd)
      Tasks: 8
     Memory: 27.3M
        CPU: 1.865s
     CGroup: /system.slice/docker.service
             └─3114 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Nov 12 16:20:51 ubuntu-22-04 systemd[1]: Starting Docker Application Container Engine...
Nov 12 16:20:51 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:51.771454830+05:30" level=info msg="Starting up"
Nov 12 16:20:51 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:51.774072108+05:30" level=info msg="detected 127.0.0.53 nameserv
Nov 12 16:20:52 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:52.051794664+05:30" level=info msg="Loading containers: start."
Nov 12 16:20:53 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:53.028201474+05:30" level=info msg="Loading containers: done."
Nov 12 16:20:53 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:53.231111803+05:30" level=info msg="Docker daemon" commit=311b9f
Nov 12 16:20:53 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:53.233235025+05:30" level=info msg="Daemon has completed initial
Nov 12 16:20:53 ubuntu-22-04 dockerd[3114]: time="2023-11-12T16:20:53.351693568+05:30" level=info msg="API listen on /run/docker.so
Nov 12 16:20:53 ubuntu-22-04 systemd[1]: Started Docker Application Container Engine.
```

```
[redacted]@ubuntu-22-04:~$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 89.3M  100 89.3M    0     0   9.8M      0  0:00:09  0:00:09 --:--:-- 13.4M
[redacted]@ubuntu-22-04:~$
[redacted]@ubuntu-22-04:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
[redacted]@ubuntu-22-04:~$
[redacted]@ubuntu-22-04:~$ minikube version
minikube version: v1.32.0
commit: 8220a6eb95f0a4d75f7f2d7b14cef975f050512d
[redacted]@ubuntu-22-04:~$
[redacted]@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ curl -LO https://storage.googleapis.com/kubernetes-release/release/`curl -s https://storage.googleapis.com
/kubernetes-release/release/stable.txt`/bin/linux/amd64/kubectl
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 47.5M  100 47.5M    0     0  8524k      0  0:00:05  0:00:05 --:--:--  9.0M
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ chmod +x kubectl
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ sudo mv kubectl /usr/local/bin/
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ kubectl version -o yaml
clientVersion:
  buildDate: "2023-10-18T11:42:52Z"
  compiler: gc
  gitCommit: a8a1abc25cad87333840cd7d54be2efaf31a3177
  gitTreeState: clean
  gitVersion: v1.28.3
  goVersion: go1.20.10
  major: "1"
  minor: "28"
  platform: linux/amd64
kustomizeVersion: v5.0.4-0.20230601165947-6ce0bf390ce3

The connection to the server localhost:8080 was refused - did you specify the right host or port?
linuxbuzz@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ minikube start --driver=docker
😄  minikube v1.32.0 on Ubuntu 22.04 (vbox/amd64)
✨  Using the docker driver based on user configuration
📌  Using Docker driver with root privileges
👍  Starting control plane node minikube in cluster minikube
🚜  Pulling base image ...
💾  Downloading Kubernetes v1.28.3 preload ...
    > preloaded-images-k8s-v18-v1...:  403.35 MiB / 403.35 MiB  100.00% 7.23 Mi
    > gcr.io/k8s-minikube/kicbase...:  453.90 MiB / 453.90 MiB  100.00% 7.36 Mi
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.28.3 on Docker 24.0.7 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
linuxbuzz@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ minikube status
minikube
type: Control Plane
host: Running
kubelet: Running
apiserver: Running
kubeconfig: Configured

linuxbuzz@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
minikube   Ready    control-plane   5m49s   v1.28.3
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ kubectl create deployment nginx-web --image=nginx
deployment.apps/nginx-web created
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ kubectl expose deployment nginx-web --type NodePort --port=80
service/nginx-web exposed
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ kubectl get deployment,pod,svc
NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/nginx-web         1/1     1            1           40s

NAME                                  READY   STATUS    RESTARTS   AGE
pod/nginx-web-5b757f798d-qnbzq        1/1     Running   0          39s

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP        12m
service/nginx-web    NodePort    10.104.243.119   <none>        80:30523/TCP   27s
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$
```

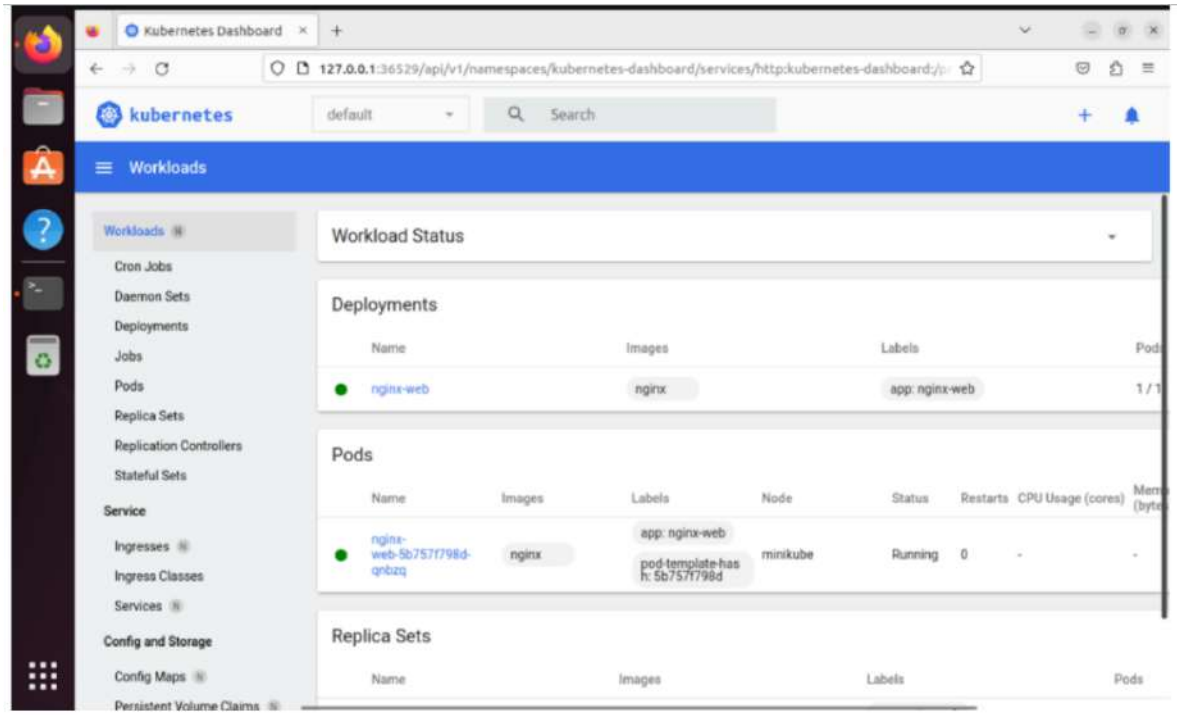
```
linuxbuzz@ubuntu-22-04:~$ minikube addons list
|-----------------------------|----------|-------------|--------------------------------|
|         ADDON NAME          | PROFILE  |   STATUS    |           MAINTAINER           |
|-----------------------------|----------|-------------|--------------------------------|
| ambassador                  | minikube | disabled    | 3rd party (Ambassador)         |
| auto-pause                  | minikube | disabled    | minikube                       |
| cloud-spanner               | minikube | disabled    | Google                         |
| csi-hostpath-driver         | minikube | disabled    | Kubernetes                     |
| dashboard                   | minikube | disabled    | Kubernetes                     |
| default-storageclass        | minikube | enabled ✅  | Kubernetes                     |
| efk                         | minikube | disabled    | 3rd party (Elastic)            |
| freshpod                    | minikube | disabled    | Google                         |
| gcp-auth                    | minikube | disabled    | Google                         |
| gvisor                      | minikube | disabled    | minikube                       |
| headlamp                    | minikube | disabled    | 3rd party (kinvolk.io)         |
| helm-tiller                 | minikube | disabled    | 3rd party (Helm)               |
| inaccel                     | minikube | disabled    | 3rd party (InAccel             |
|                             |          |             | [info@inaccel.com])            |
| ingress                     | minikube | disabled    | Kubernetes                     |
| ingress-dns                 | minikube | disabled    | minikube                       |
| inspektor-gadget            | minikube | disabled    | 3rd party                      |
|                             |          |             | (inspektor-gadget.io)          |
| istio                       | minikube | disabled    | 3rd party (Istio)              |
| istio-provisioner           | minikube | disabled    | 3rd party (Istio)              |
| kong                        | minikube | disabled    | 3rd party (Kong HQ)            |
| kubeflow                    | minikube | disabled    | 3rd party                      |
| kubevirt                    | minikube | disabled    | 3rd party (KubeVirt)           |
| logviewer                   | minikube | disabled    | 3rd party (unknown)            |
| metallb                     | minikube | disabled    | 3rd party (MetalLB)            |
| metrics-server              | minikube | disabled    | Kubernetes                     |
| nvidia-device-plugin        | minikube | disabled    | 3rd party (NVIDIA)             |
| nvidia-driver-installer     | minikube | disabled    | 3rd party (Nvidia)             |
| nvidia-gpu-device-plugin    | minikube | disabled    | 3rd party (Nvidia)             |
| olm                         | minikube | disabled    | 3rd party (Operator Framework) |
| pod-security-policy         | minikube | disabled    | 3rd party (unknown)            |
| portainer                   | minikube | disabled    | 3rd party (Portainer.io)       |
| registry                    | minikube | disabled    | minikube                       |
| registry-aliases            | minikube | disabled    | 3rd party (unknown)            |
```

```
linuxbuzz@ubuntu-22-04:~$ minikube addons enable dashboard
💡  dashboard is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image docker.io/kubernetesui/dashboard:v2.7.0
    ▪ Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
💡  Some dashboard features require the metrics-server addon. To enable all features please run:

        minikube addons enable metrics-server


🌟  The 'dashboard' addon is enabled
linuxbuzz@ubuntu-22-04:~$
linuxbuzz@ubuntu-22-04:~$ minikube addons enable ingress
💡  ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
    ▪ Using image registry.k8s.io/ingress-nginx/controller:v1.9.4
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
    ▪ Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v20231011-8b53cabe0
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
linuxbuzz@ubuntu-22-04:~$
```

```
linuxbuzz@ubuntu-22-04:~$ minikube dashboard
🤔  Verifying dashboard health ...
🚀  Launching proxy ...
🤔  Verifying proxy health ...
🎉  Opening http://127.0.0.1:36529/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/ in your default
 browser...
```



Niyati Savant
C31-TE
156

Assignment - 1

THADOMAL SHAHANI
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q1 What is Fog Computing? Explain need of Fog Computing

→ The term 'fog computing' was coined by Cisco in January 2014. This was because fog is referred to as clouds that are close to ground in the same way, fog computing is related to nodes which are present near nodes somewhere in between host and cloud. It was intended to bring computational capabilities of system close to host machine.

It is a form of distributed computing that brings computation and data storage closer to network edge where many IOT devices are located. By doing this, fog computing reduces the reliance on cloud for these resource-intensive tasks, improving performance and reducing latency.

The following are some reasons why fog computing is used-

1) To improve latency and performance
Fog nodes are deployed at network edge, closer to IOT devices

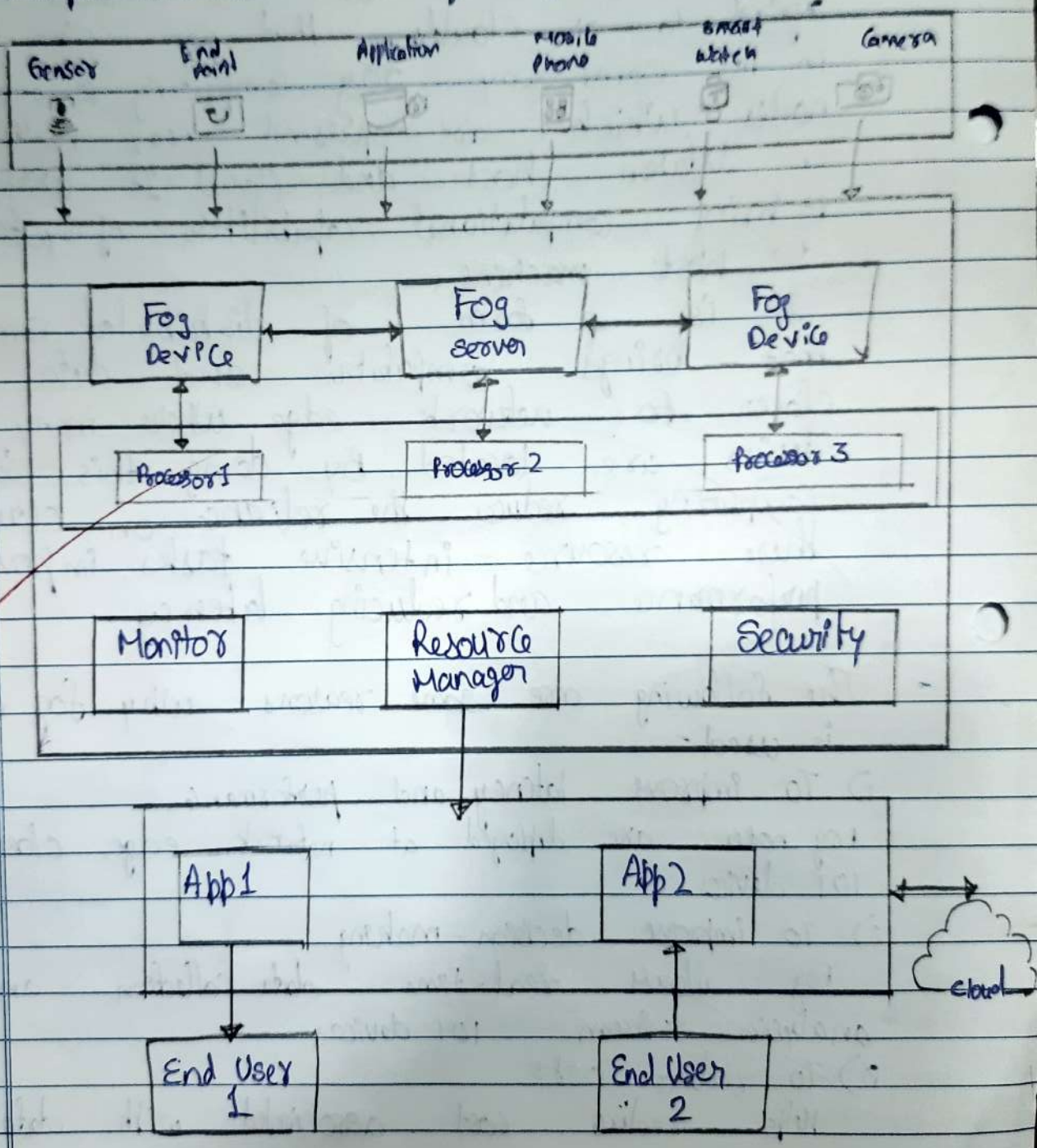
2) To improve decision-making
Fog allows real-time data collection and analysis from IOT devices

3) To reduce costs
Helps reduce cost associated with data

Niyati
Niyati
TE - C31
156

storage and analysis by reducing the amount of data that needs to be transmitted back to a central location for processing.

Q2] Explain architecture of Fog Computing



Niyati
CSE - TE
156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

The common components across the architectures -

1) Physical and virtual nodes (end devices)
End devices are contact points to real world and include mobiles, smartwatches or sensors. These are data generators and can span large spectrum of technology.

2) Fog nodes
Independent devices that pick up the generated information. Fog node can be fog device, server or gateway. These store necessary data while servers compute this data to decide actions. The devices are linked to fog servers while gateways redirect the information

3) Monitoring services
The services include APIs that track system's performance and resource availability. They ensure continuous communication exist between fog nodes and all end devices.

4) Data processors
Programs that run on fog nodes that filter, trim, and even reconstruct faulty data. They decide what to do with data - to store it locally on servers or send to cloud for long term storage.

5) Resource Manager
Fog computing consists of independent nodes that must work in synchronized manner. Manager allocates and deallocates resources to nodes and schedules data transfer.

Niyati
TE - C31
156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q3 Applications of Fog Computing

→ · Can be used to monitor and analyze patient's condition and even alert doctors in case of emergency.
· Used for real-time rail monitoring as for high-speed trains
· Gas and oils pipeline optimization.
· In healthcare for remote patient monitoring wearable devices, telemedicine, wearable devices, ambulatory care
· Smart homes where the gadgets are merged into single platform

Q4 Explain Fog, Edge and Cloud Computing:

→ 1. Fog Computing:
It is an extension of edge computing that emphasizes bringing computing resources closer to the data source, typically within the same network infrastructure. It aims to address limitations of traditional cloud by edge computing by providing decentralized infrastructure

2. Edge Computing:
Edge computing refers to the practice of processing data near the edges of the network. This approach reduces latency, bandwidth usage of reliance on cloud infrastructure.

Niyati
TE - C31
156

It is useful for applications that require real-time processing or low-latency response such as IOT devices.

3. Cloud Computing.

Cloud Computing involves the delivery of various computing services - servers, storage, databases, networking as more over the internet. Users can access those services on demand from cloud providers, paying only for the resources they use. It offers scalability flexibility cost-effectiveness making it possible for business to large scale.

Q5. Case study of fog computing

→ The Barcelona smart city project, in collaboration with cisco, exemplifies how fog computing & IOT technology can revolutionize urban infrastructure as ennance citizen well being. By deploying thousands of IOT sensors across the city collected real time data of various aspect of urban life from traffic to our quality

- To process this data efficiently, also implemented a fog computing infrastructure comprising edge devices and fog nodes

Niyati
TE - C31
156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

strategically placed throughout the city. These nodes act as intermediate processing points between the IOT devices and the cloud reducing latency and enabling real-time insights at the edge of the network.

3. The collected data underwent advance analytics to derive actionable insights. forming basis for development of smart applications aimed at addressing specific challenges. Intelligent traffic management systems where implemented to reduce congestion.

4. The project yielded significant benifits for city and its resident. Urban services like transportation, waste management as public safety were improved. Leading to a better quality of life. Additionally, the project contributed to enhances sustainability by optimising resource usage and reducing environment ~~impact~~ impact.

8/4/24 (A)

Niyati Savant
TE - C31
2103156

Assignment 02

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Q. Explore & Compare similar types of services provided by AWS, Azure and Google Cloud Platform.

→ These are 3 major cloud service providers, offering a ~~wide~~ wide range of services while each provider offers similar types of services there are differences in terms of offering features, pricing and integration

1. Compute Services
a) AWS EC2 provides resizable compute capacity in cloud
b) Azure Virtual Machines offer scalable computing power for Windows and Linux workloads
c) Google Compute Engine offer machines of ~~diff~~ flexible configurations and fast SSD storage.

2. Storage Services
a) AWS S3 highly scalable object storage service designed for durability and availability
b) Azure Blob Storage offers scalable, cost-effective storage for unstructured data.
c) Google Cloud Storage provides scalable storage for objects and data, with strong consistency and global edge-caching.

3. Database Services
a) AWS RDS offers managed relational databases like MySQL, PostgreSQL, and SQL Server

Niyati

Niyati
TEC 31
2103156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

b) Azure SQL Database is fully managed relational database service with built-in intelligence
c) Google Cloud SQL provides managed MySQL, PostgreSQL, and SQL Server databases.

4. Networking services
a) AWS VPC enables you to provision a logically isolated section of AWS Cloud.
b) Azure Virtual Network allows you to create private networks and connect to on-premises infrastructure.
c) Google VPC lets you create and ~~mage~~ manage virtual private clouds for your Google Cloud resources

5. Containers services
a) AWS ECS: fully managed container orchestration service for Docker containers.
b) Azure AKS simplifies deployment, management, and scaling of containerized applications using Kubernetes
c) Google GKE offers managed Kubernetes clusters for containerized applications

6. Serverless computing:
a) AWS lambda allows you to run code without provisioning or managing servers.
b) Azure Functions enables event-driven, serverless computing with seamless integration
c) Google Cloud Functions allows you to run event-driven functions in serverless environment

Niyati
C31
156

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

7. AI/ML Services:
a) AWS Sage Maker provide tools to build, train and deploy ML models at scale.
b) Azure ML offers comprehensive ML service with SDKs and visual tools
c) Google Cloud AI provide end-to-end AIML along with data labeling, training and deployment

8. Big Data & Analytics
a) AWS EMR is managed big data platform for processing large scale data using Apache Hadoop and Spark.
b) Azure HDInsight offer managed Apache Hadoop, Spark and Hbase clusters for big data
c) Google Cloud Dataproc provide managed Apache Spark for data processing

9. DevOps Tools:
a) AWS CodePipeline is a continuous integration and continuous delivery service
b) Azure DevOps offers suit of tools for planning, development, testing, deployment.
c) Google Cloud Build automates build, test & deployment processes.

10. Identity and Access Management (IAM)
a) AWS IAM enables you to manage access to AWS services and resources securely

Niyati
TE - C31
156

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

b) Azure Active Directory provides identity and access management services for Azure & Microsoft 365.
c) Google Cloud IAM offers centralized access management for resources and services

8/4/24

A

CCL MINI-PROJECT

Title: Real-time Application using Cyclic.sh, Node.js, and MongoDB for understanding Cloud Infrastructure (Storage as a Service)

Shruti Punjabi 2103141
Parth Puranik 2103142
Niyati Savant 2103156
Tanishqa Sawant 2103157

Description of problem statement

Develop a real-time application to demonstrate the utilisation of Cloud Infrastructure, focusing on Storage as a Service, using Cyclic.sh, Node.js, and MongoDB. The application should showcase the integration of these technologies to provide scalable and efficient storage solutions in a cloud environment.

Key Objectives:

- Real-Time Application Development: Build a functional real-time application using Node.js that interacts with MongoDB for data storage and retrieval.

- Utilisation of Cyclic.sh: Integrate Cyclic.sh for managing cloud infrastructure, specifically focusing on storage resources. Utilize its features for scalability, monitoring, and efficient resource allocation.

- Cloud-Based Storage Solution: Implement MongoDB as the database solution, emphasizing its role as a cloud-based storage service. Utilize MongoDB's features for document storage and retrieval.

- Scalability and Performance: Design the application and infrastructure to be scalable, ensuring efficient handling of increasing workloads and concurrent users.

- Real-Time Interaction: Implement real-time features in the application using technologies like WebSockets or server-sent events to demonstrate instant updates and communication between clients and the server.

- Monitoring and Logging: Implement monitoring and logging mechanisms within the application and infrastructure using tools provided by Cyclic.sh and Node.js. Monitor resource usage, application performance, and system health.

- Documentation and Presentation: Document the entire development process, including setup instructions, architecture overview, code documentation, and any challenges faced during implementation. Prepare a presentation to demonstrate the application functionality and the utilization of cloud infrastructure

SaaS (Software as a Service): SaaS is a software distribution model where applications are hosted by a third-party provider and made available to customers over the internet. Instead of installing and maintaining software locally, users can access the application via a web browser. SaaS eliminates the need for organisations to handle the installation, maintenance,

and upgrades of software, as these responsibilities are managed by the SaaS provider. Examples of SaaS include Google Workspace, Salesforce, and Microsoft 365.

Cyclic.sh: Cyclic.sh is a cloud infrastructure management platform that provides tools and services for deploying, managing, and scaling applications in cloud environments. It offers features such as automated scaling, monitoring, and logging to help optimize resource usage and ensure application reliability. Cyclic.sh supports various cloud providers like AWS, Google Cloud Platform, and Microsoft Azure, allowing users to deploy and manage their applications across different cloud environments efficiently.

Node.js Runtime: Node.js is an open-source, cross-platform JavaScript runtime environment that executes JavaScript code outside of a web browser. It allows developers to build scalable and high-performance applications using JavaScript on the server-side. Node.js is built on Chrome's V8 JavaScript engine and provides a rich set of libraries and frameworks for developing web applications, APIs, and real-time applications. Node.js is commonly used for building backend services, microservices, and server-side applications. It supports event-driven, non-blocking I/O model, making it suitable for building highly responsive and scalable applications.

4.Requirement Specification

For a project aiming to demonstrate the utilization of Cloud Infrastructure (Storage as a Service) using Cyclic.sh, Node.js, and MongoDB, the following are the specific requirements and specifications:

Functional Requirements:

- User Authentication: Implement user authentication mechanisms to ensure secure access to the application.
- Data Storage: Utilize MongoDB as the database for storing and retrieving application data.
- Real-Time Communication: Implement real-time features using WebSockets or server-sent events for instant updates and communication between clients and the server.
- CRUD Operations: Implement CRUD (Create, Read, Update, Delete) operations to manage data stored in MongoDB.
- Scalability: Design the application to be horizontally scalable, allowing it to handle increasing workloads efficiently.
- Monitoring: Implement monitoring mechanisms to track application performance, resource usage, and system health using tools provided by Cyclic.sh.
- Logging: Implement logging mechanisms to capture and analyze application events, errors, and system activities.

Non-Functional Requirements:

- Performance: Ensure the application delivers optimal performance under various workloads, with minimal latency and response times.
- Reliability: Design the application to be reliable, with failover mechanisms in place to handle system failures gracefully.
- Security: Implement robust security measures to protect user data, prevent unauthorized access, and mitigate security threats.
- Scalability: Design the application and infrastructure to be scalable, allowing it to handle increasing user loads without compromising performance.
- Maintainability: Write clean, well-documented code and establish coding standards to facilitate code maintenance and future enhancements.
- Usability: Design an intuitive user interface with clear navigation and responsive design to enhance user experience.

Integration Requirements:

- Integration with Cyclic.sh: Configure and integrate Cyclic.sh for managing cloud infrastructure, including storage resources, monitoring, and scaling.
- Integration with MongoDB: Connect the Node.js application to MongoDB for storing and retrieving data.
- Integration with Real-Time Communication: Integrate real-time communication features using WebSockets or server-sent events for instant updates.
- Deployment Requirements:
- Deployment Automation: Implement automated deployment processes using CI/CD pipelines for deploying application updates seamlessly.
- Cloud Deployment: Deploy the application and infrastructure to a cloud environment supported by Cyclic.sh, such as AWS, Google Cloud Platform, or Microsoft Azure.
- Containerization: Utilize containerization technologies like Docker to package the application and its dependencies for easy deployment and scalability.

Documentation and Training Requirements:

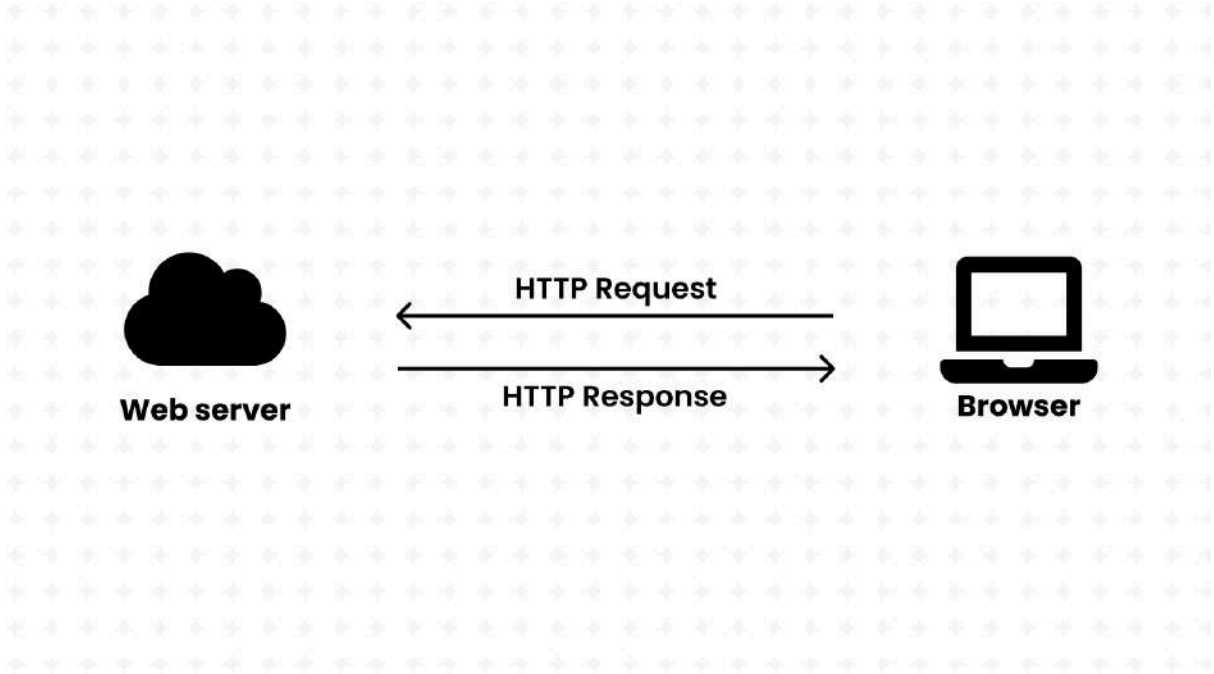
- Setup Instructions: Provide detailed instructions for setting up the development environment, including installing dependencies and configuring Cyclic.sh and MongoDB.
- Architecture Overview: Document the application architecture, including components, data flow, and interaction between different modules.
- Code Documentation: Write comprehensive documentation for the codebase, including code comments, API references, and usage examples.
- Training Materials: Prepare training materials and resources to onboard users and developers to the application and cloud infrastructure management using Cyclic.sh.
- Testing Requirements:
    - Unit Testing: Implement unit tests to verify the functionality of individual components and modules.
    - Integration Testing: Perform integration tests to ensure seamless interaction between different parts of the application and external services.

    - Load Testing: Conduct load tests to evaluate the performance and scalability of the application under heavy workloads.
    - Security Testing: Perform security testing to identify and address potential vulnerabilities and security weaknesses in the application and infrastructure.

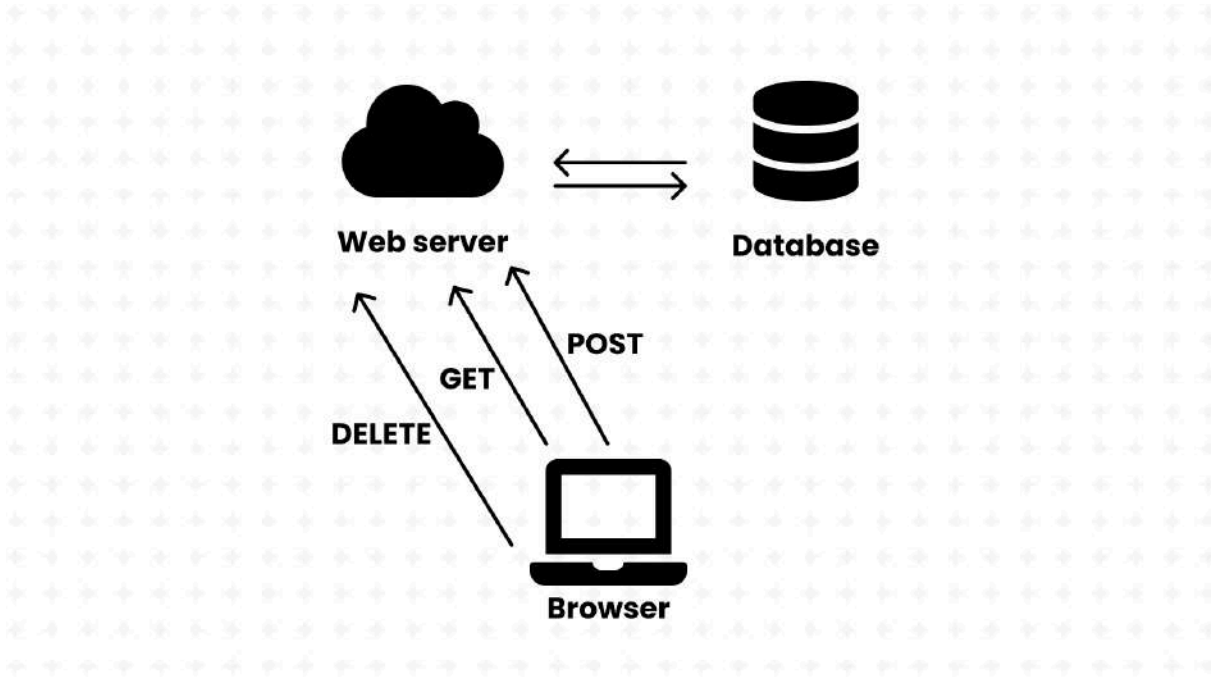
Compliance Requirements:

- Compliance with Data Protection Regulations: Ensure compliance with relevant data protection regulations, such as GDPR, by implementing appropriate data encryption, access controls, and privacy measures.
- Compliance with Cloud Service Provider Policies: Adhere to the policies and guidelines set forth by the chosen cloud service provider (e.g., AWS, GCP, Azure) and Cyclic.sh for deploying and managing applications in their respective environments.

5.Block diagram /Architecture Diagram



The client-server model is a computing architecture where clients request services or resources from servers over a network. Clients initiate communication, servers provide resources, and communication follows a request-response model using standardised protocols. It offers scalability, security, and efficient resource management.



In a client-server architecture, the process of fetching data from a database involves several sequential steps. Initially, the client sends a request to the server, outlining the desired data or operation. Upon receipt, the server interprets the request, potentially verifying permissions

and conducting authentication. Subsequently, the server formulates a query based on the client's request and dispatches it to the database management system (DBMS). The DBMS executes the query, retrieving the pertinent data from the database. Optionally, the server may further process the retrieved data, such as filtering or sorting, before constructing a response. Once prepared, the server transmits the response back to the client. Upon receiving the response, the client processes the data according to its requirements, potentially displaying it to the user or utilizing it for further actions. This coordinated exchange ensures seamless communication and efficient data retrieval between clients and servers in the networked environment.

6.Main code/Major steps

Setup Development Environment:

Install Node.js and npm (Node Package Manager) on your development machine.
Install MongoDB locally or set up a MongoDB instance in the cloud (e.g., MongoDB Atlas).
Set up a Cyclic.sh account and configure it to manage cloud infrastructure.
Initialize Node.js Project:

Create a new directory for your project.
Initialise a new Node.js project using npm init command.
Install necessary dependencies such as Express.js (for building web server), socket.io (for real-time communication), and MongoDB driver for Node.js.
Develop Application Backend:

Create a Node.js server using Express.js framework.
Implement routes for user authentication, data CRUD operations, and real-time communication.
Connect to MongoDB database using the MongoDB driver for Node.js.
Implement authentication middleware to protect routes that require authentication.
Implement Real-Time Communication:

Integrate socket.io library to enable real-time bidirectional communication between clients and the server.
Implement event listeners and emitters for handling real-time messages, updates, and notifications.
Integrate with MongoDB:

Define MongoDB schemas for your application data models.
Implement functions to perform CRUD operations (Create, Read, Update, Delete) on MongoDB collections.
Handle errors and edge cases related to database operations.
Utilise Cyclic.sh for Cloud Infrastructure Management:

Configure Cyclic.sh to provision cloud resources such as virtual machines, storage, and networking components.
Utilise Cyclic.sh features for automated scaling, monitoring, and logging of cloud infrastructure.
Integrate Cyclic.sh APIs or CLI commands into your deployment scripts for managing cloud resources programmatically.
Frontend Development (Optional):

Develop a frontend interface using HTML, CSS, and JavaScript (or a frontend framework like React.js or Angular) to interact with the backend server.
Implement user interfaces for authentication, data management, and real-time updates.
Testing and Debugging:

Write unit tests for backend logic using testing frameworks like Jest or Mocha.
Perform integration tests to ensure smooth interaction between different components.
Debug and fix any issues encountered during development.
Deployment and Continuous Integration:

Containerize your application using Docker for easy deployment and scalability.
Set up CI/CD pipelines using tools like Jenkins, Travis CI, or GitHub Actions for automated testing and deployment.
Deploy the application to cloud infrastructure managed by Cyclic.sh, ensuring proper configuration and resource allocation.
Monitoring and Maintenance:

Monitor application performance, resource usage, and system health using built-in monitoring tools provided by Cyclic.sh.
Set up logging to capture and analyse application logs, errors, and events.
Perform regular maintenance tasks such as database backups, security patches, and software updates.
Documentation and Training:

Document the entire development process, including setup instructions, architecture overview, code documentation, and deployment procedures.
Prepare training materials and resources to onboard users and developers to the application and cloud infrastructure management using Cyclic.sh.

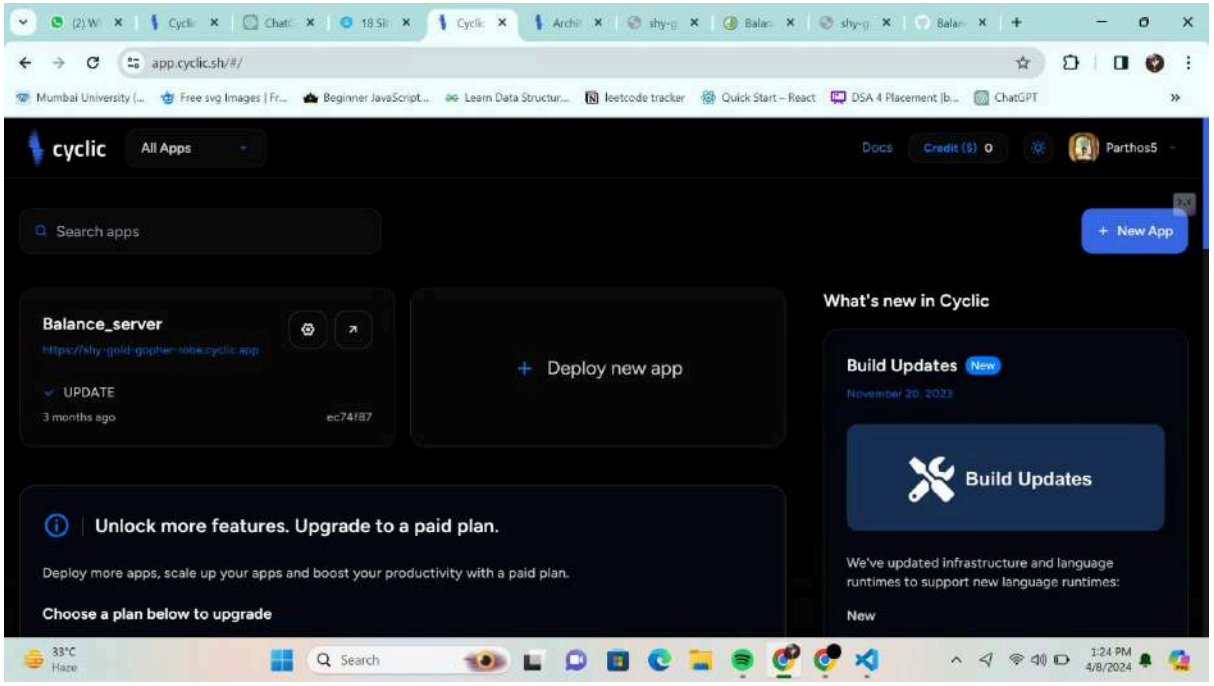
Example of response from Server:

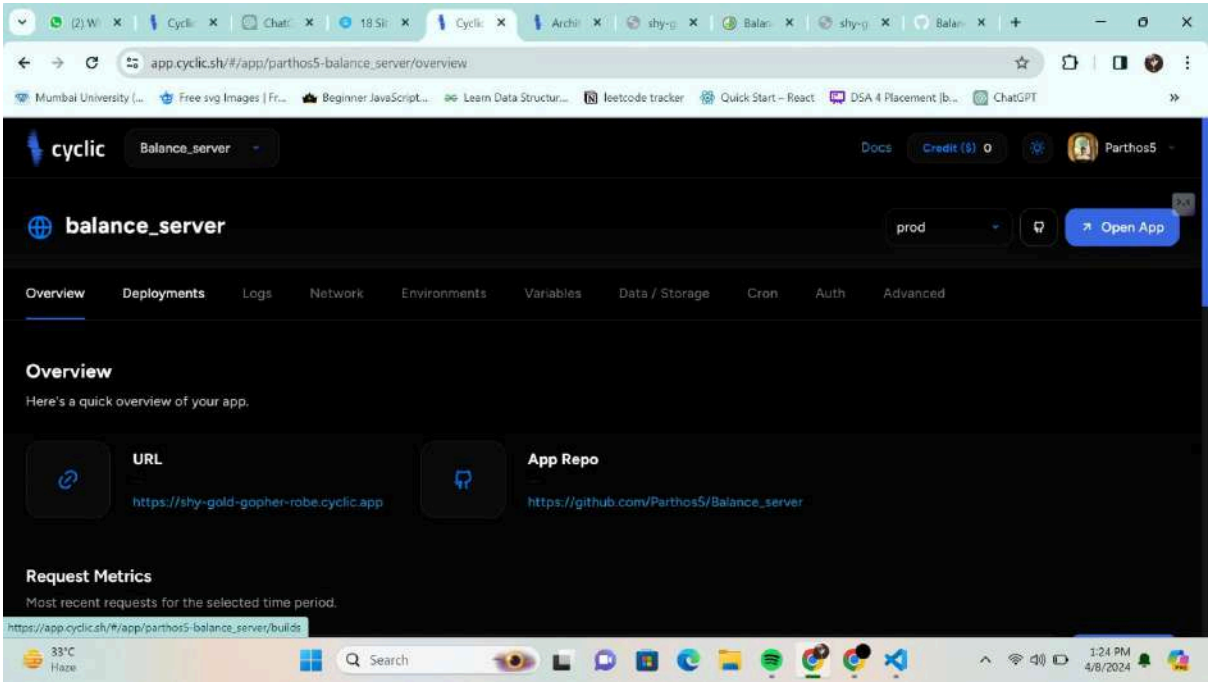
Get Tasks Request:

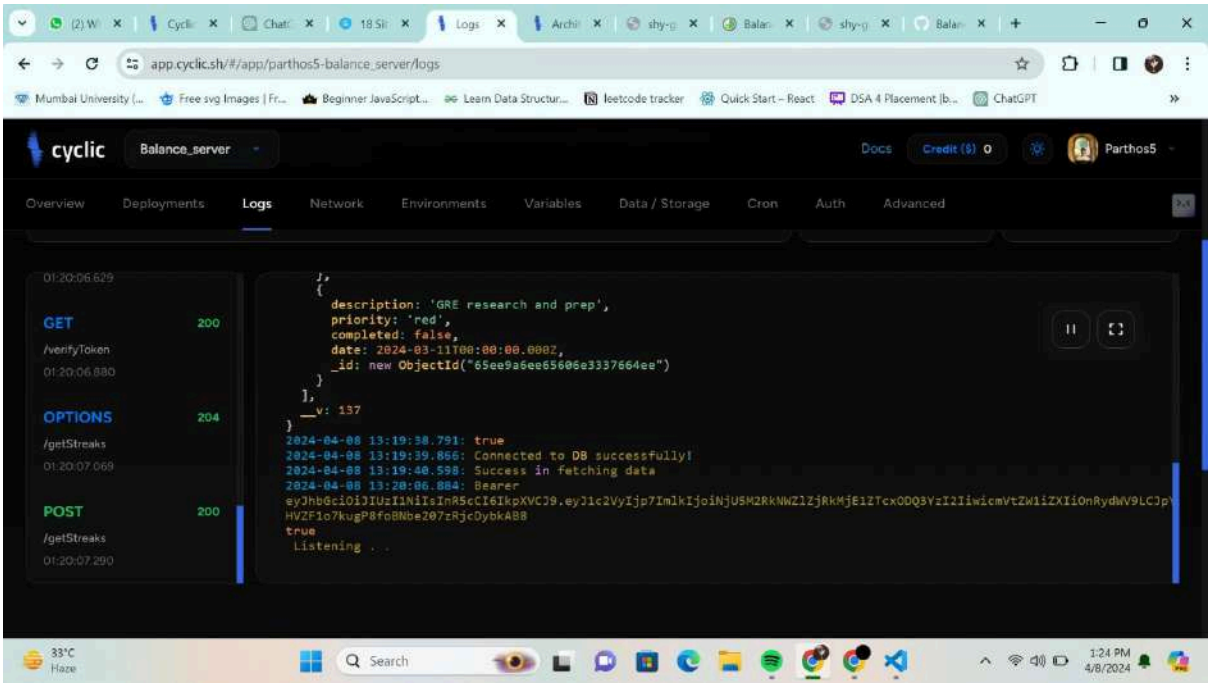
```
{
   "message": "success",
   "tasks": [
      {
```

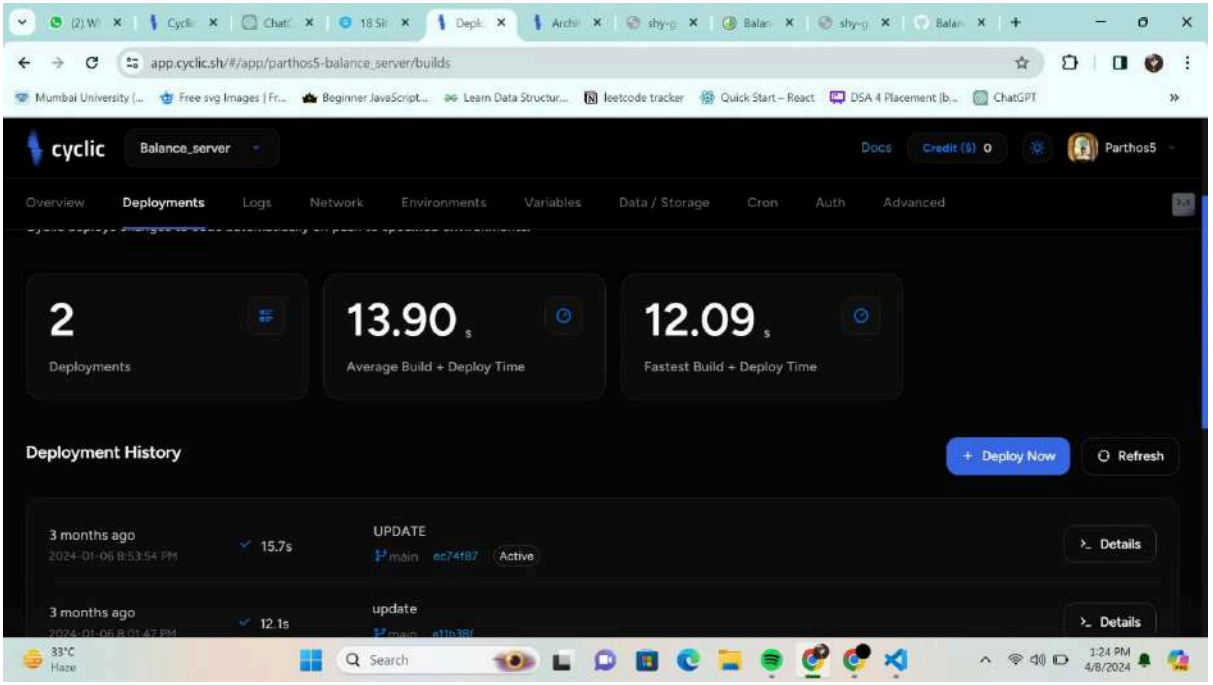
"description": "Research ppr + us unis",
"priority": "red",
"completed": false,
"date": "2024-03-11T00:00:00.000Z",
"_id": "65ee9a52e65606e3337664e8"
},
{
"description": "GRE research and prep",
"priority": "red",
"completed": false,
"date": "2024-03-11T00:00:00.000Z",
"_id": "65ee9a6ee65606e3337664ee"
},
{
"description": "Experiments and assignments",
"priority": "blue",
"completed": false,
"date": "2024-03-11T00:00:00.000Z",
"_id": "65ee9a23e65606e3337664df"
}
]
}

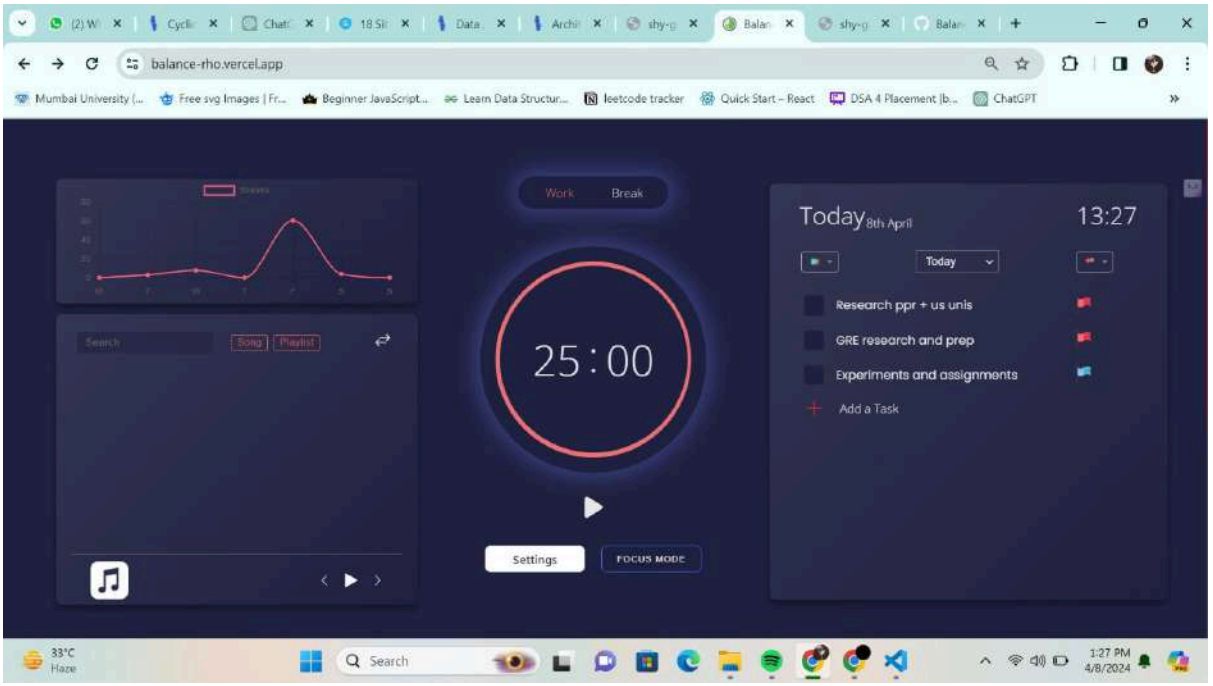
7. Screenshots of project in sequence

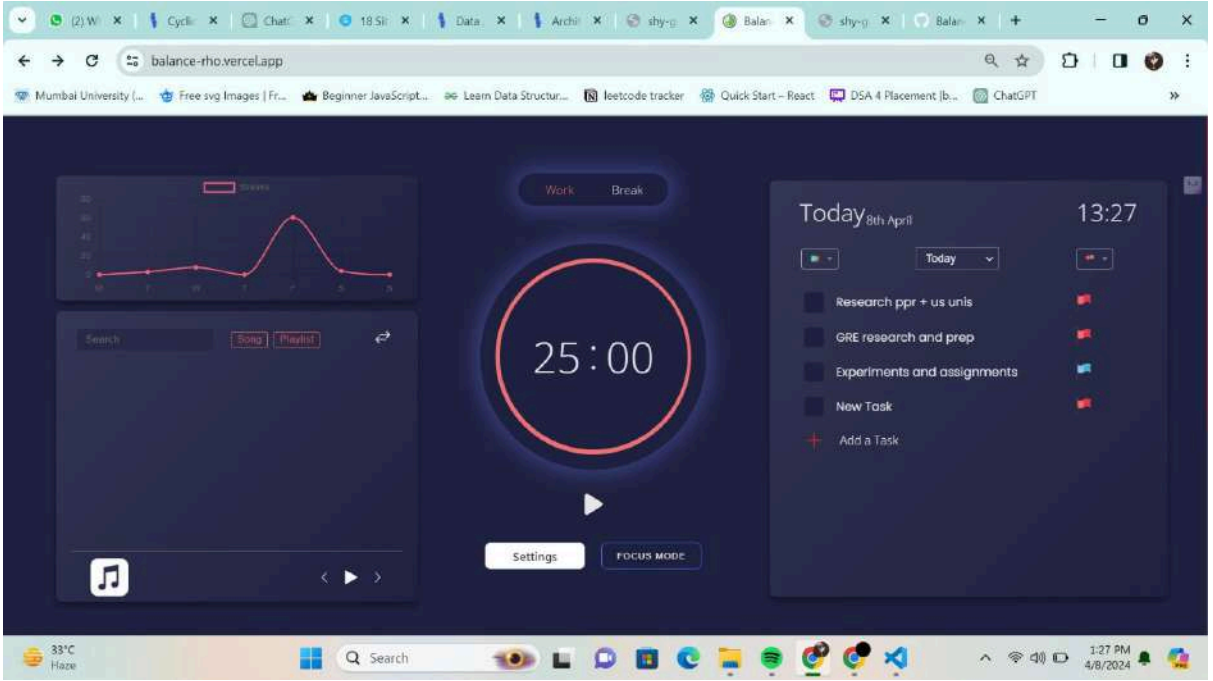



app.cyclic.sh/#/app/parthos5-balance_server/overview
cyclic
Balance_server
Docs
Credit ($) 0
Parthos5
balance_server
prod
Open App
Overview
Deployments
Logs
Network
Environments
Variables
Data / Storage
Cron
Auth
Advanced
Overview
Here's a quick overview of your app.
URL
https://shy-gold-gopher-robe.cyclic.app
App Repo
https://github.com/Parthos5/Balance_server
Request Metrics
Most recent requests for the selected time period.


app.cyclic.sh/#/app/parthos5-balance_server/logs
cyclic
Balance_server
Docs
Credit ($) 0
Parthos5
Overview
Deployments
Logs
Network
Environments
Variables
Data / Storage
Cron
Auth
Advanced
GET 200
/verifyToken
OPTIONS 204
/getStreaks
POST 200
/getStreaks
description: 'GRE research and prep',
priority: 'red',
completed: false,
date: 2024-03-11T00:00:00.000Z,
_id: new ObjectId("65ee9a6ee65606e3337664ee")
__v: 137
2024-04-08 13:19:38.791: true
2024-04-08 13:19:39.866: Connected to DB successfully!
2024-04-08 13:19:40.598: Success in fetching data
2024-04-08 13:20:06.884: Bearer
true
Listening . .


app.cyclic.sh/#/app/parthos5-balance_server/builds
cyclic
Balance_server
Docs
Credit ($) 0
Parthos5
Overview
Deployments
Logs
Network
Environments
Variables
Data / Storage
Cron
Auth
Advanced
2
Deployments
13.90 s
Average Build + Deploy Time
12.09 s
Fastest Build + Deploy Time
Deployment History
+ Deploy Now
Refresh
3 months ago
2024-01-06 8:53:54 PM
15.7s
UPDATE
main
Active
Details
3 months ago
12.1s
update
Details


balance-rho.vercel.app
Work
Break
25:00
Settings
FOCUS MODE
Today 8th April
13:27
Today
Research ppr + us unis
GRE research and prep
Experiments and assignments
Add a Task



8.Conclusion

In conclusion, the development of a real-time application leveraging Node.js, MongoDB, and integration with Cyclic.sh for cloud infrastructure management presents a robust solution for achieving scalable and efficient storage services within a cloud environment. By meticulously following the outlined steps and adhering to the specified requirements, a functional application can be crafted to demonstrate the prowess of cloud-based storage solutions while showcasing adept utilization of contemporary technologies.

One of the paramount achievements of this project lies in its scalability and performance. Through thoughtful design considerations and integration with Cyclic.sh, the application can seamlessly adapt to increasing workloads and concurrent users. Automated scaling and monitoring mechanisms inherent in Cyclic.sh ensure that the application maintains optimal performance levels, even under substantial load.

Moreover, the incorporation of real-time communication features using WebSockets or server-sent events enhances the application's interactivity and user engagement. This capability fosters instant updates and facilitates smooth communication between clients and the server, thereby enriching the overall user experience and promoting collaborative interactions.
In essence, through the development and deployment of this real-time application, a profound understanding of cloud infrastructure (Storage as a Service) is demonstrated, coupled with a demonstration of adeptness in employing modern technologies to tackle real-world challenges. This project exemplifies the transformative potential of cloud-based solutions and underscores their significance in the realm of contemporary software development.