Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr. ~~Miss~~ Atharva Jadhav

of I. T. Department, Semester V with

Roll No. 44 has completed a course of the necessary

experiments in the subject Advance DevOps under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Agarwal

Teacher In-Charge

Head of the Department

Date 20/10/2023.

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	DATE	PAGE	TEACHERS SIGN.
1.	To study and perform the setup of AWS EC2 service & launch EC2 instance.	18/7/23		
2.	To study and perform the setup of AWS cloud9 service & launch a python program in cloud9	25/7/23		
3.	To study AWS codepipeline & deploy web application using codepipline.	9/8/23		
4.	To study AWS S3 service & create a bucket hosting web application.	1/8/23		
5.	To understand Kubernetes Cluster architecture.	13/10/23		
6.	To understand terraform lifecycle and to build, change & destroy AWS infrastructure using Terraform.	22/8/23		
7.	To perform static analysis on python using SonarQube	29/8/23		
8.	To understand continous monitoring using Nagios.	12/9/23		
9.	To understand AWS lambda function and create a lambda function.	5/9/23		
10.	To create a lambda function using python for adding data to DynamoDB database.	5/9/23		
11.	Written Assignment 1	1/8/23		
12.	Written Assignment 2	13/10/23		

Agarwal 20/10/23

Lab Assginment -01

Aim- To make a EC2 Machine in AWS

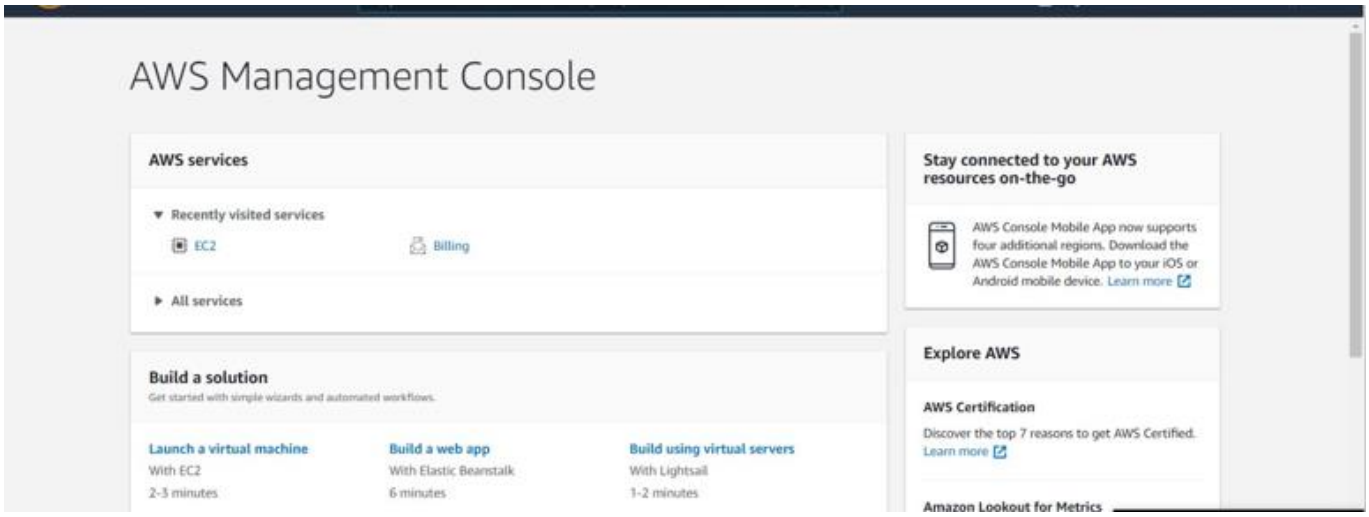
THEORY-

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

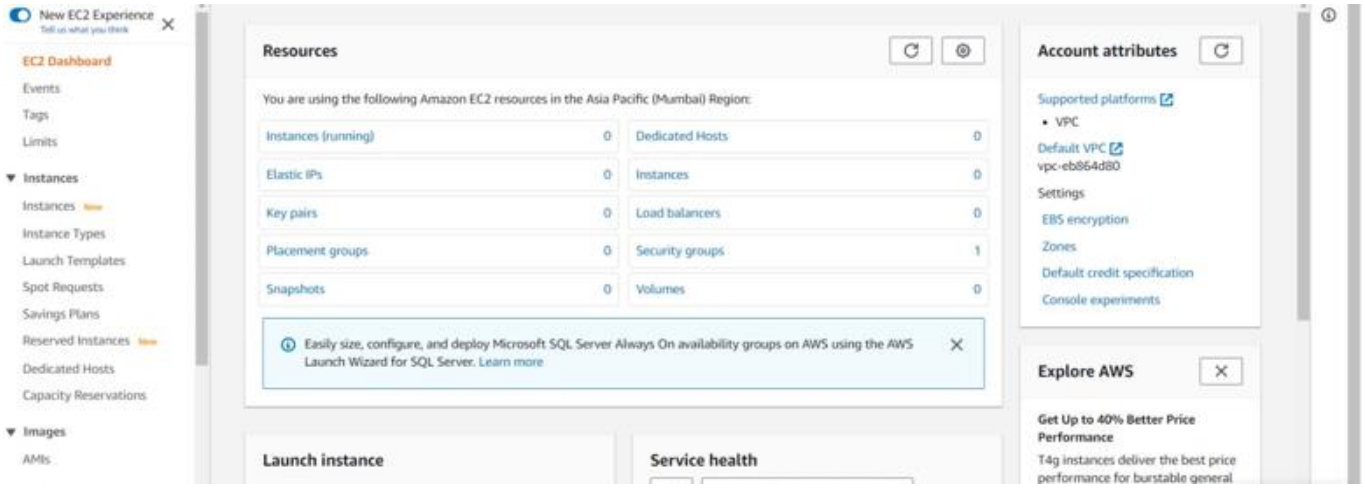
STEPS-

LOGIN TO AWS ACCOUNT,

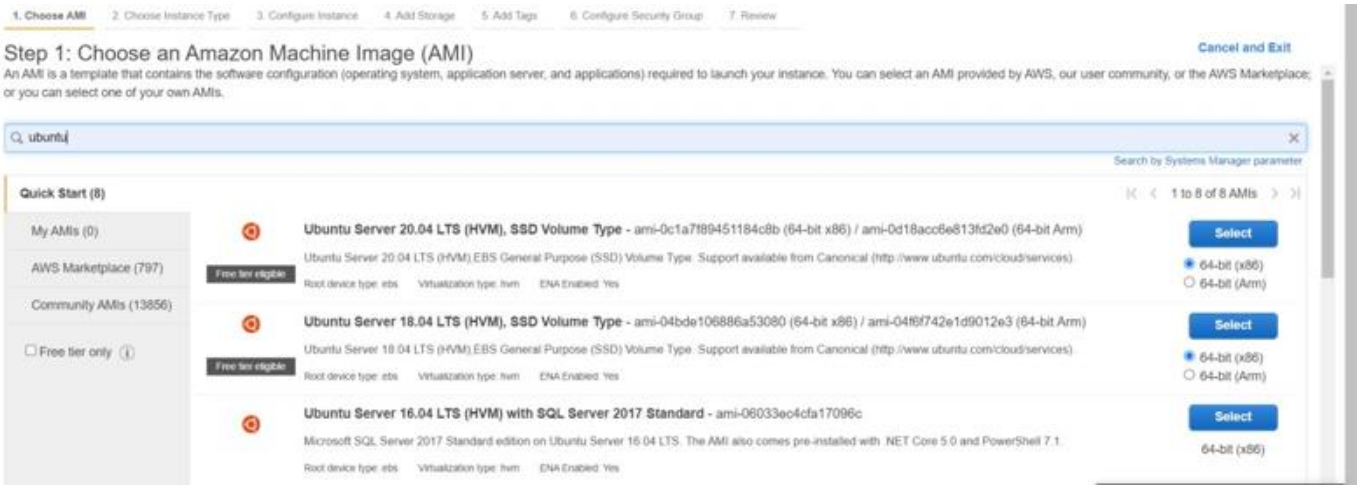
THEN SEACH EC2.



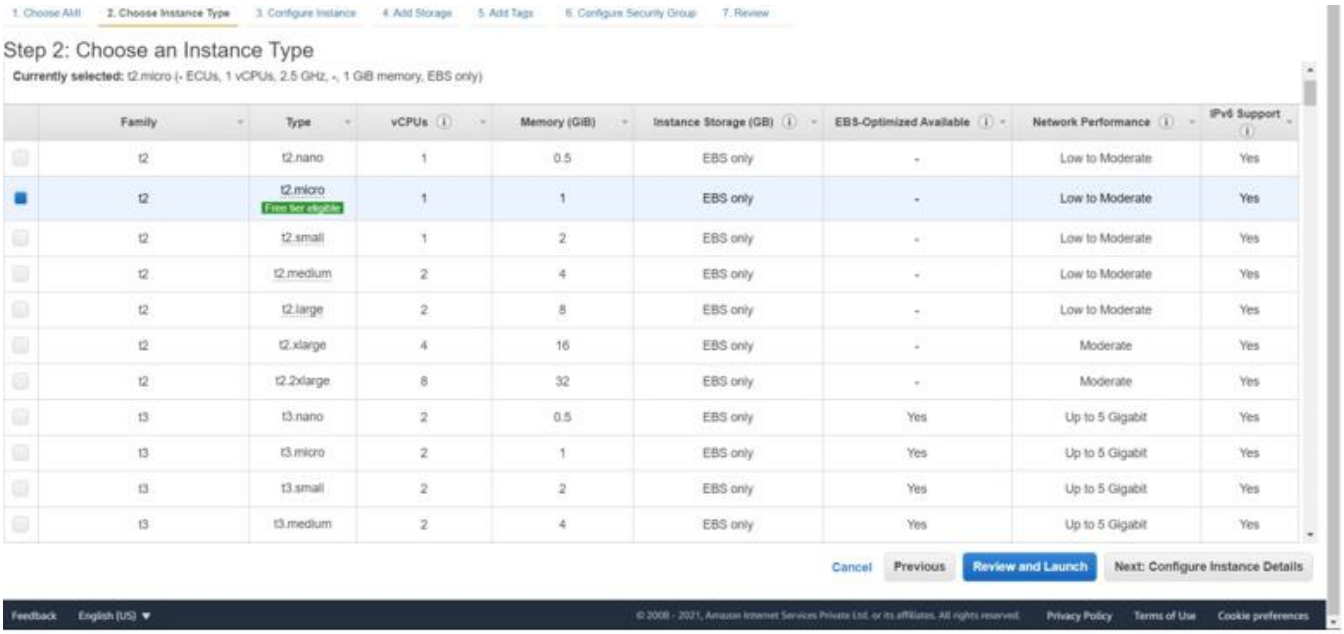
NOW CLICK ON LAUNCH / CREATE NEW INSTANCES.



Choose any machine you want to create here I am creating UBUNTU(free tier).



Click on T2 micro (free tier one)



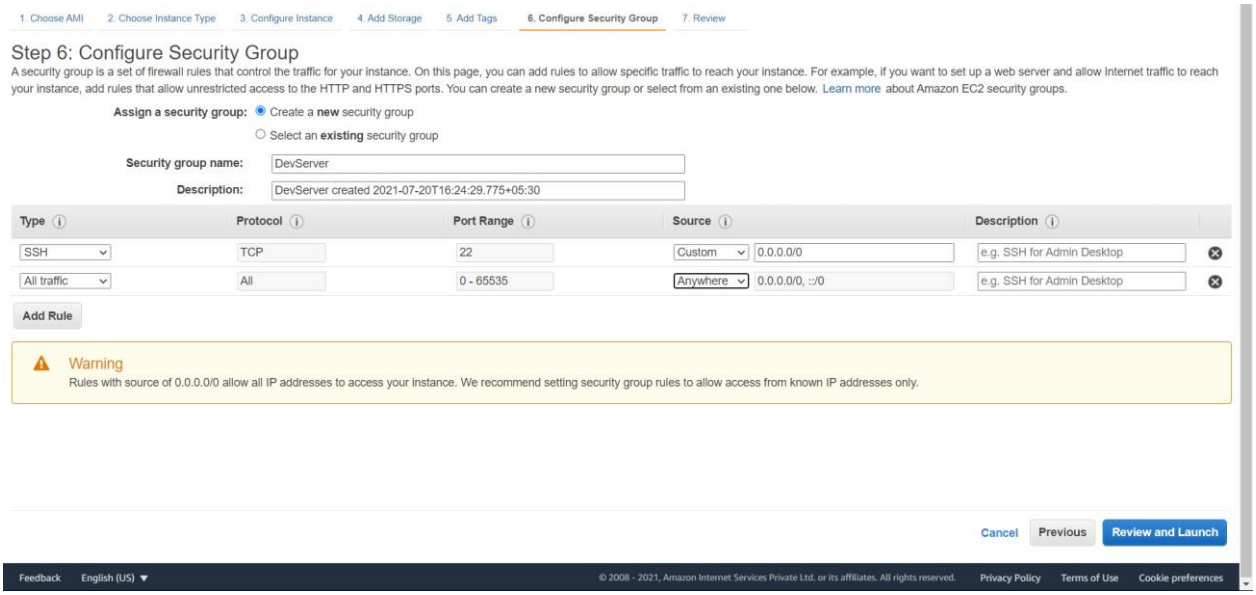
Click on NEXT, then Again Click Next.

THEN CREATE A KEY PAIR BY ANY NAME AND DOWNLOAD IT. THEN CLICK NEXT

Now add security group ALL TRAFFIC ,

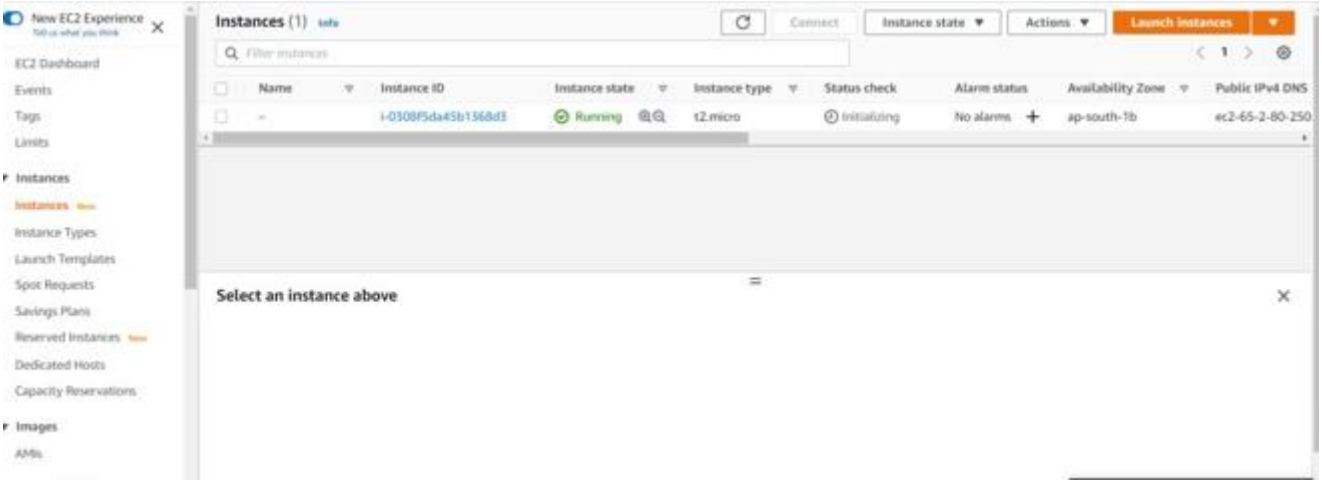
PROTOCOL – ALL,

SOURCE- ANYWHERE.

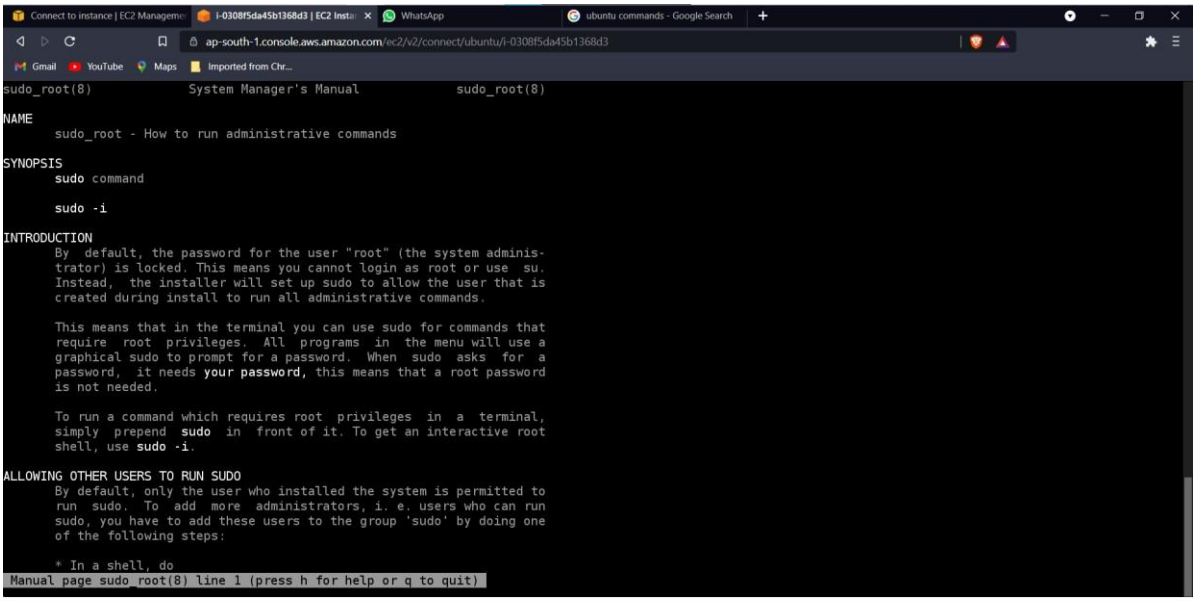


NOW WAIT TILL THE STATUS CHECK IS 2/2 and Instance is running.

Once Check is complete click on launch instances.

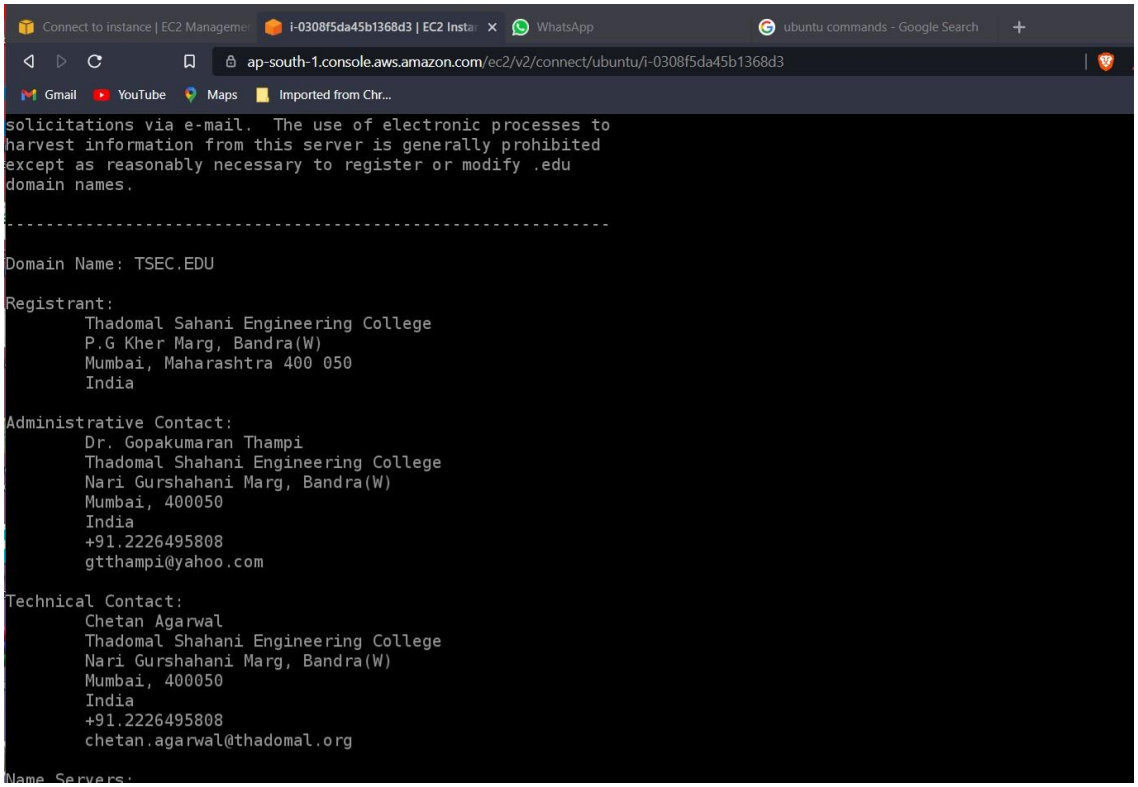


FOLLOW SOME BASIC LINUX COMMANDS AS SHOWN BELOW-



i-0308f5da45b1368d3

Public IPs: 65.2.80.250 Private IPs: 172.31.8.68



Lab Outcome:

LO1 is mapped.

LO1: To Understand the fundamental of Cloud Computing and fully procficient with Cloud based DevOps Solution deployment options to meet your business requriements.

CONCLUTION – HENCE LEARNED AND IMPLEMETED THE STEPS TO CREATE AN EC2 MACHINE.

LAB ASSGINMENT -02

AIM- To create a Cloud9 Environment.

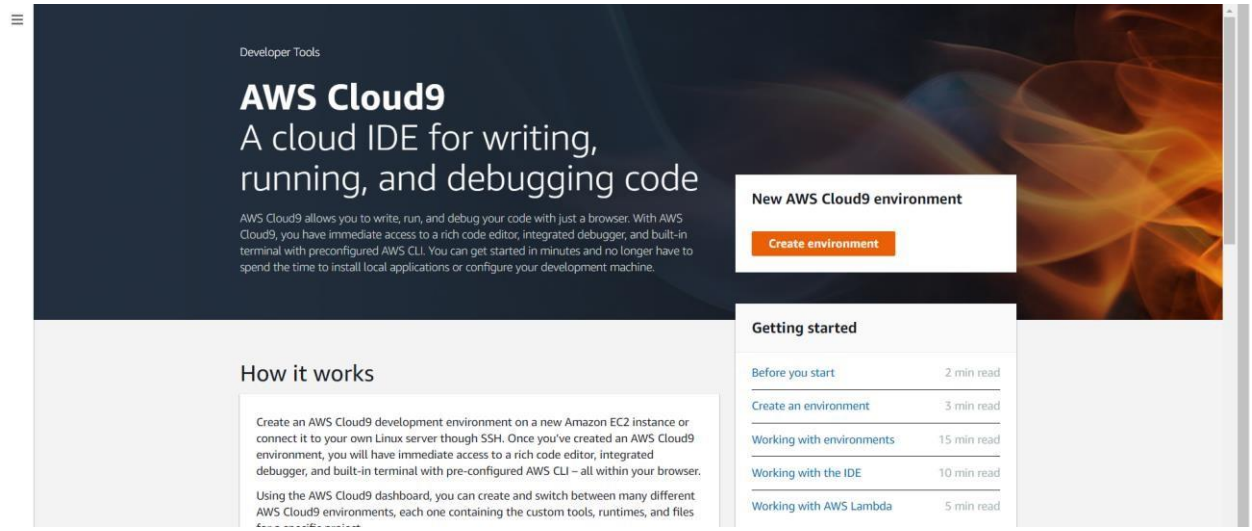
Theory-

Cloud9 IDE is an Online IDE, published as open source from version 2.0, until version 3.0. It supports multiple programming languages, including C, C++, PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go. It is written almost entirely in JavaScript, and uses Node.js on the back-end.

STEPS-

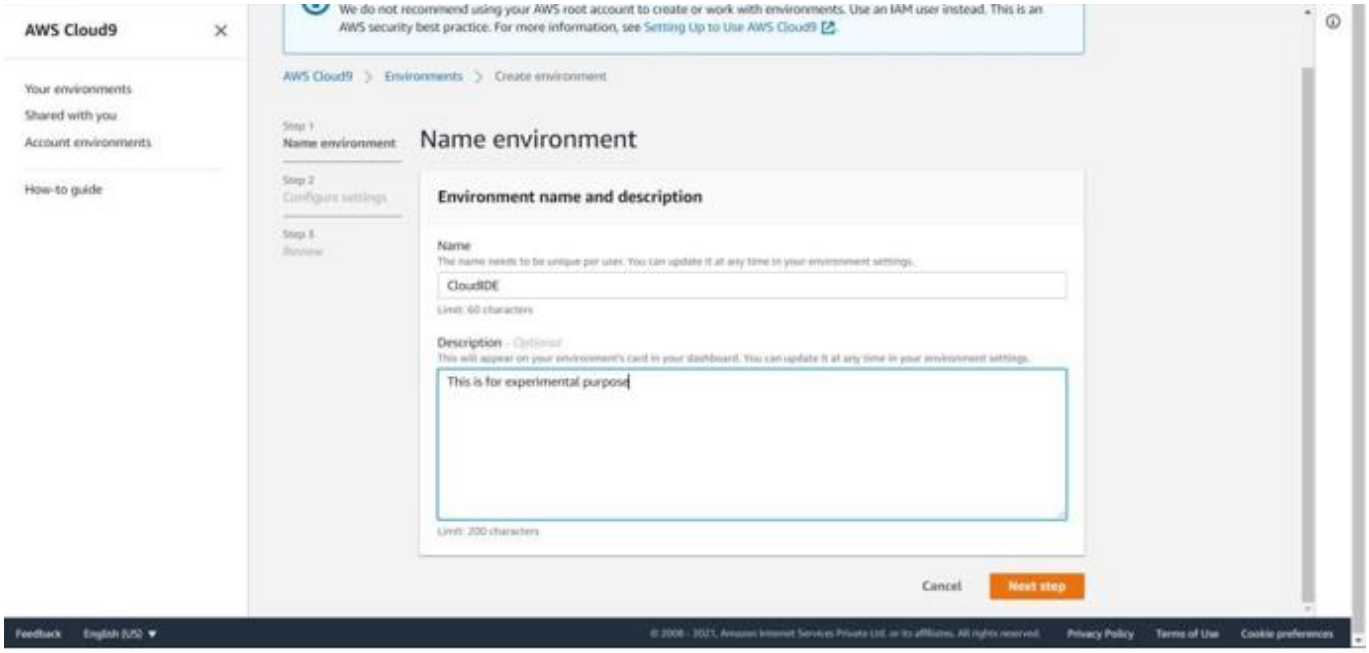
LOG IN TO YOUR AWS ACCOUNT,

SEARCH FOR CLOUD 9 IN THE SEARCH BAR

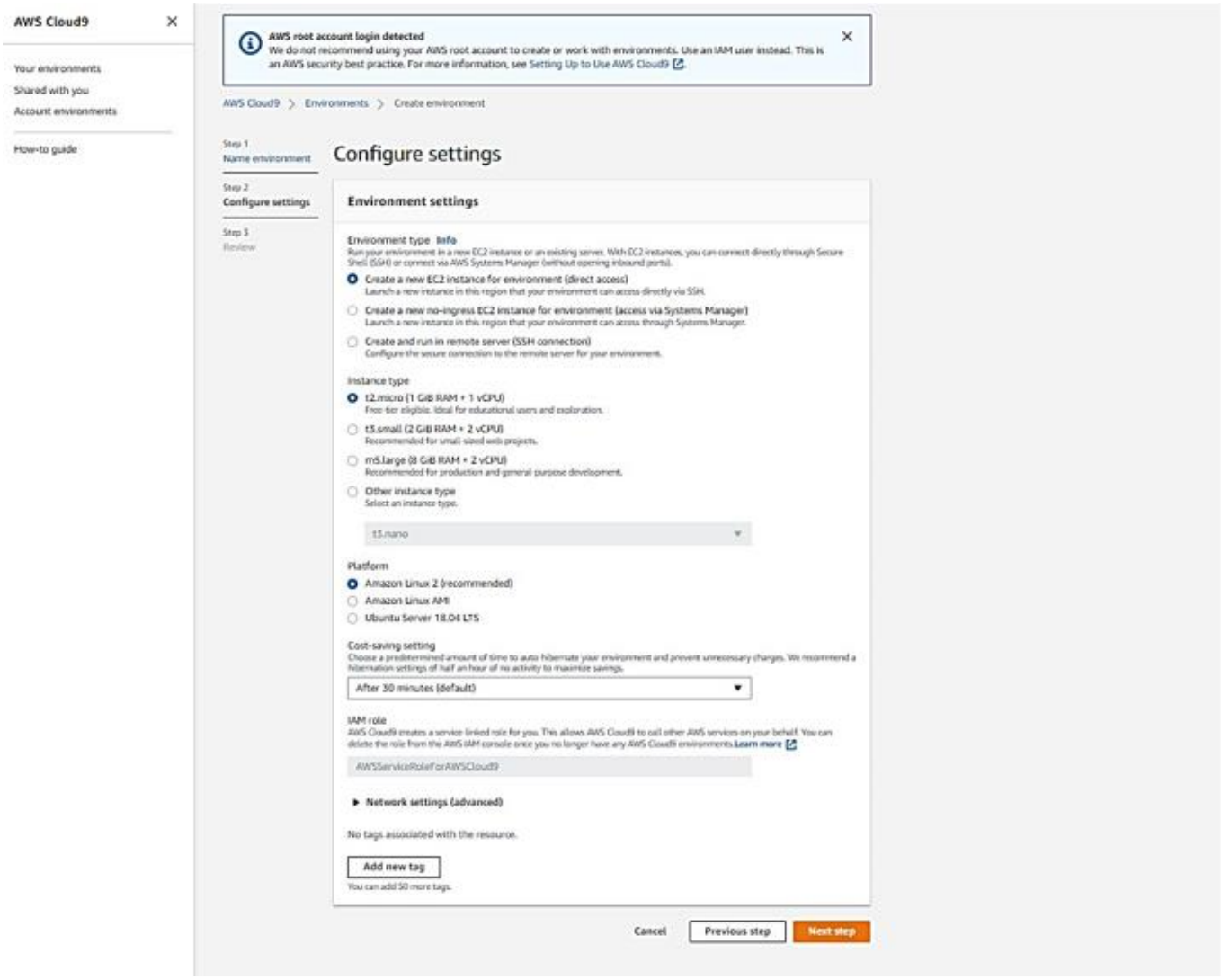


CLICK ON CREATE ENVIRONMNET,

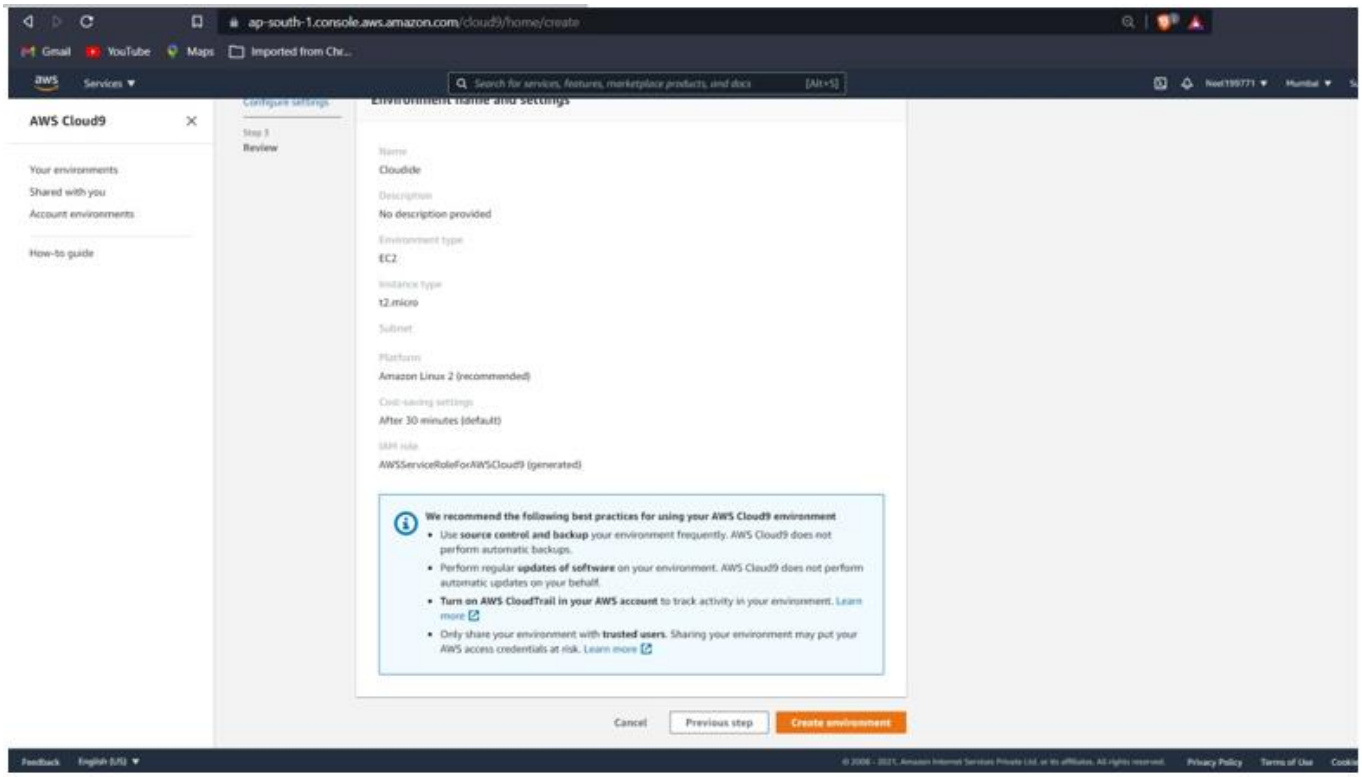
NAME THE ENVIRONMNET



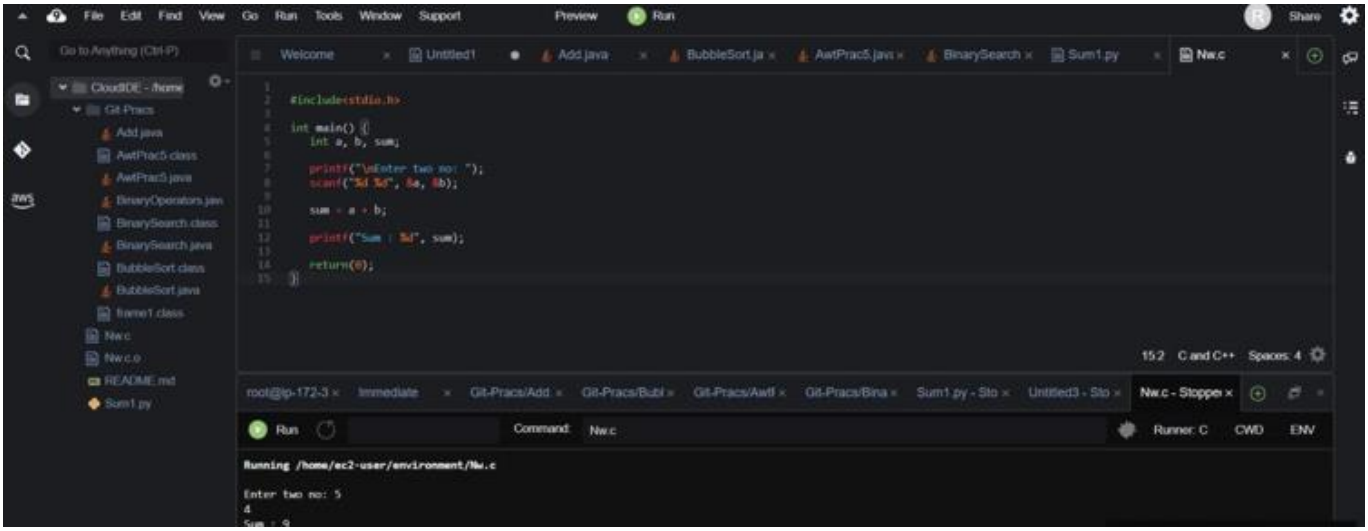
Now click on next step.



Again, click on Next Step,
Now click on Create Environment.



Now select any coding language and perform any operation via a code. Shown below



Conclusion – Hence learned and implemented steps to Create an Cloud9 environment.

Name : Atharva Jadhav
Roll No. : 2105044

ADVANCE DEVOPS LAB -3

AIM- To study AWS S3 service and create a bucket for hosing static web application.

THEORY-

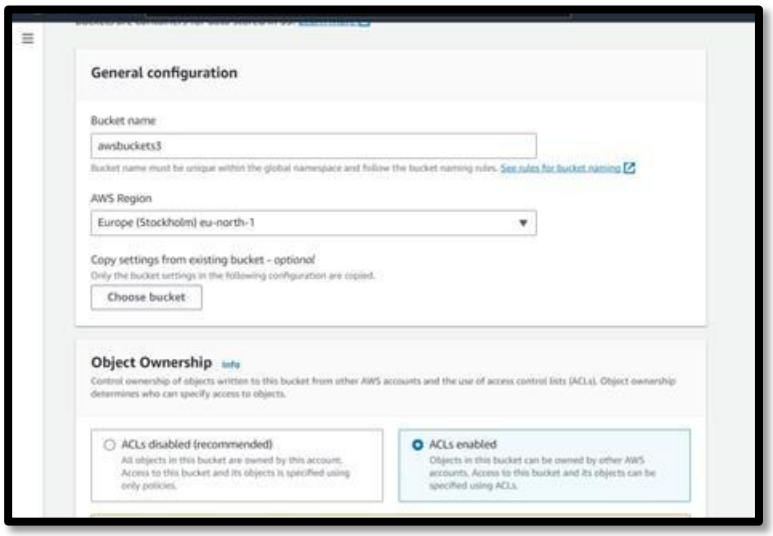
AWS Simple Storage Service (S3) from the aforementioned list, S3, is the object storage service provided by AWS. It is probably the most commonly used, go-to storage service for AWS users given the features like extremely high availability, security, and simple connection to other AWS Services.

An Amazon S3 bucket can be set up to operate similarly to a website. This section illustrates how to host a website using Amazon S3. There are mainly 7 steps to hosting a static website using Amazon Web Service(AWS) S3.

Step 1: Creating a Bucket

1. First, we have to launch our S3 instance. Follow these steps for creating a Bucket
2. Open the Amazon S3 console by logging into the AWS Management Console at https://console.aws.amazon.com/s3/.
3. Click on Create Bucket.

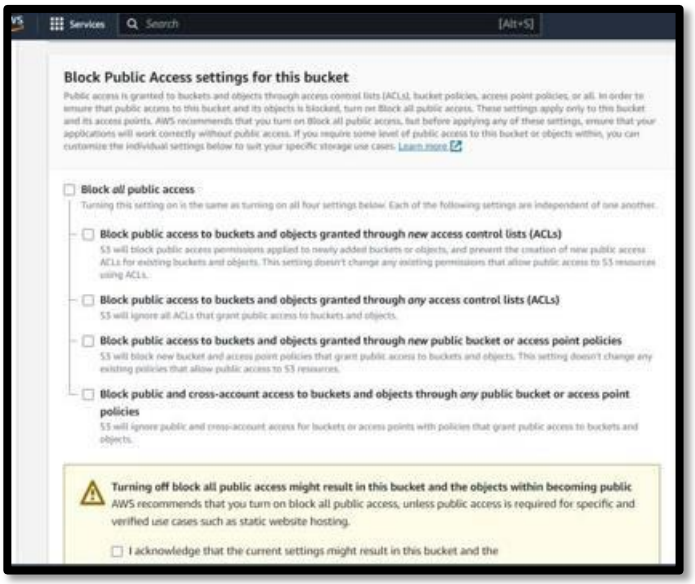
4. Choose Bucket Name – Bucket Name Should be Unique
5. Object Ownership – Enable for making Public, Otherwise disable



Name : Atharva Jadhav
Roll No. : 2105044

Step 2: Block Public Access settings for the bucket

1. Uncheck (Block all public access) for the public, otherwise set default. If you uncheck (Block all public keys).



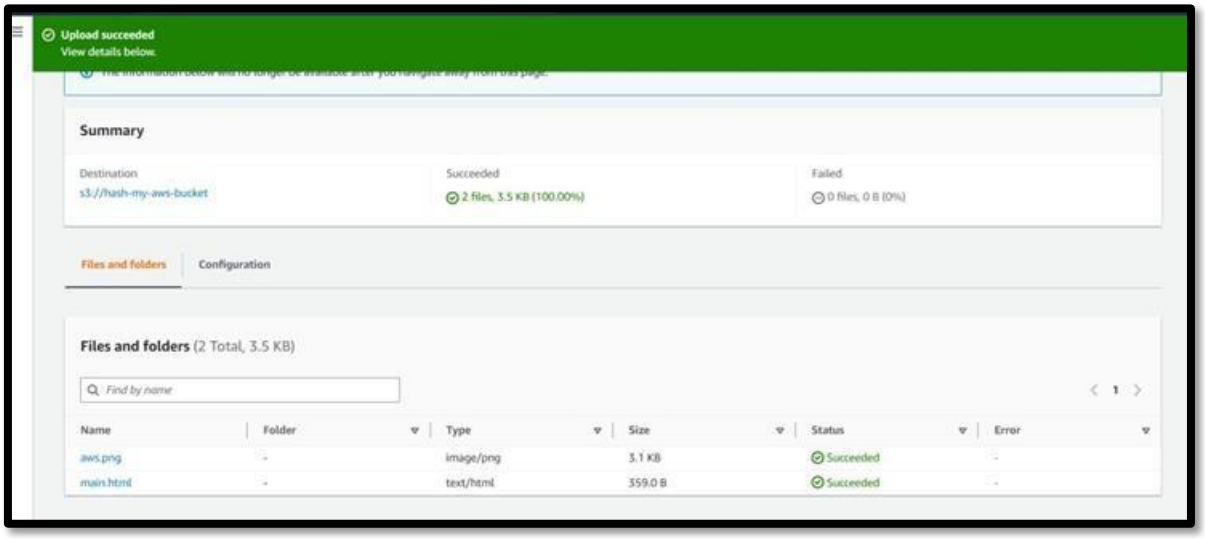
2. Now click on create bucket
3. Bucket is created

Step 3: Now upload code files

Select Bucket and Click your Bucket Name.

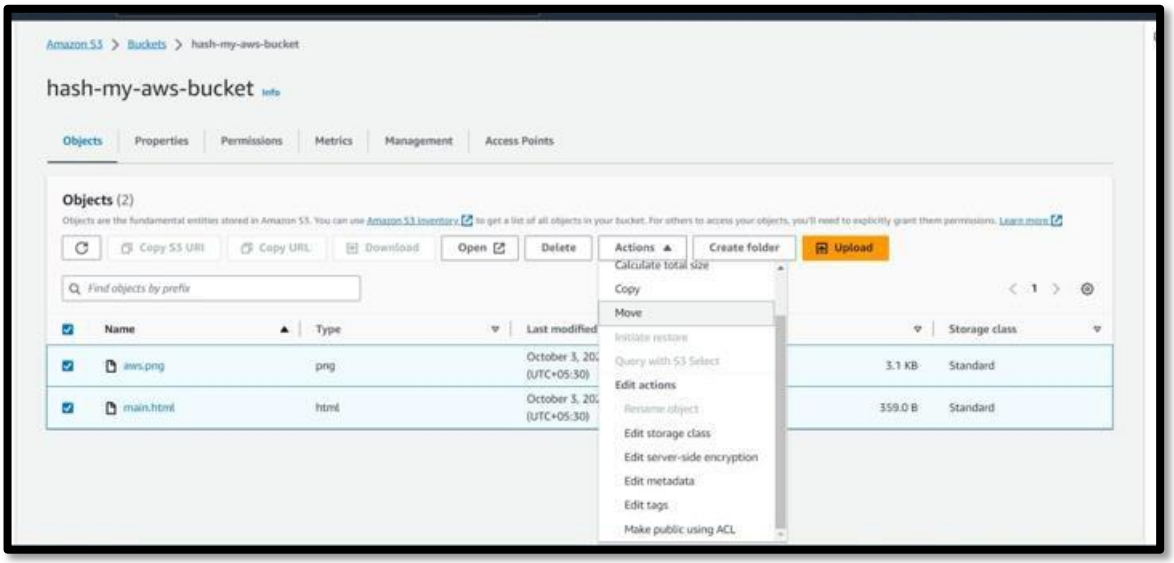
Now, click on upload (then click add File/folder) and select your HTML code file from your PC/Laptop.

Name : Atharva Jadhav
Roll No. : 2105044



Step 4: Once the Files are uploaded successfully, click on Permissions and now follow this Process –

- Block public access
- Object Ownership
- Make public Object



Step 6: Copy your Object URL

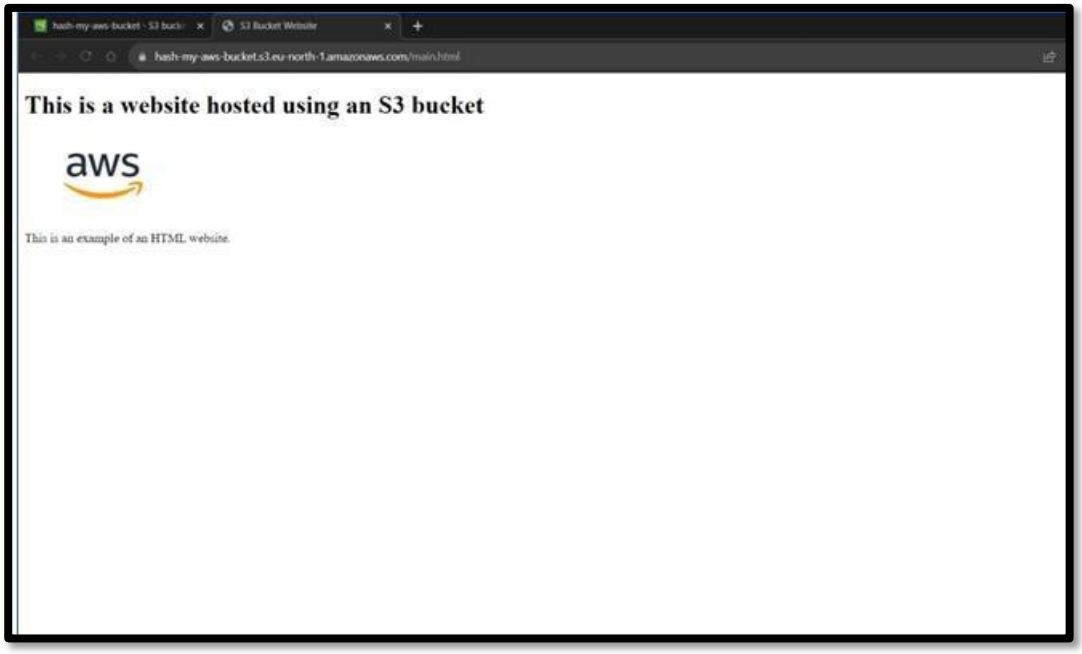
Now, click on your HTML File Object Name.

Copy the Object URL.

Name : Atharva Jadhav
Roll No. : 2105044

Step 7: Check out your Website!

Directly Paste this URL into the Other Tab or your other System.



CONCLUSION:- This experiment demonstrated how to utilize AWS S3, a powerful and scalable cloud storage solution, to host a static web application. S3's ability to serve static content with low latency and high reliability makes it a suitable choice for hosting static websites. By completing this experiment, we gained practical insights into leveraging cloud services for web hosting, which is vital for modern web development and deployment. AWS S3 offers a cost-effective and efficient solution for hosting various types of web applications, contributing to the agility and scalability of web development projects.

ASSIGNMENT 4

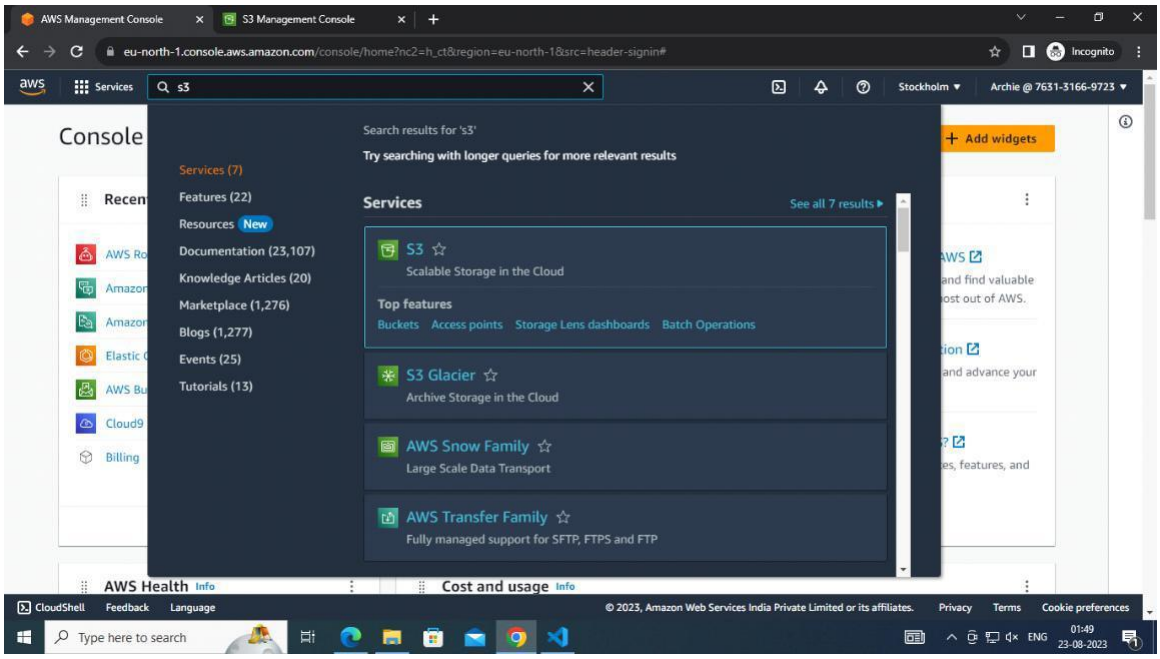
AIM: To build your application using AWS Code Build and Deploy on S3 using AWS Code Pipeline

THEORY:

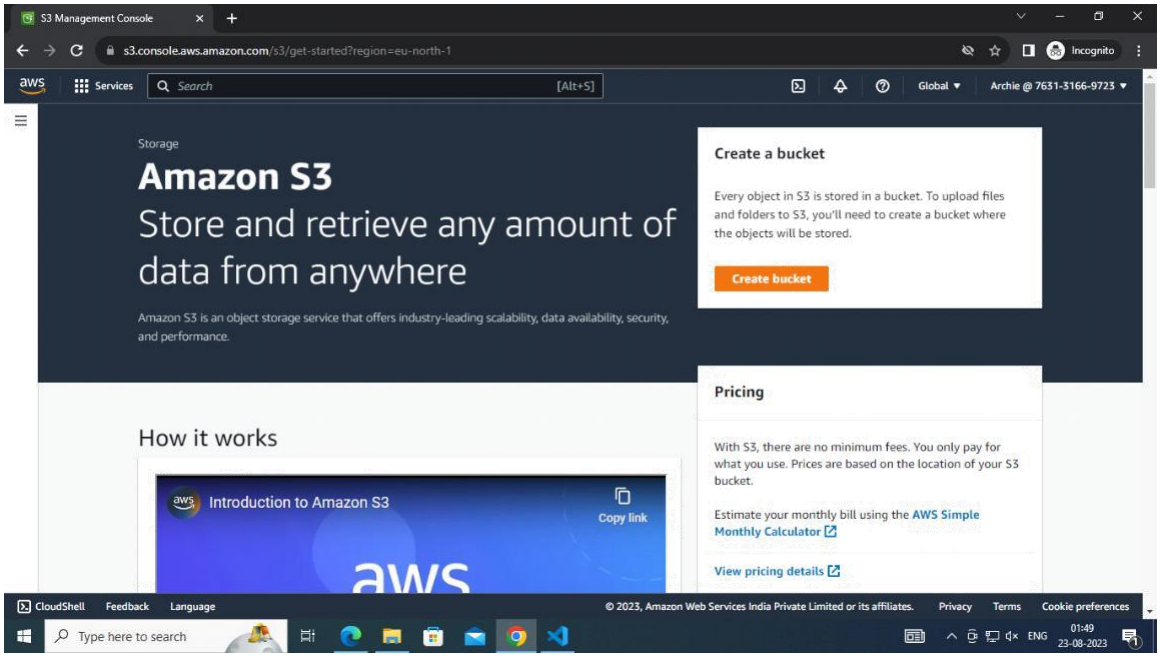
Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps. With cost-effective storage classes and easy-to-use management features, you can optimize costs, organize data, and configure fine-tuned access controls to meet specific business, organizational, and compliance requirements.

STEPS:

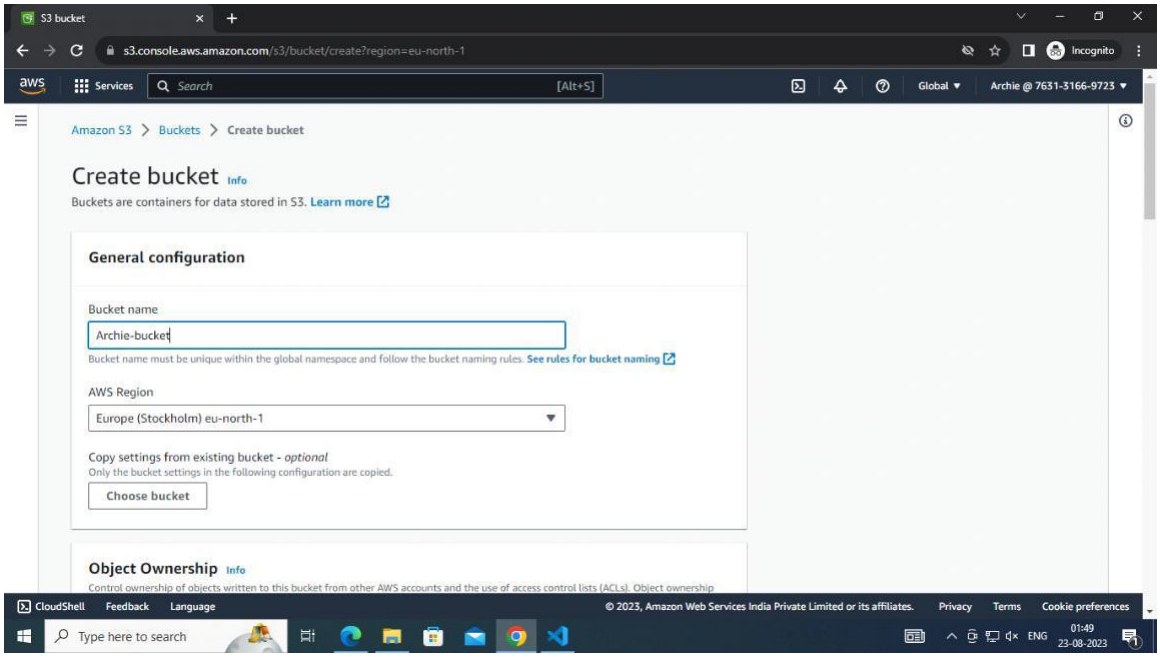
1. Log in as IAM User. Search for S3 in console

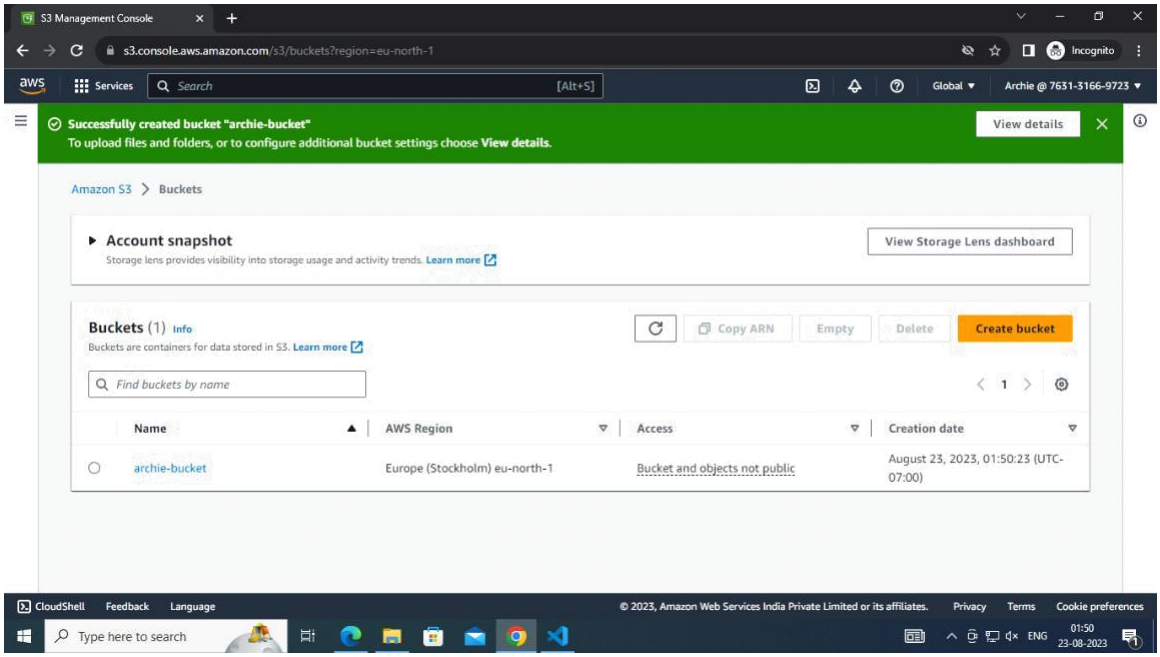


2. Create a new bucket by clicking Create Bucket Buton

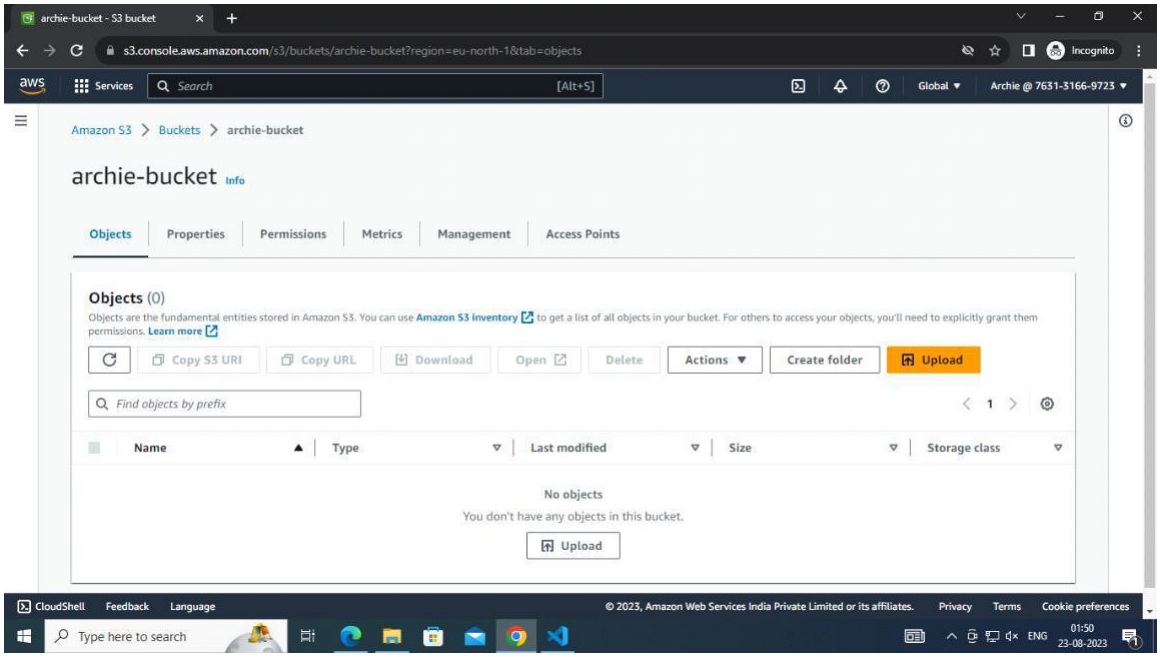


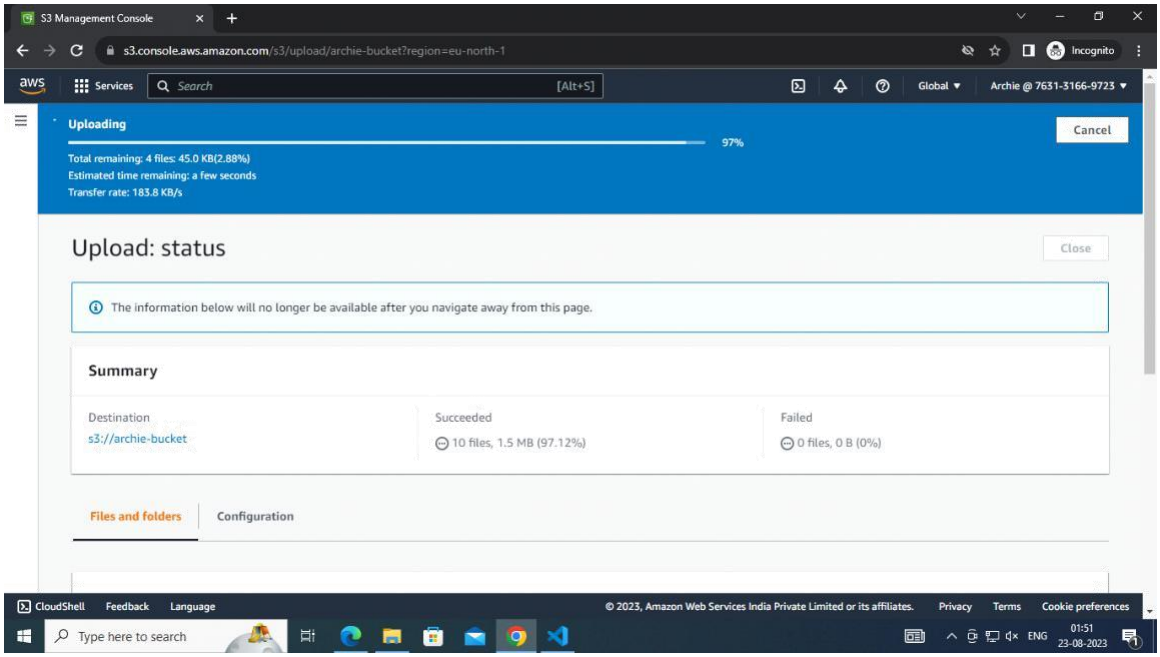
4. Give the bucket a name and click on Create Bucket to make a new bucket.



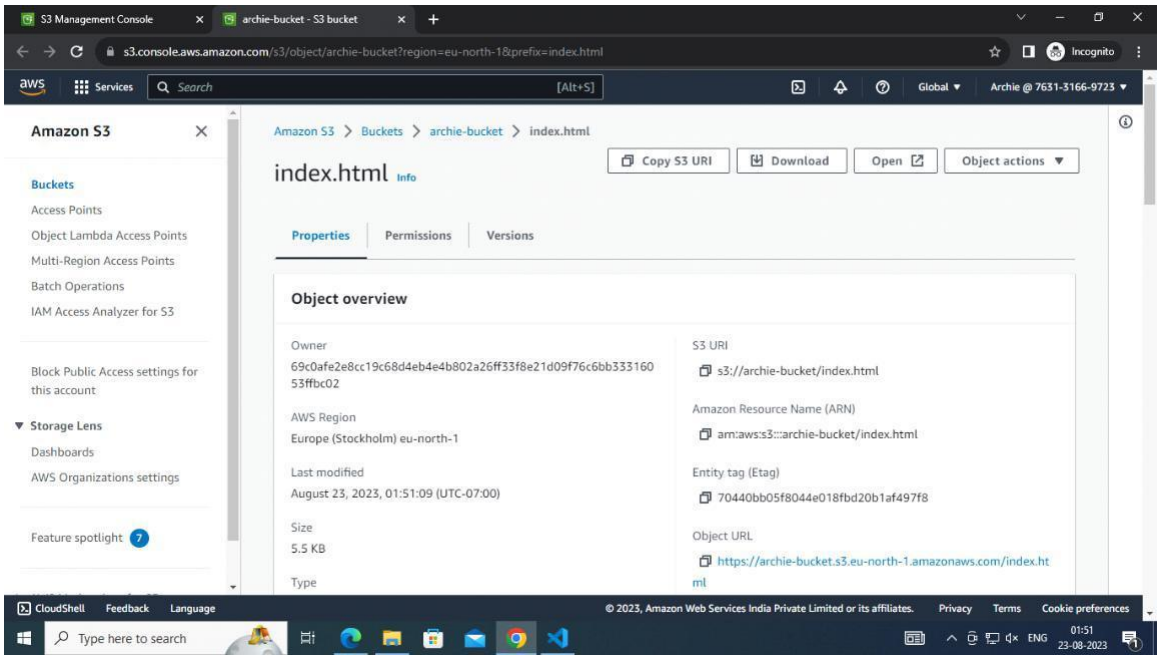


5. Click on Upload button to add files and folders of your projects.

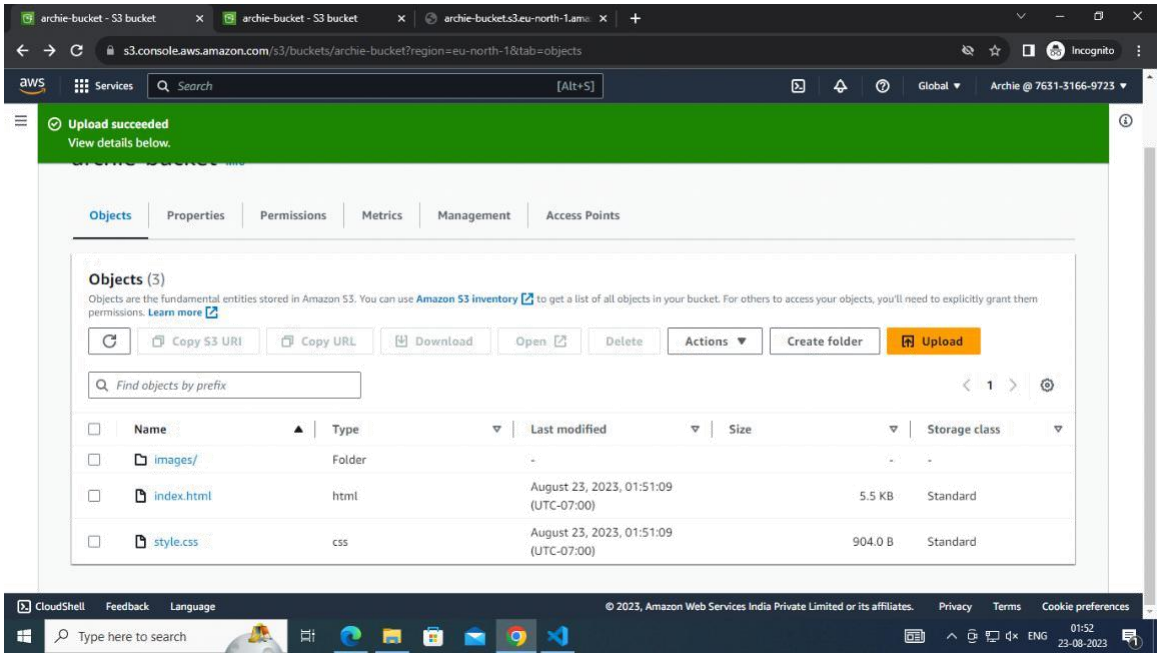




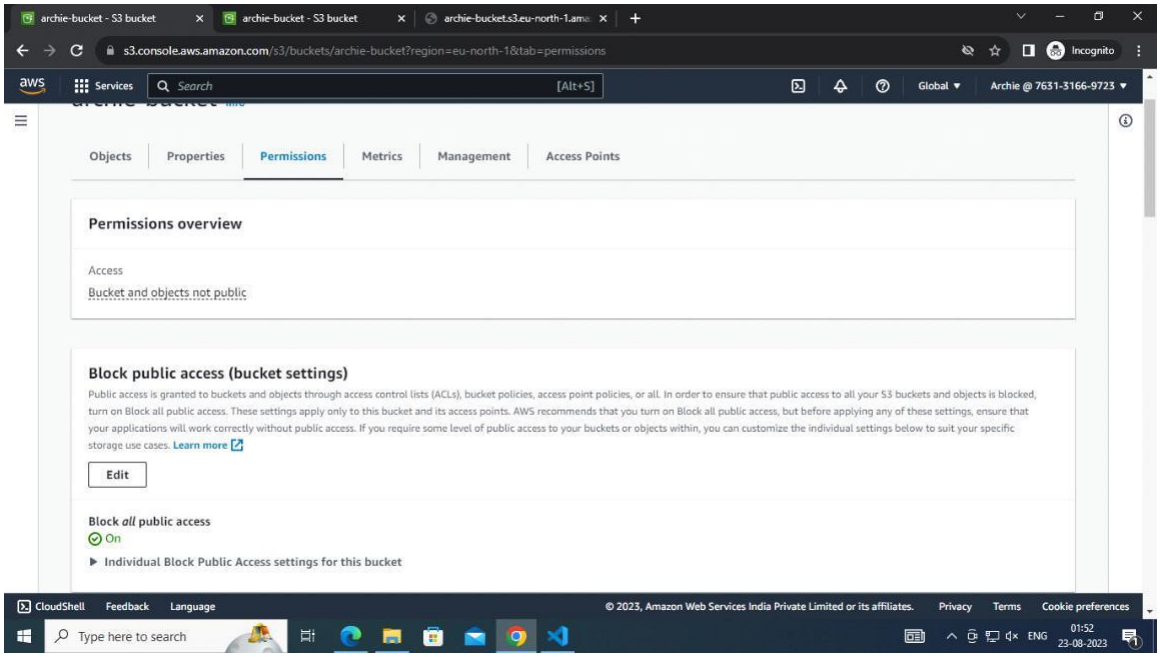
6. Go to the index.html folder and copy the Object URL. Copy this URL to another web page, it will show error.

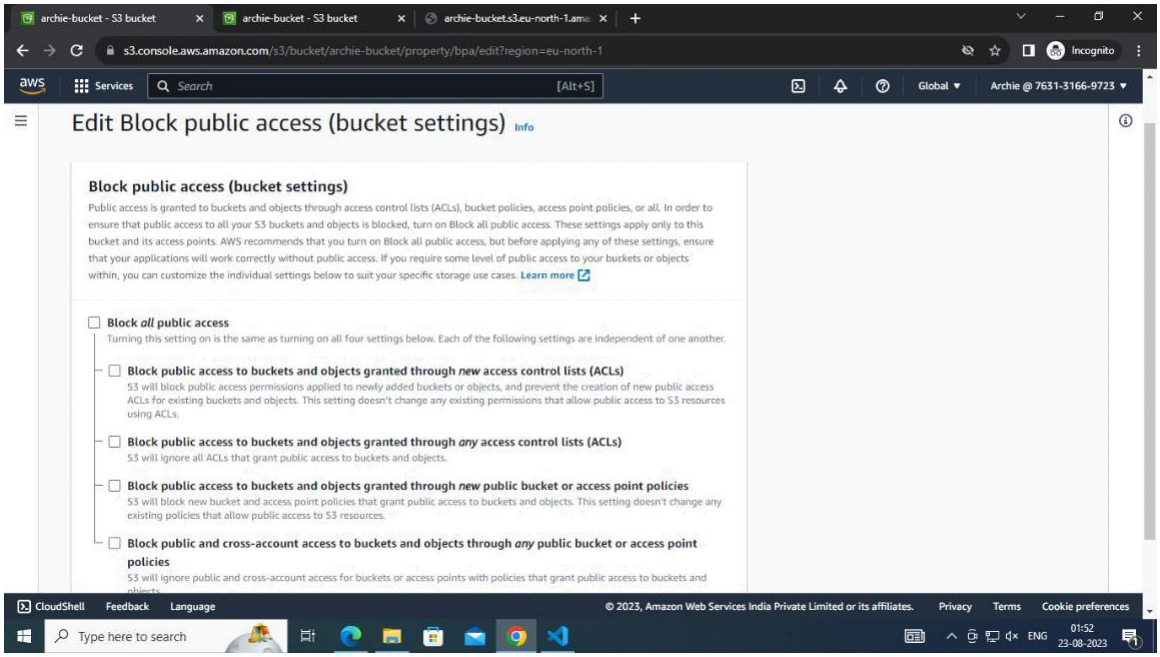


7. Now we need to enable permissions. Click on Premission tab.

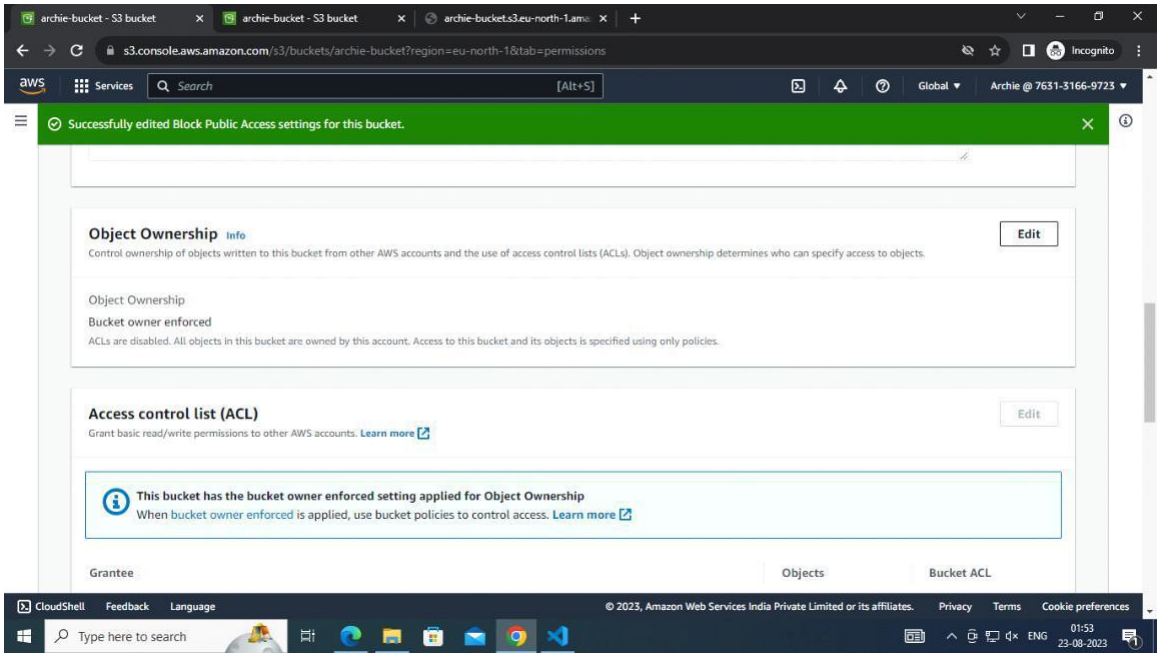


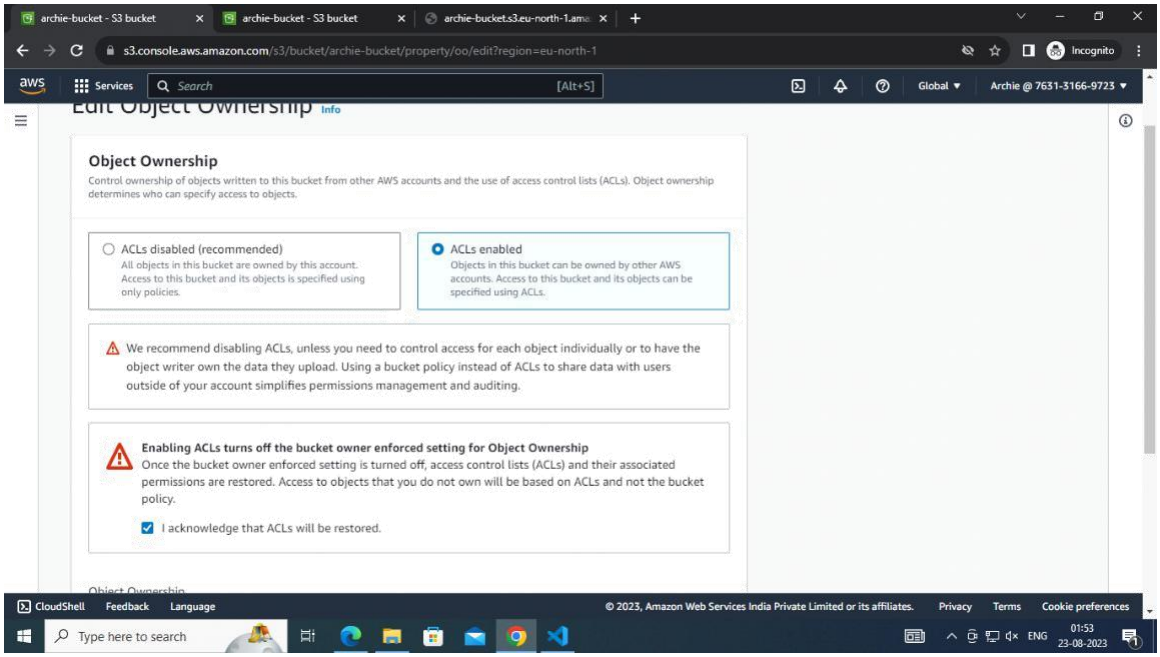
8. Change the public access permission.



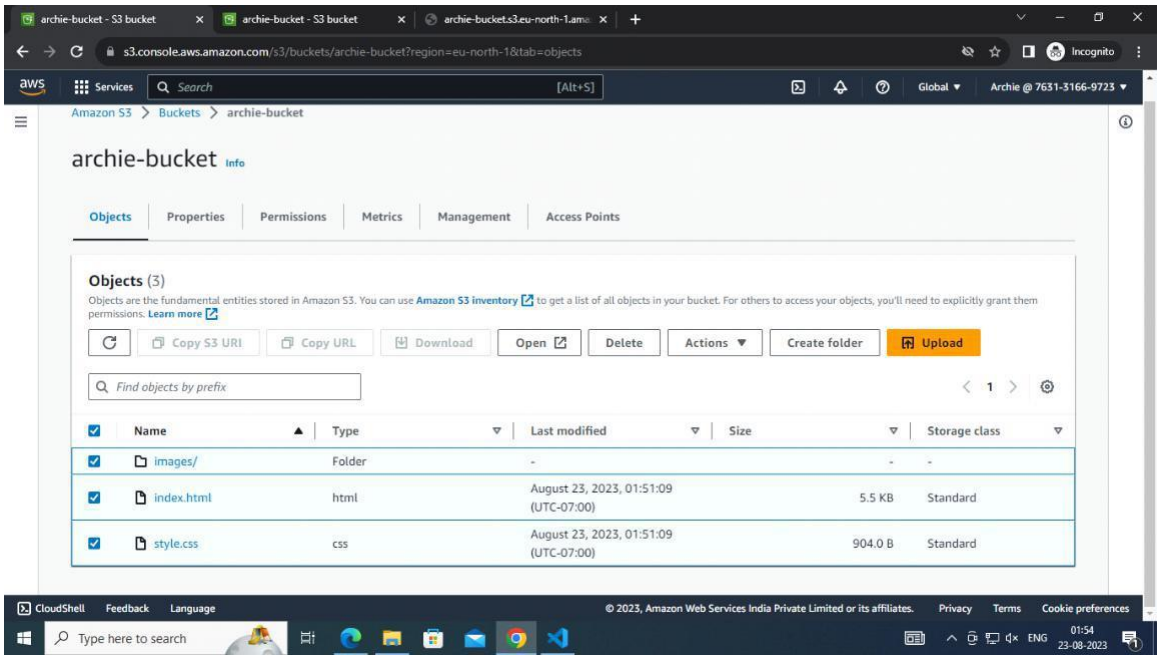


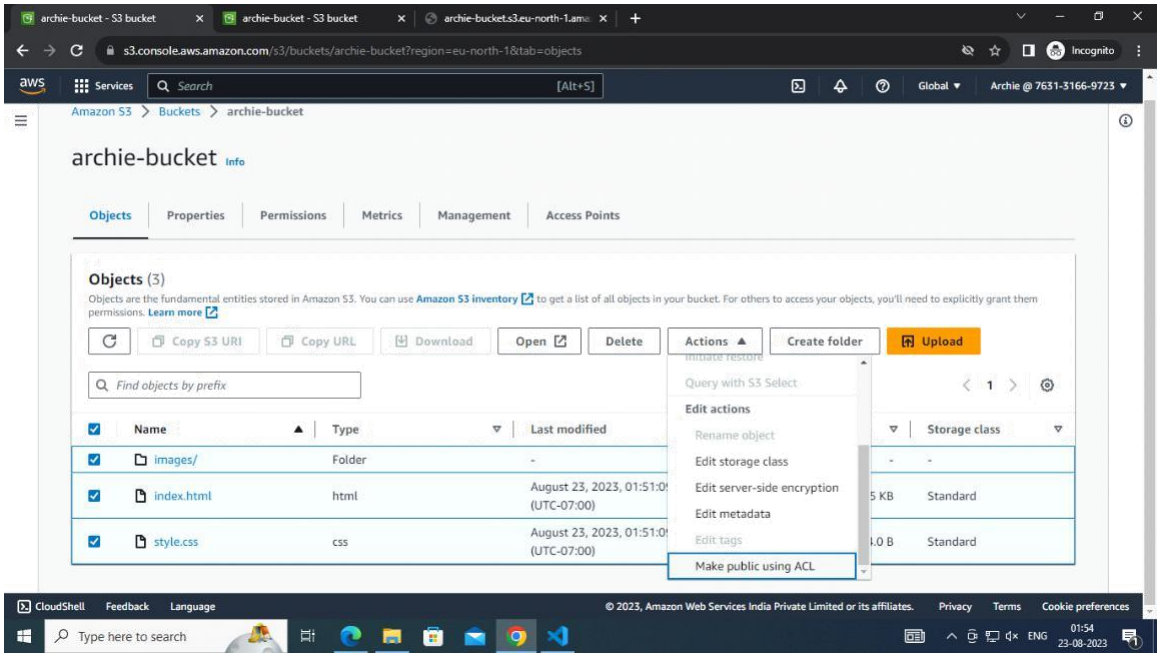
9. Go to the Object Ownership and Enable the ACLs



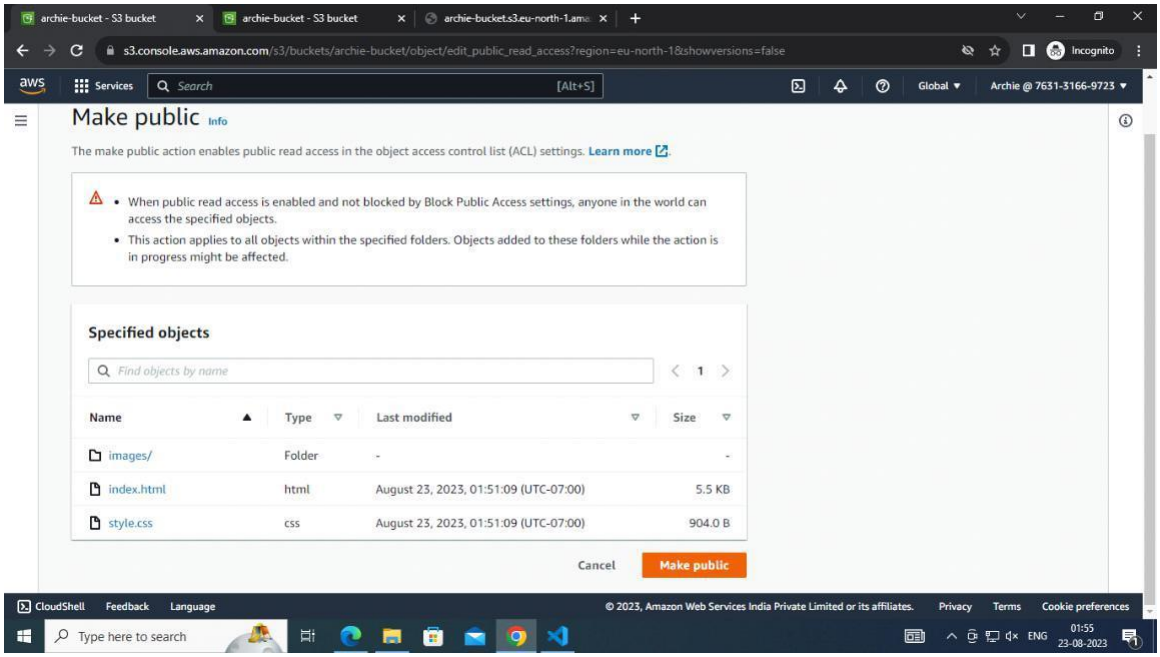


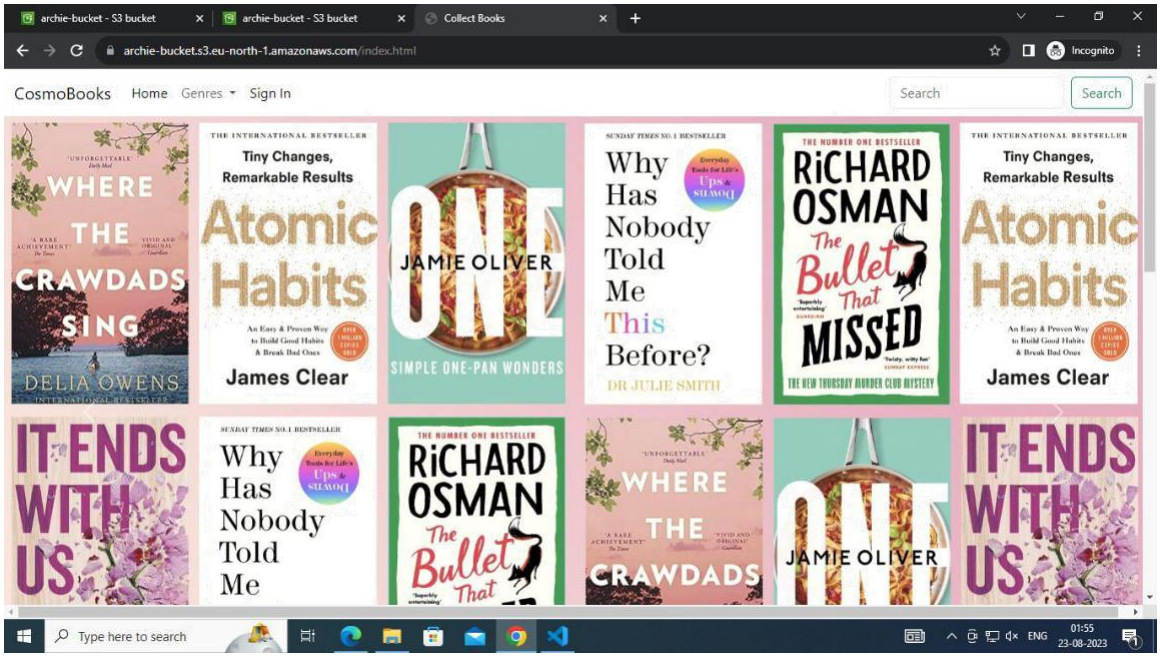
10. Finally select all the files and folders and click on Actions and click on 'Make public using ACL'.





11. Finally click on the 'Make Public' button and reload the web page.





CONCLUSION:

In this assignment, we learnt how to build our application using AWS Code Build and Deploy on S3 using AWS Code Pipeline.

Roll No:- 44
Batch:- T13
Name :-Atharva Jadhav

ADVANCE DEVOPS ASSIGNMENT-5

AIM: To understand the Kubernetes Cluster Architecture.

LO MAPPED: LO1 , LO3

THEORY:

Q.1 What are the various Kubernetes services running on nodes? Describe the role of each service.

In a Kubernetes cluster, there are several essential services that run on nodes. These services are critical for the proper functioning of the cluster. Below, I'll describe the role of each service in detail:

kubelet:
The kubelet is responsible for managing containers on a node. It ensures that the containers in a Pod are running and healthy. It communicates with the control plane to receive Pod specifications and takes actions to make sure the containers match the desired state. For example, if a Pod specification indicates that it should run three containers, the kubelet ensures that those containers are up and running. If a container fails, the kubelet restarts it.
Example: Let's say you have a Pod with three containers, and one of them crashes due to a software issue. The kubelet will detect the failure and restart the failed container to maintain the desired state.

kube-proxy:
Kube-proxy is responsible for managing network connectivity to and from Pods. It maintains network rules on the host to enable communication between Pods and external networks. It sets up routes, handles load balancing, and ensures that network traffic is properly directed to the correct Pod.

Example: If you have a service in your cluster that needs to load balance traffic to a set of Pods, kube-proxy manages this load balancing by configuring network rules and routes, directing traffic to the appropriate Pods.

Container Runtime:
The container runtime is responsible for running containers within Pods. Kubernetes supports various container runtimes, such as Docker, containerd, and CRI-O. These runtimes are responsible for pulling container images, creating containers, and managing their lifecycle.
Example: If you define a Pod that runs a Docker container with a specific image, the container runtime (e.g., Docker) pulls the image from a container registry and runs the container as specified.

cAdvisor (Container Advisor): cAdvisor is responsible for collecting and exposing resource usage and performance data for containers. It provides valuable information about CPU, memory, network, and disk usage of running containers.
Example: You can use cAdvisor to monitor the resource consumption of your containers. For instance, it can help you identify a container that is consuming an unusually high amount of CPU or memory, indicating a potential performance issue.

Node Problem Detector (Optional):
The Node Problem Detector is responsible for detecting and reporting hardware and system failures on the node. It helps in identifying and isolating issues with nodes, such as hardware errors, kernel panics, or out-of-memory conditions.
Example: If a node experiences a hardware issue, such as a failing disk drive, the Node Problem Detector can detect this problem and report it, allowing administrators to take action and potentially drain the node to prevent further issues.

Device Plugins (Optional):
Device plugins are used to expose and manage specialized hardware resources on the node, such as GPUs, FPGAs, or hardware accelerators. They enable Pods to use these resources when required.

Example: If you have GPUs on your nodes and want to run machine learning workloads that require GPU acceleration, you can use a GPU device plugin to expose these GPUs to your Pods. Pods that need GPU resources can request them in their specifications.

OS Services (e.g., SSH, NTP):
These services, including SSH for remote access and NTP for time synchronization, are essential for maintaining the health and reliability of the node. SSH provides administrative access for troubleshooting and maintenance, while NTP ensures the node's clock is synchronized with the cluster, preventing time-related issues.
Example: You can use SSH to log in to a node for troubleshooting or updates. NTP ensures that all nodes in the cluster have synchronized clocks, which is crucial for maintaining consistency in distributed systems.

Kubelet Container:
The kubelet itself runs in its own container on the node. It is responsible for interacting with the container runtime, managing container logs, and performing garbage collection to reclaim disk space from unused container images.
Example: If you examine a running node, you'll find the kubelet running in its own container. It manages container-related tasks on the node, such as cleaning up old container images to free up storage space.

These roles collectively ensure the smooth operation of a Kubernetes node and the containers within it, making it possible to run and manage containerized applications in a distributed environment.

Q.2 What is Pod Disruption Budget (PDB)?

A **Pod Disruption Budget (PDB**) is a Kubernetes resource that allows you to control the disruption or eviction of Pods during voluntary disruptions (e.g., maintenance) and involuntary disruptions (e.g., hardware failures). PDBs define the minimum availability requirements for Pods in a set of related Pods, such as those belonging to a Deployment or StatefulSet. They ensure that a certain number of Pods are available at all times, helping to maintain the stability and availability of your applications in a Kubernetes cluster.

Here's a detailed explanation of PDBs and an example to illustrate their use:

Components of a Pod Disruption Budget (PDB):

minAvailable: This field specifies the minimum number of Pods that must be kept running in the group (e.g., a Deployment or StatefulSet). This ensures that a minimum number of replicas remain available during disruptions.

maxUnavailable: This field specifies the maximum number of Pods that can be unavailable during disruptions. It's complementary to minAvailable. You can choose to define one or the other, but not both. It provides a way to limit the maximum unavailability of Pods.

selector: PDBs are associated with Pods using label selectors. The selector is used to match Pods in the group that the PDB applies to.

Use Cases for PDBs:

Rolling Updates: When performing rolling updates of applications using Deployments or StatefulSets, PDBs can be used to ensure that a certain number of Pods remain available during the update process, preventing unintended disruptions to the application.

Node Drains: When nodes need maintenance or are being drained, PDBs can prevent the simultaneous eviction of too many Pods, ensuring that the application maintains its desired level of availability.

High Availability: PDBs can be used to enforce high availability requirements for critical components of an application. For example, a database cluster may require a certain number of replicas to be available at all times to prevent data loss.

Example of a Pod Disruption Budget:

Let's say you have a Deployment managing a web application with a replica count of 5. You want to ensure that at least 3 replicas of the web application are available at all times during updates or node maintenance. Here's how you would define a PDB for this use case:

```
apiVersion: policy/v1beta1 kind:
PodDisruptionBudget metadata:
  name: web-app-pdb
spec:
minAvailable: 3
selector:
matchLabels:
app: web-app
```

In this example: minAvailable: 3 specifies that at least 3 replicas of Pods with the label app: webapp must remain available during disruptions. selector specifies that this PDB applies to Pods with the label app: web-app. With this PDB in place, if you perform a rolling update of the Deployment or if nodes need maintenance, Kubernetes will ensure that at least 3 Pods of the webapp Deployment remain operational during these events, thereby meeting the specified availability requirements.

PDBs are a powerful tool for maintaining application stability and availability in Kubernetes clusters, particularly when handling planned or unplanned disruptions to your workloads.

Q.3 What is the role of Load Balance in Kubernetes?

In Kubernetes, a Load Balancer is a critical component that helps distribute network traffic evenly across a set of Pods or Services. Load balancing is essential for ensuring high availability, scaling applications, and maintaining stable network connections. Here's a detailed explanation of the role of Load Balancers in Kubernetes, along with an example:

Role of Load Balancers in Kubernetes:

Distributing Traffic: Load Balancers evenly distribute incoming network traffic across multiple Pods or Services. This ensures that no single Pod or Service becomes overwhelmed, improving the responsiveness and availability of the application.

High Availability: Load Balancers are typically configured with health checks to monitor the status of Pods or Services. If a Pod or Service becomes unhealthy or unresponsive, the Load Balancer can automatically route traffic away from it, ensuring the application remains available.

Scaling: As your application grows and you need to add more instances (Pods) to handle increased traffic, Load Balancers can seamlessly adapt to include these new instances in the traffic distribution. This makes it easier to scale your application horizontally.

Session Persistence: Some Load Balancers support session persistence or sticky sessions, which ensure that requests from the same client are consistently routed to the same backend Pod. This is useful for stateful applications that rely on session data.

External Access: Load Balancers often act as a point of entry for external traffic into your cluster. They can route traffic to the appropriate Services within the cluster based on the configuration and rules you define.

Types of Load Balancers in Kubernetes:
Service Type: LoadBalancer: Kubernetes provides a native LoadBalancer Service type. When you define a Service of type LoadBalancer, the Kubernetes cluster provisions an external Load Balancer, typically provided by the cloud provider, to distribute traffic to the Service. For example:

```
apiVersion: v1 kind:
Service metadata:
  name: my-service spec:
  selector:
    app: my-app
ports:
   - protocol: TCP
port: 80
```

targetPort: 9376
type: LoadBalancer

Ingress Controllers: Ingress controllers, such as Nginx Ingress or HAProxy, are used to manage external access to Services within a cluster. Ingress controllers provide more advanced routing and traffic management capabilities than the basic LoadBalancer Service type. **Example of Load Balancer in Kubernetes:**
Let's say you have a web application deployed as a set of Pods and you want to make it accessible to external users. You can create a LoadBalancer Service to achieve this:

```
apiVersion: v1 kind:
Service metadata:
name: web-service
spec:
  selector:
    app: web-app
ports:
   - protocol: TCP
port: 80
targetPort: 8080
type: LoadBalancer
```

In this example:
metadata.name is the name of the Service. spec.selector specifies the Pods to which the traffic should be load balanced. spec.ports define the ports to which the Load Balancer should forward traffic.
type: LoadBalancer indicates that you want to provision an external Load Balancer for this Service.
Once this configuration is applied, Kubernetes (or the cloud provider) will provision an external Load Balancer and assign it an IP address. Users can then access your web application using this IP address, and the Load Balancer will distribute incoming requests across the Pods running your web application. Load Balancers are a fundamental component for ensuring the availability, scalability, and external accessibility of applications in a Kubernetes cluster. They play a

crucial role in maintaining a stable and responsive environment for your applications.

CONCLUSION: Hence,In this assignment we learned what is the Kubernetes Cluster Architecture.

Terraform
Lab-6

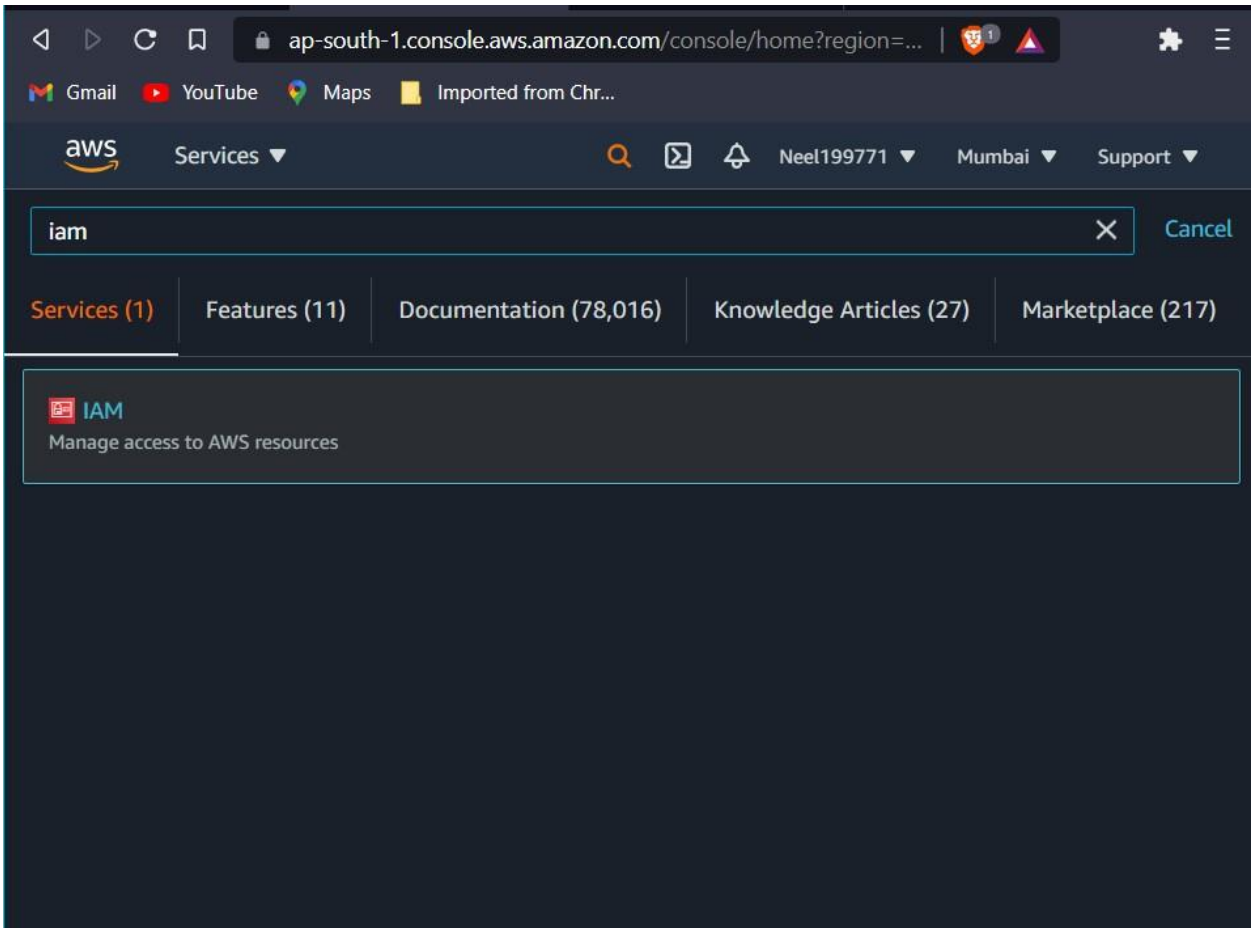
Aim:- To understand terraform lifecycle and to Build, change, and destroy AWS infrastructure using Terraform.

Theory:-

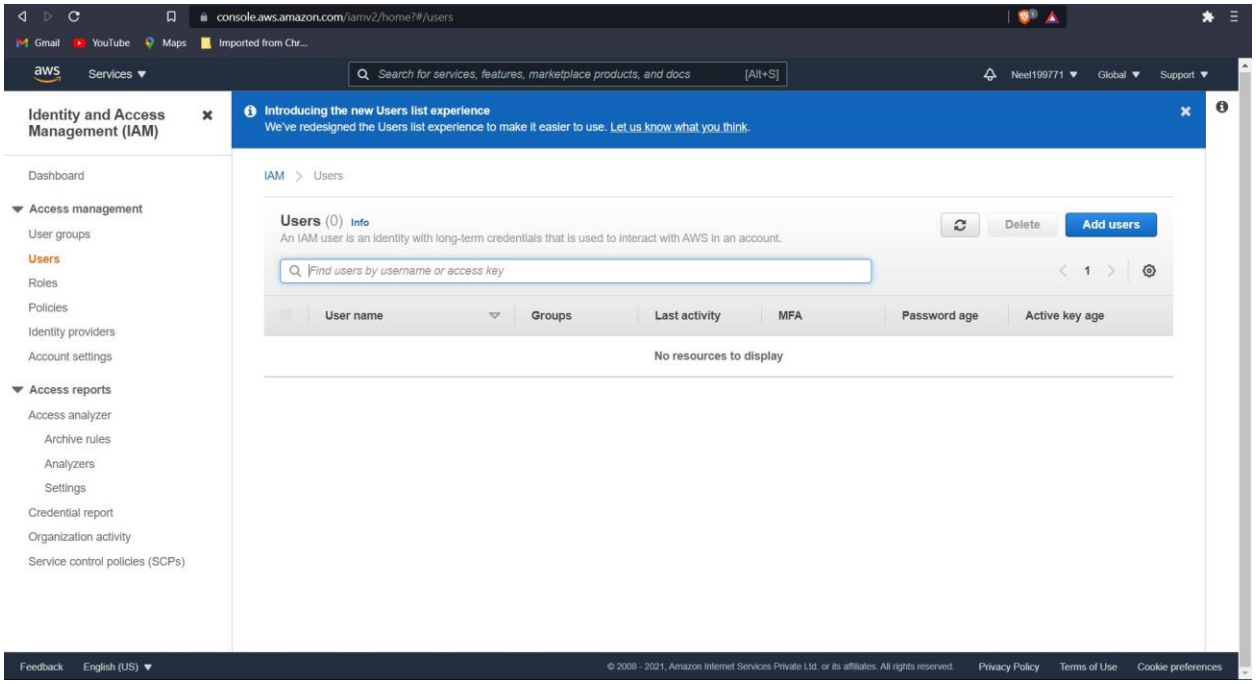
Terraform is an infrastructure as code (IaC) tool that allows you to build, change, and version infrastructure safely and efficiently.

Steps for Implementation:-

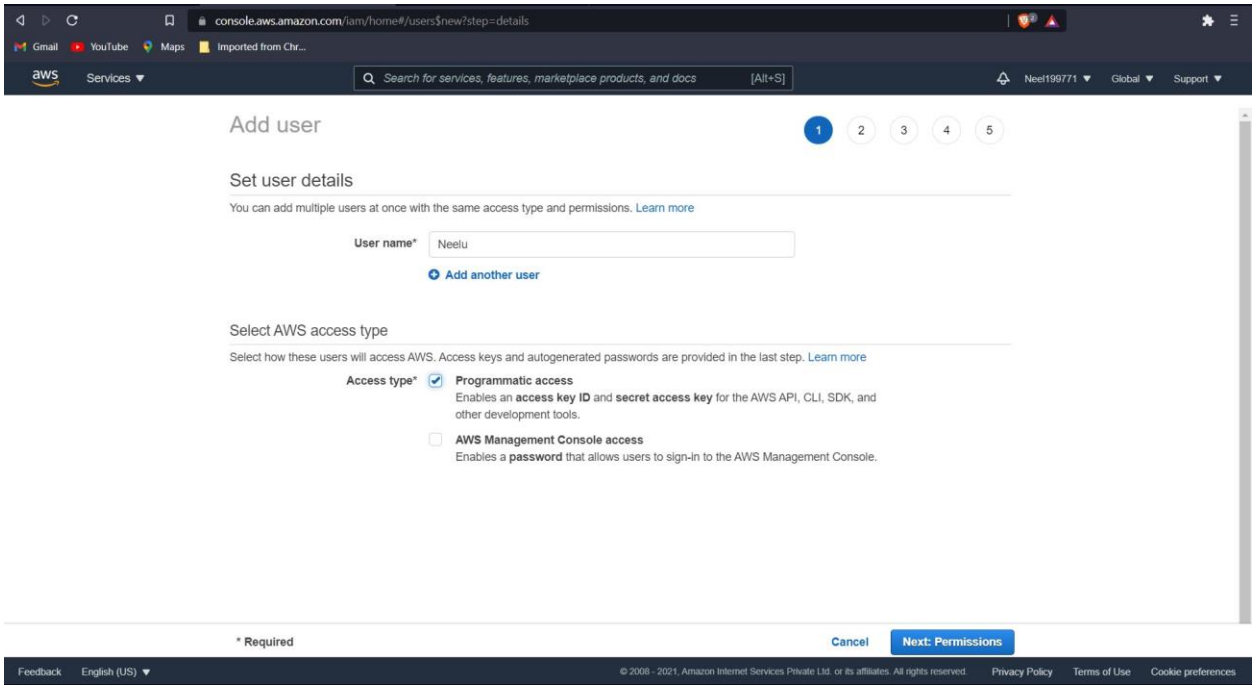
1- Open And Login to your AWS console and open IAM.



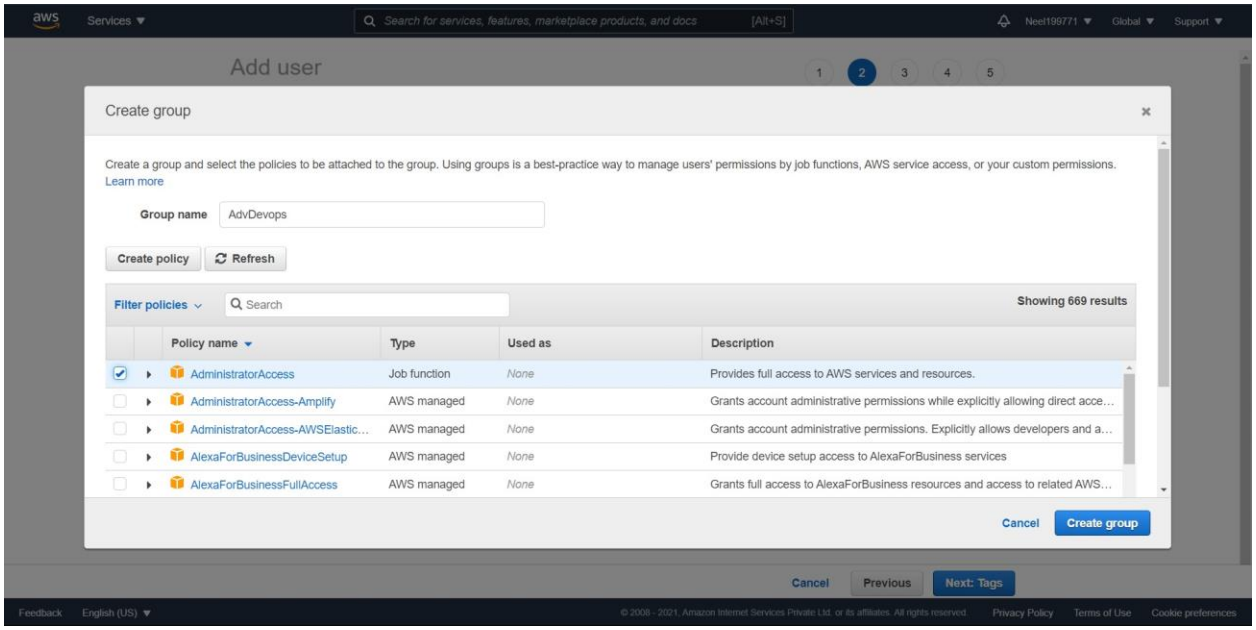
2- Click on Add Users



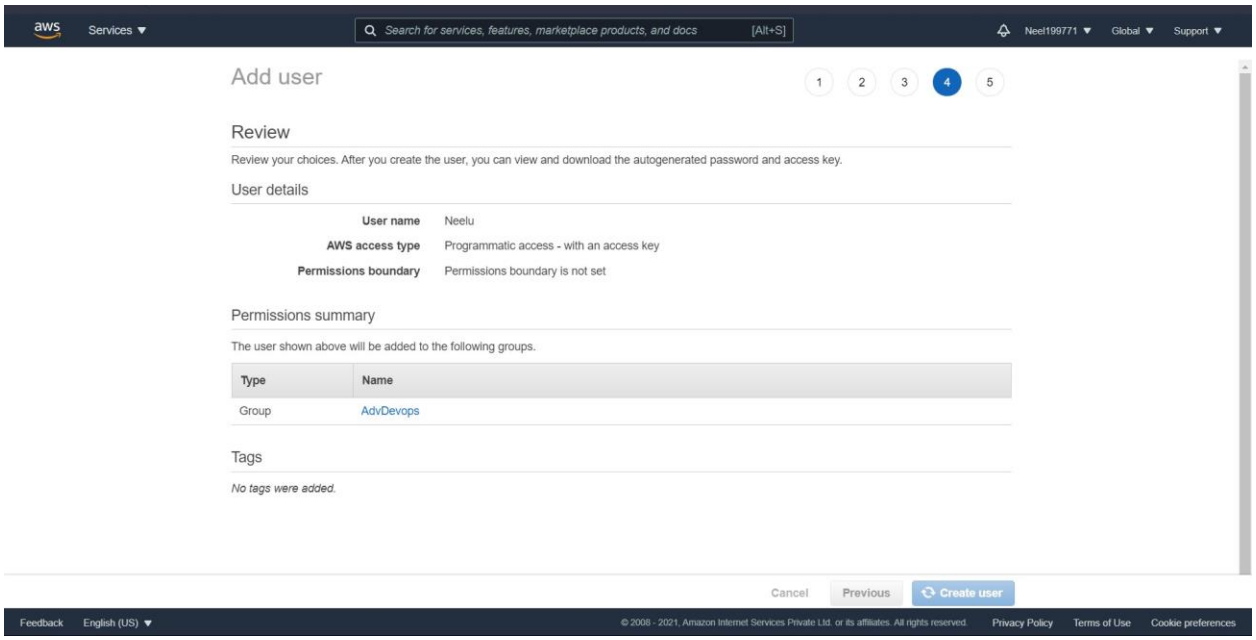
3- Give any name for username and check the programmatic access field shown below



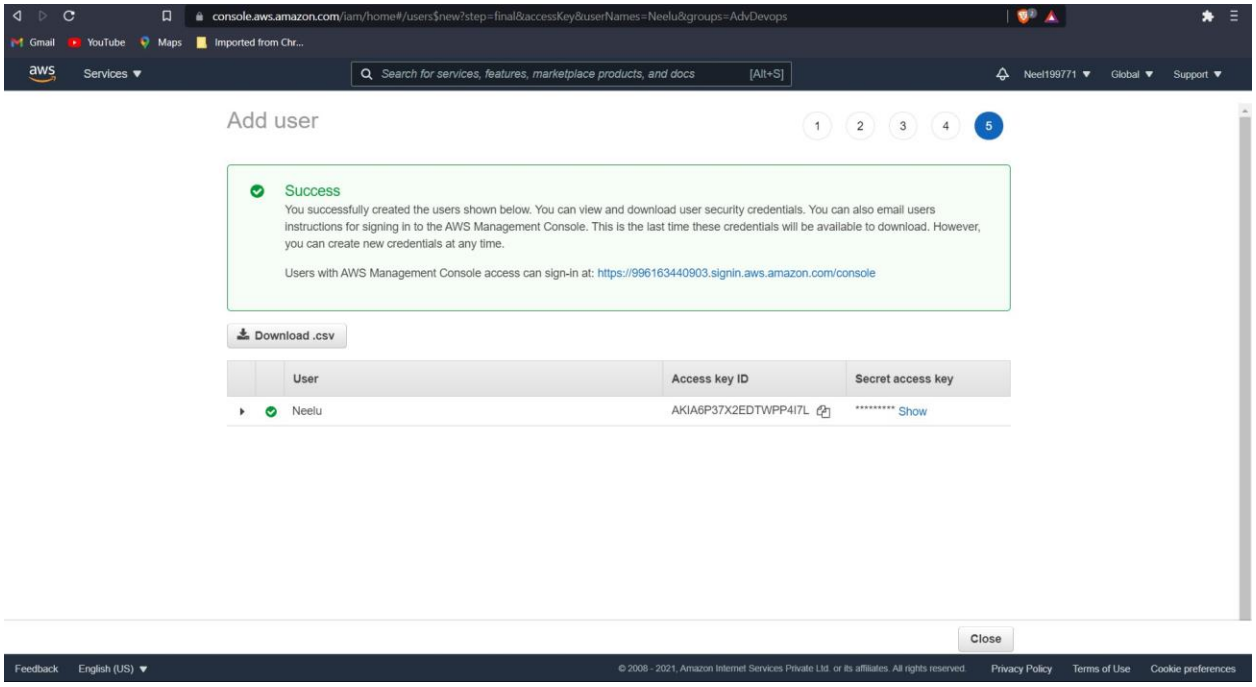
4- Add group name and add AdministratorAccess policy by clicking checkbox



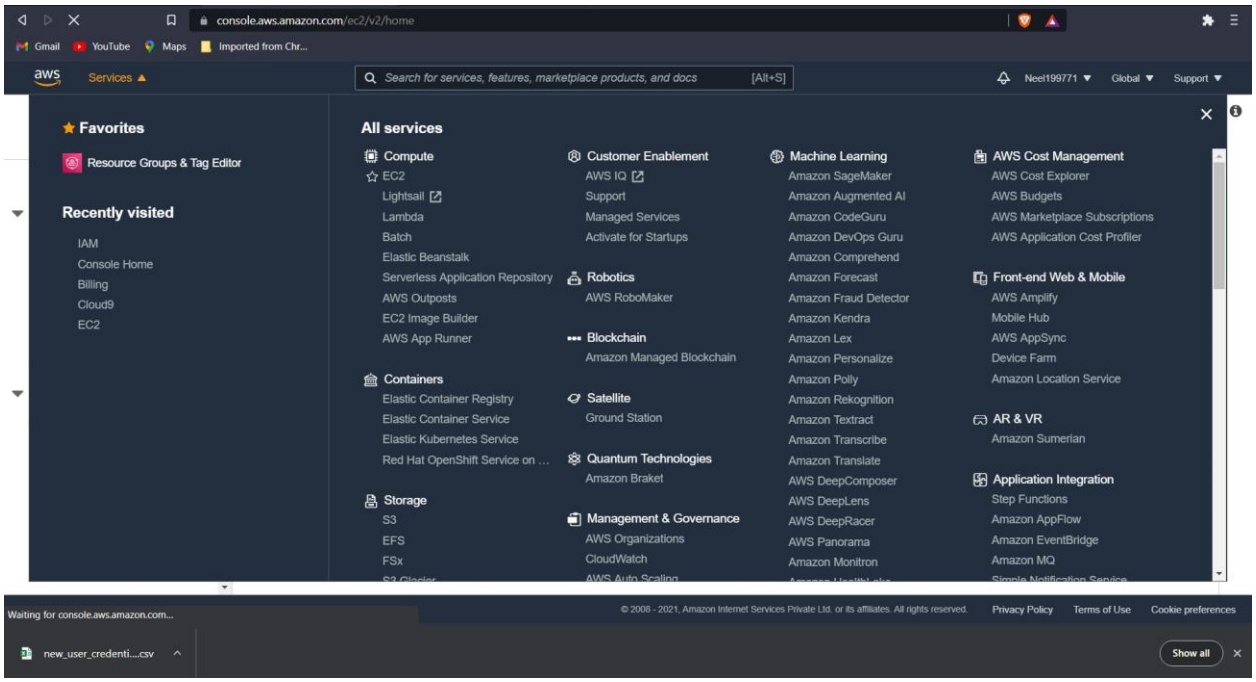
5- Don’t add tags



6- Now Download .csv file



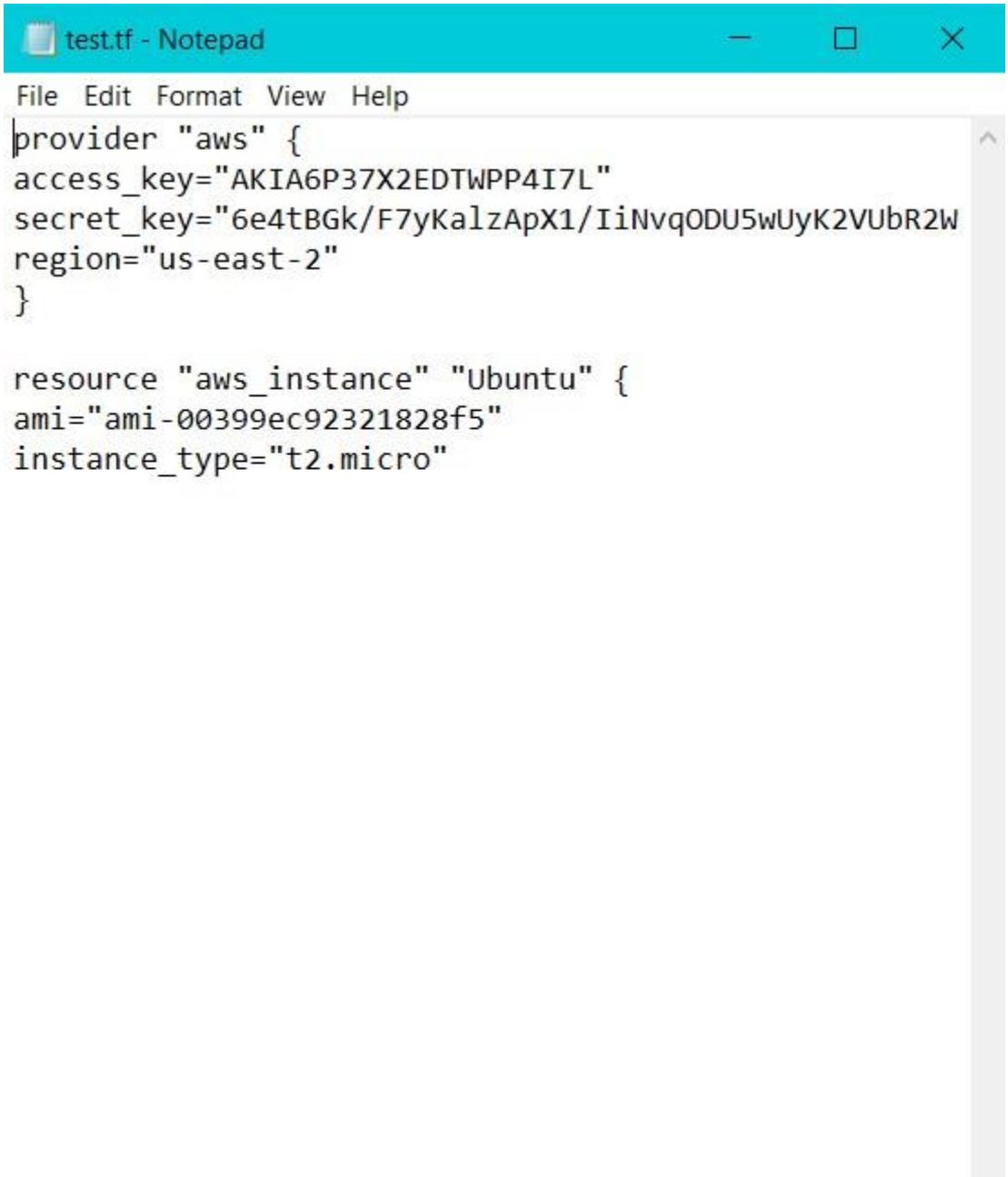
7- Go to services and Ec2



Created a folder Name Terraform Scripts in the C drive where the AdvDevops folder was created

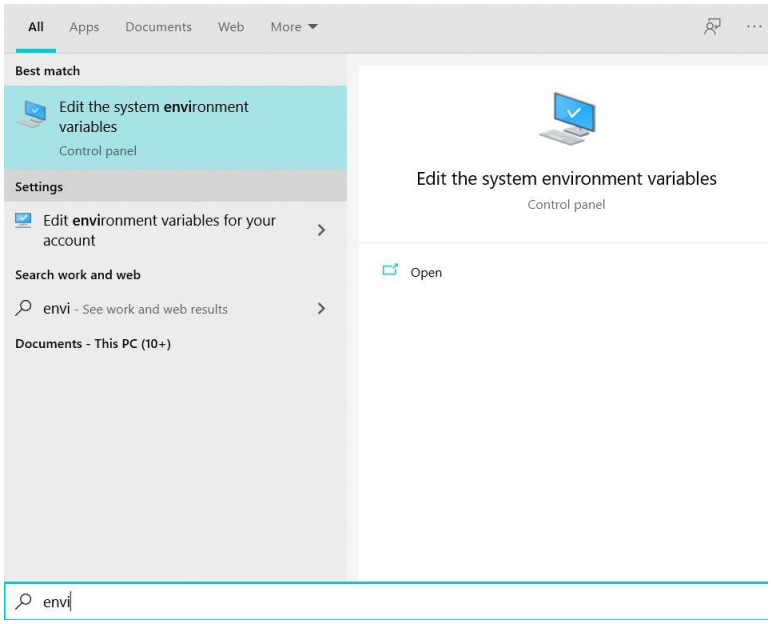
Now Go to note pad And Type the below Details properly But before It Just Change the ACCESS KEY AND SECRECT KEY TO THE ONE IN YOUR .csv File .

Set region same as below if you want MUMBAI as your region.



Now search EDIT THE SYSTEM ENVIRONMENT VARIABLES in your windows search.

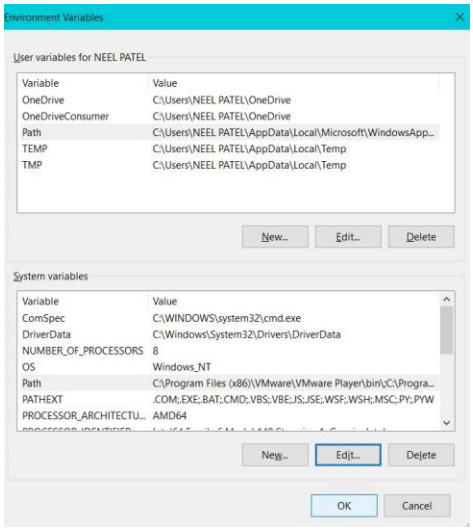
Open it



Now click on PATH OF USER VARIBLES, then click on Edit option

Now go to edit and then add new path C:\AdvDevOps

Repeat same procedure for system variables.



Now Open Command Prompt and then pate the path of Terraform script

Eg. CD C:\Terraform Script as shown below

Now type Terraform Init command

```
C:\Teraform Script>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v3.52.0...
- Installed hashicorp/aws v3.52.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Then if there are no errors type Terraform Plan as shown below (type YES when command prompt ask)

```
C:\Teraform Script>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.Ubuntu will be created
  + resource "aws_instance" "Ubuntu" {
      + ami                                  = "ami-0c1a7f89451184c8b"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
      + cpu_threads_per_core                 = (known after apply)
      + disable_api_termination              = (known after apply)
      + ebs_optimized                        = (known after apply)
      + get_password_data                    = false
      + host_id                              = (known after apply)
```

Now Finally Type Terraform Apply

```
C:\Teraform Script>terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with t
following symbols:
  + create

Terraform will perform the following actions:

  # aws_instance.Ubuntu will be created
  + resource "aws_instance" "Ubuntu" {
      + ami                                  = "ami-0c1a7f89451184c8b"
      + arn                                  = (known after apply)
      + associate_public_ip_address          = (known after apply)
      + availability_zone                    = (known after apply)
      + cpu_core_count                       = (known after apply)
```

```
Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_instance.Ubuntu: Creating...
aws_instance.Ubuntu: Still creating... [10s elapsed]
aws_instance.Ubuntu: Still creating... [20s elapsed]
aws_instance.Ubuntu: Still creating... [30s elapsed]
aws_instance.Ubuntu: Creation complete after 31s [id=i-0eac948a456860494]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

Now go to EC2 and check that is an instance created by the name of UBUNTU and is it in running status or not If it is in Running Status then Come back to Command prompt And Terminate the Instance by

Typing - Terraform destroy

```
Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

aws_instance.Ubuntu: Destroying... [id=i-0eac948a456860494]
aws_instance.Ubuntu: Still destroying... [id=i-0eac948a456860494, 10s elapsed]
aws_instance.Ubuntu: Still destroying... [id=i-0eac948a456860494, 20s elapsed]
aws_instance.Ubuntu: Destruction complete after 30s

Destroy complete! Resources: 1 destroyed.

C:\Teraform Script>
```

Now go back to EC2 if the instance is terminated, if yes then logout of the Aws Console.

And close the command prompt!

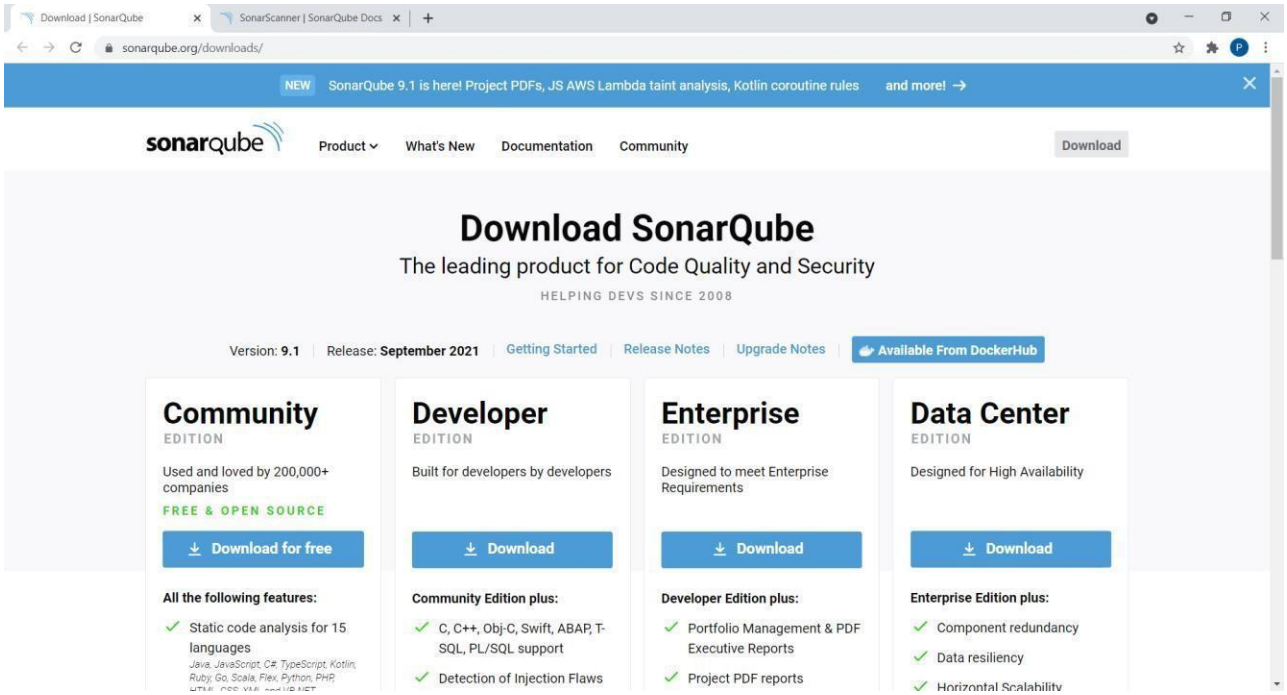
Conclusion:-

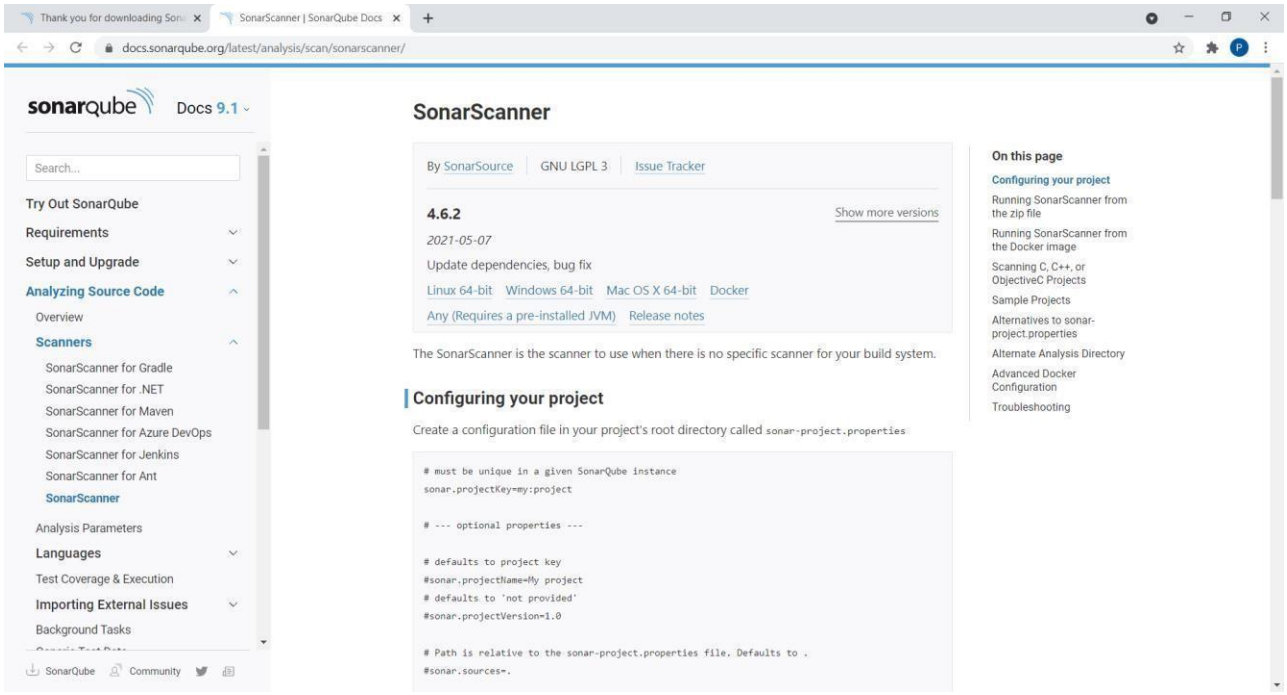
Terraform is a powerful Infrastructure as Code (IaC) tool that automates the provisioning, management, and destruction of AWS infrastructure. It can help you to save time, reduce errors, and improve the consistency of your infrastructure.

Advanced DevOps Experiment – 7

Aim: To perform static analysis using sonarQube

Download SonarQube and Sonar Scanner

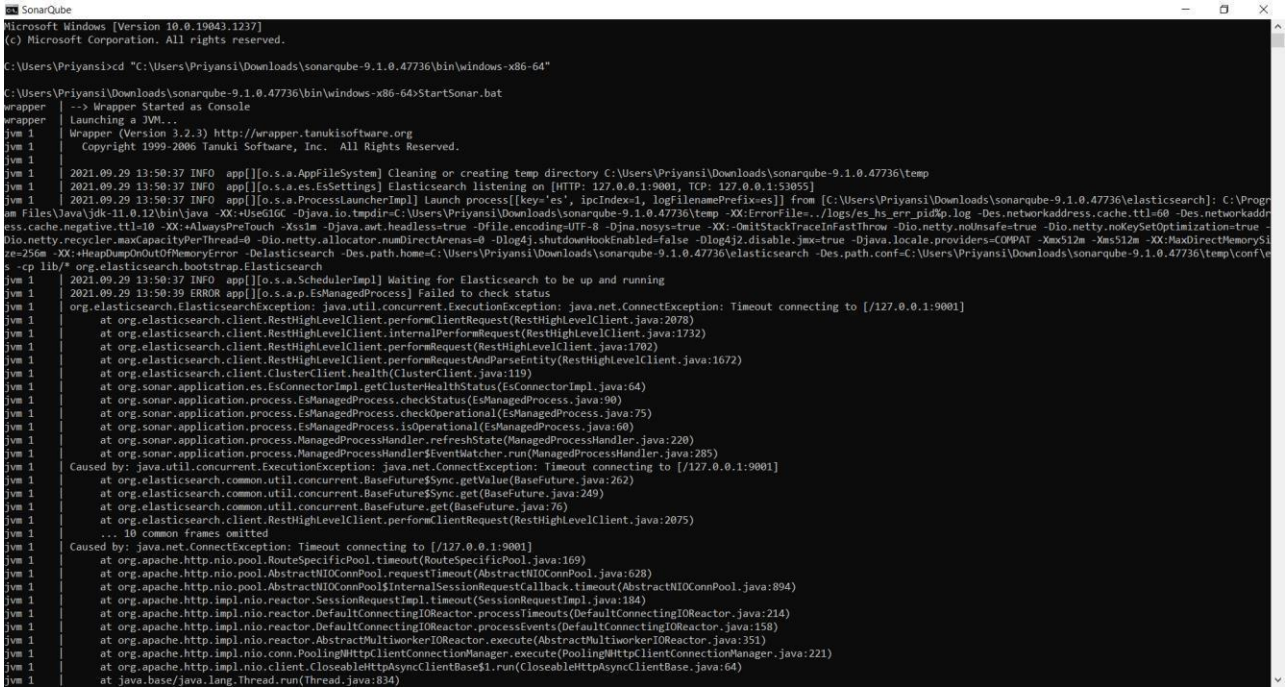


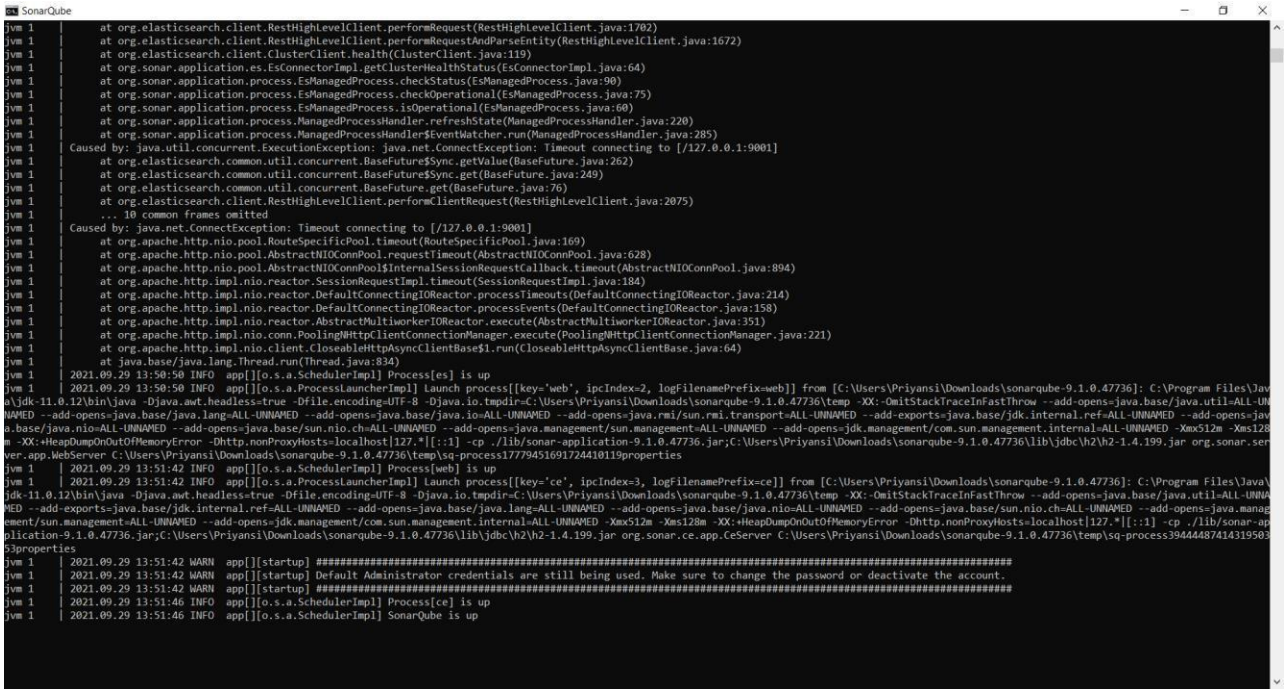


After downloading, set Environment Variables. Add “sonarqube-9.1.0.47736\bin” to Path.

Open command prompt. Run commands:

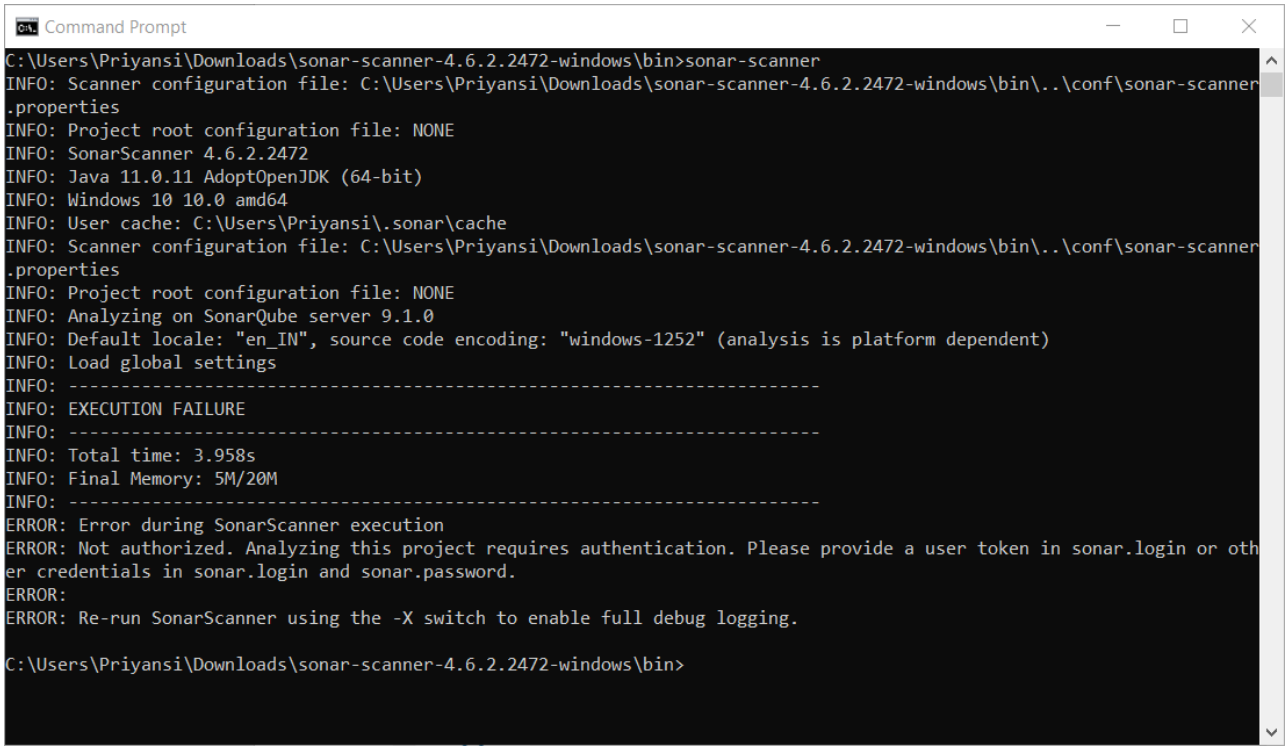
- cd “sonarqube-9.1.0.47736\bin\windows-x86-64”
- StartSonar.bat



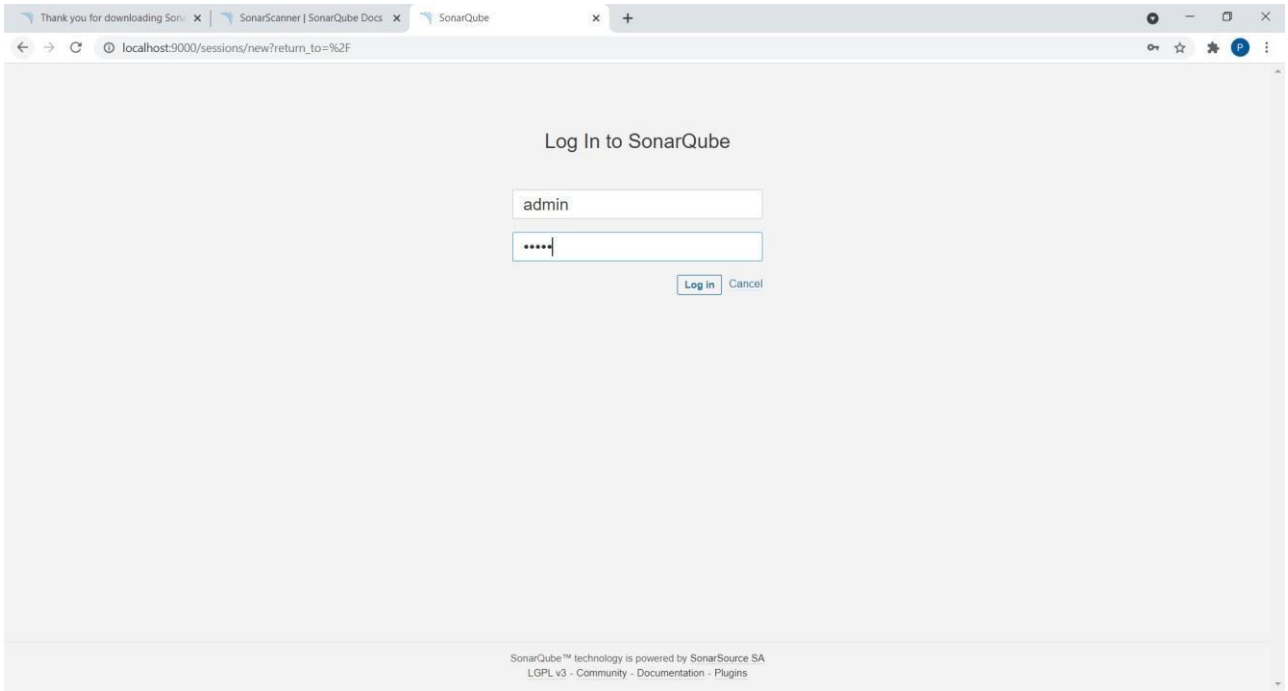


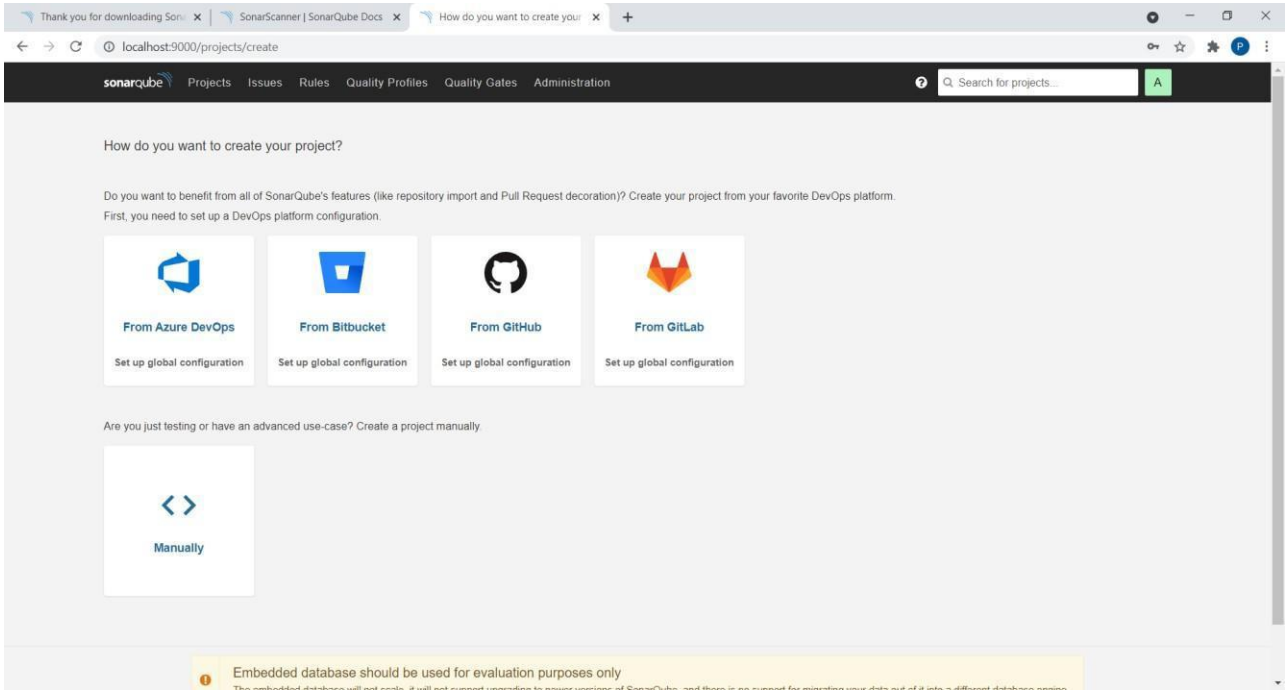
Open another command prompt. Run command:

- cd “sonar-scanner-4.6.2.2472-windows\bin”
- sonar-scanner

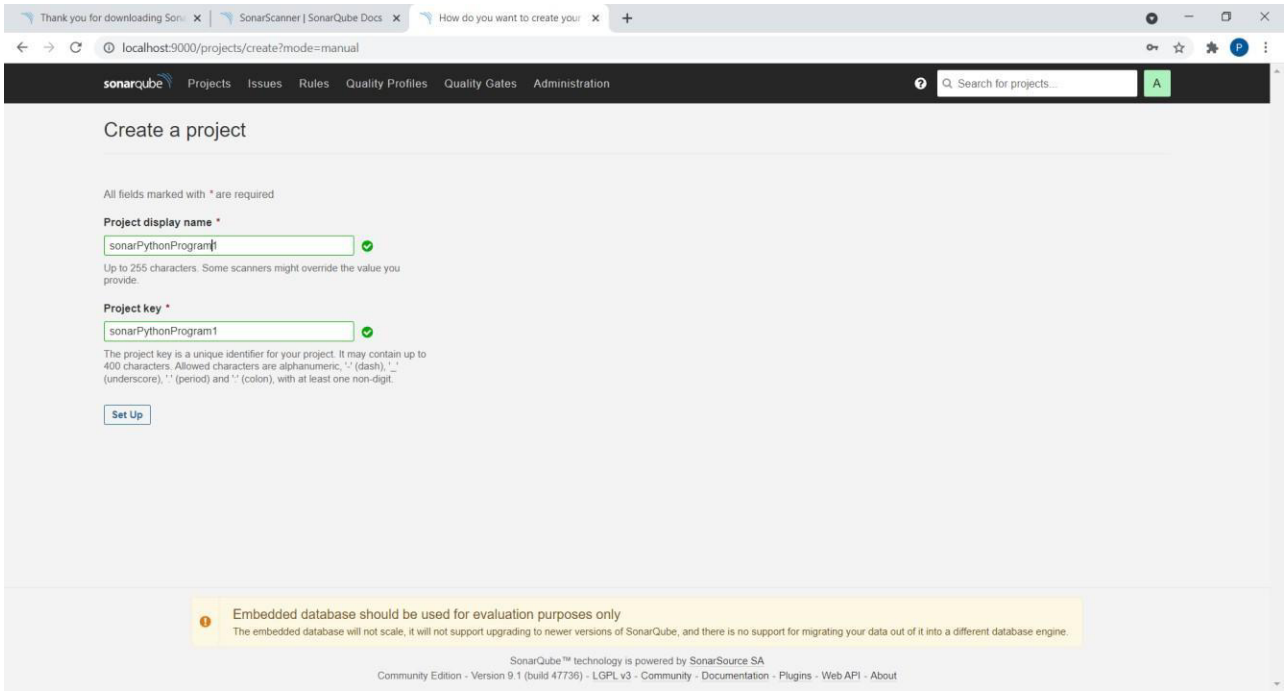


Server up and running on **localhost:9000**
Login using credentials as User: admin and Password: admin and Set a new password

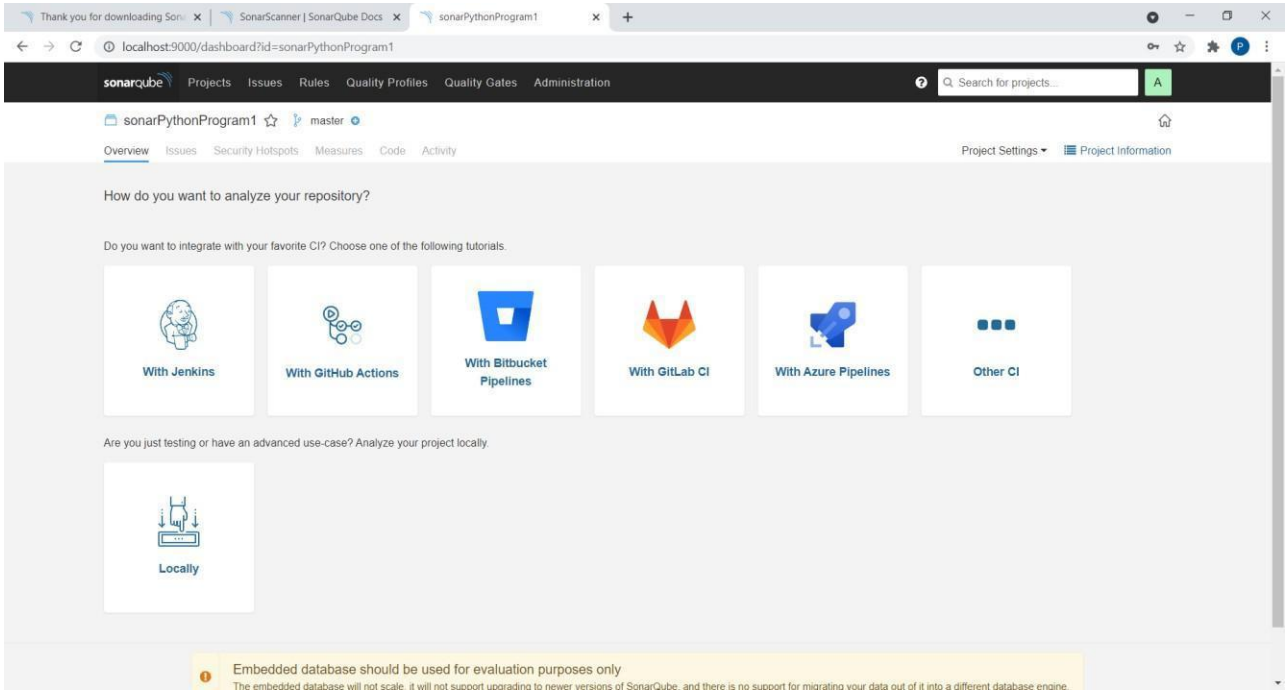




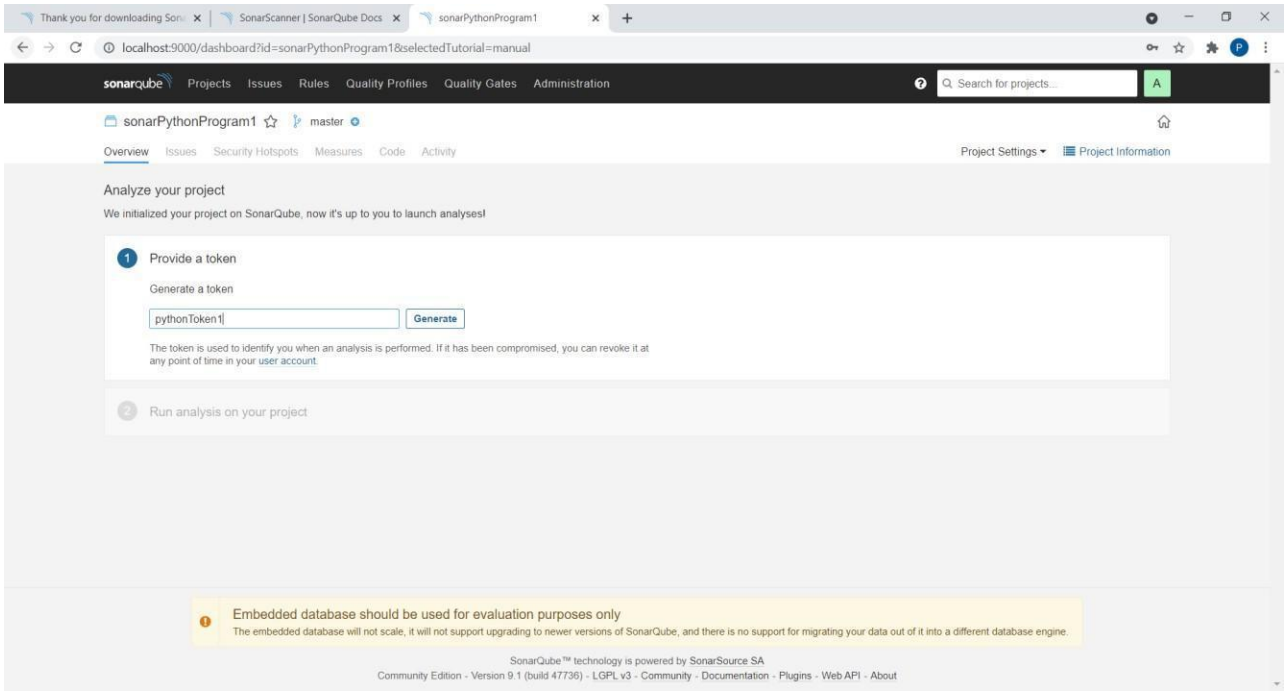
Click on Create a project **Manually.**



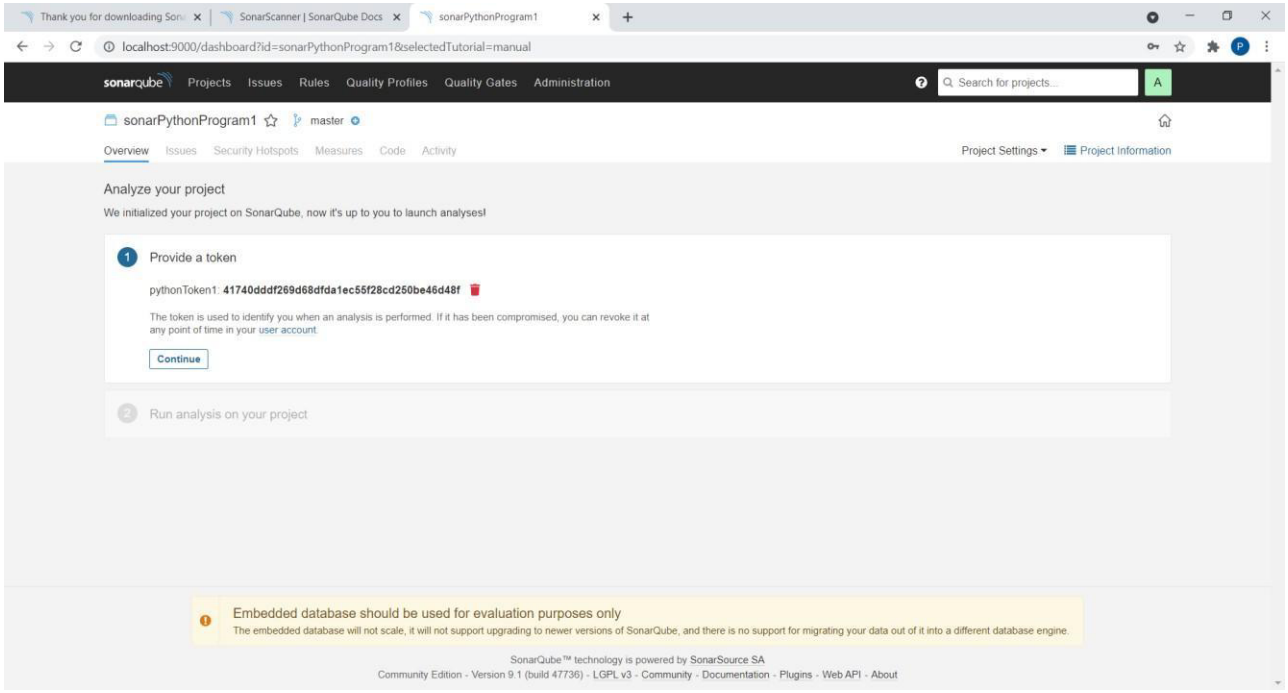
Give any Project display name.



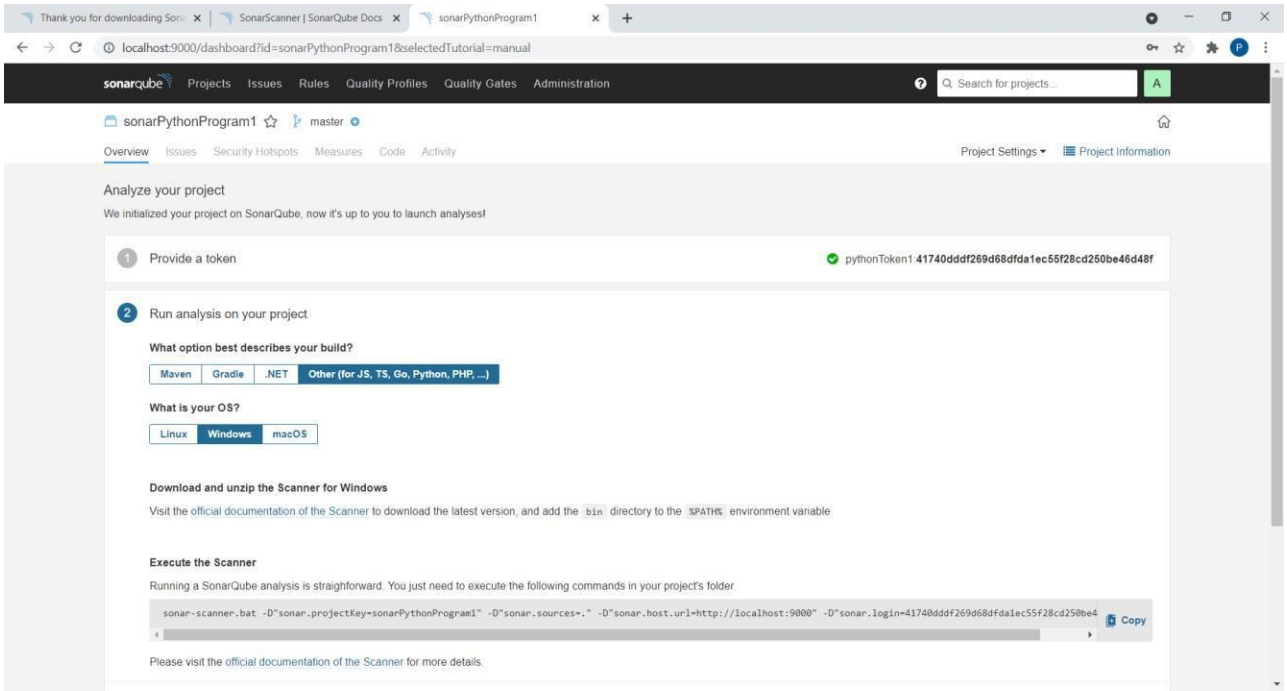
Click on **Locally.**



Give any name to token and click on **Generate.**



Click on **Continue.**

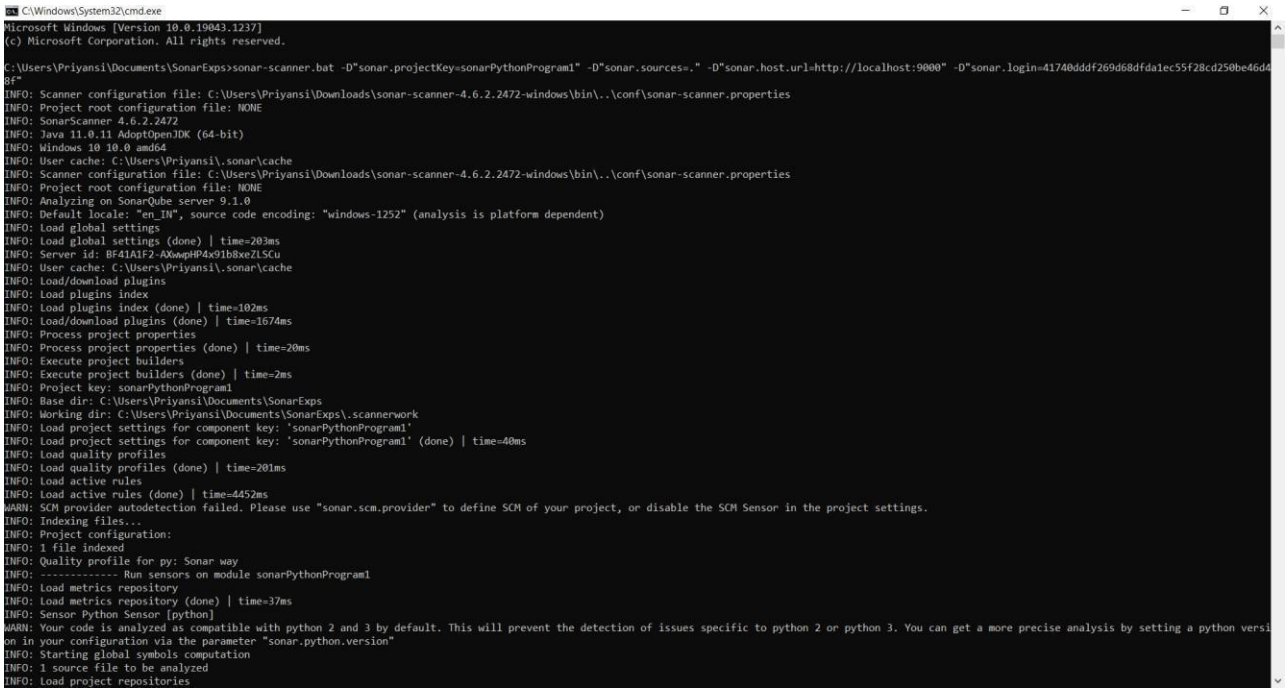


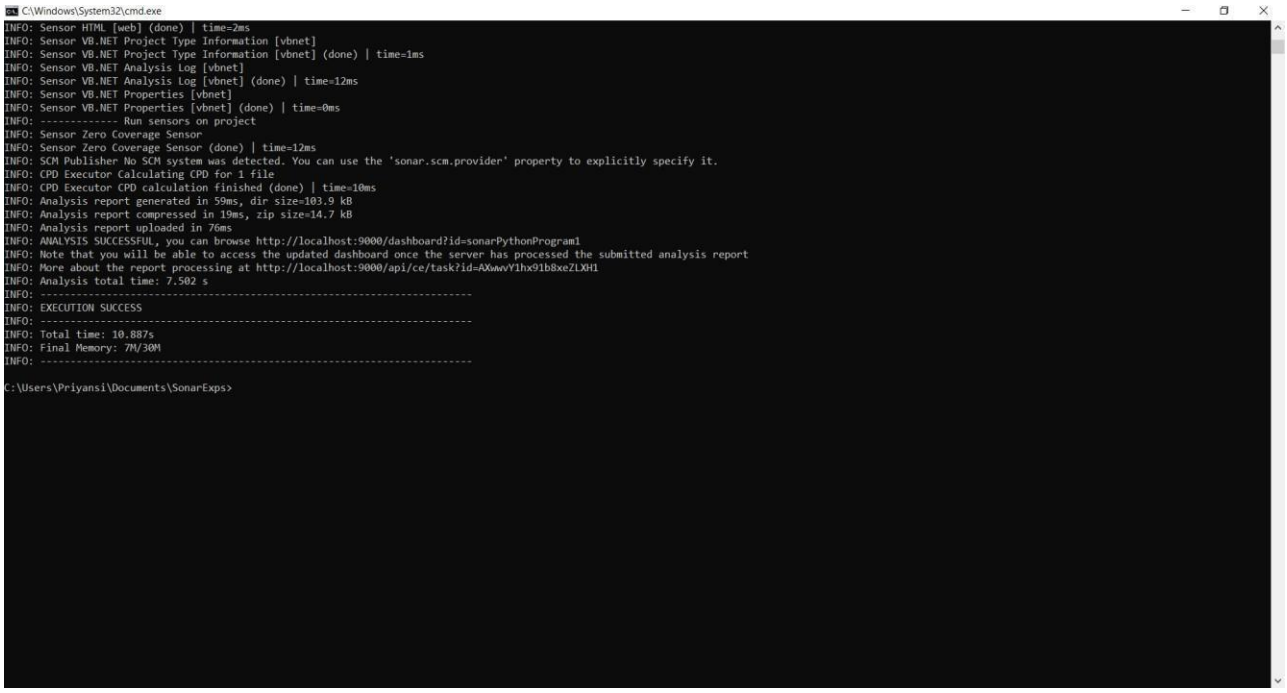
Save a Python program in a folder.

```
class Solution(object):
   def romanToInt(self, s):
```

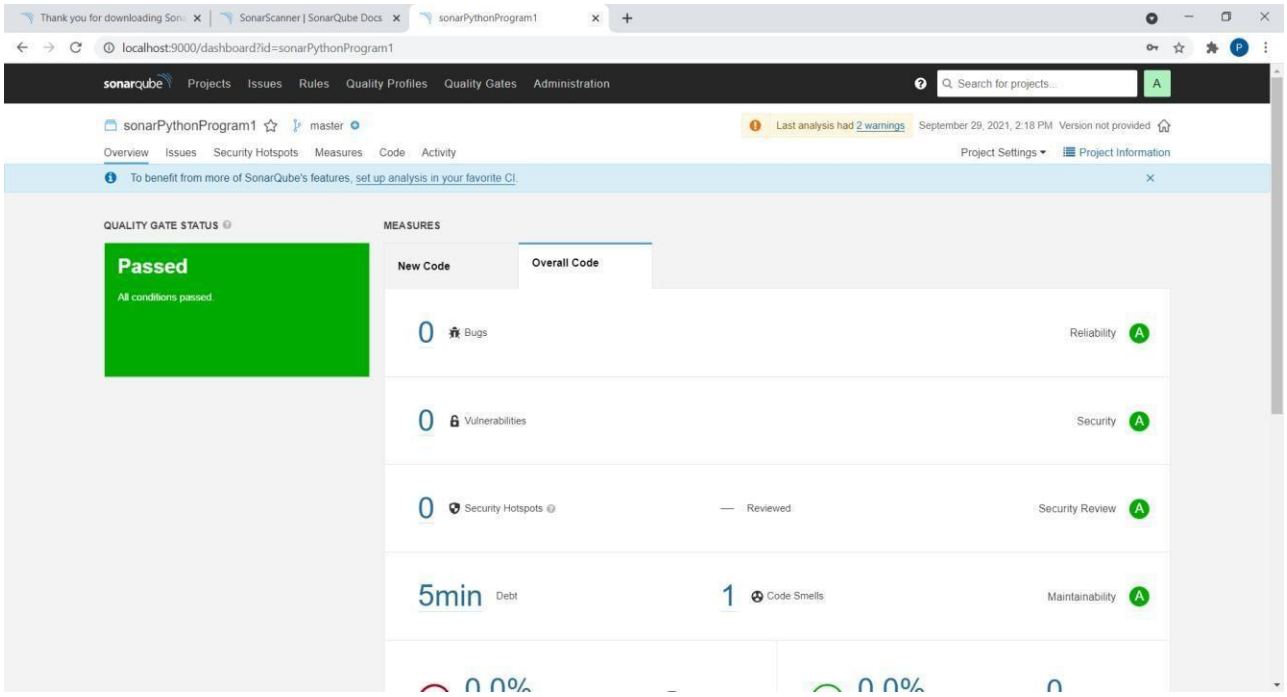
```
    roman =
{'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000,'IV':4,'IX':9,'XL':40,'XC':90,'CD':400,'CM':900}
    i = 0
    num = " "
    while i < len(s):
      if i+1<len(s) and s[i:i+2] in roman:
        num+=roman[s[i:i+2]]
        i+=2
      else:
        #print(i)
        num+=roman[s[i]]
        i+=1
    return num
ob1 = Solution()
print(ob1.romanToInt("III"))
print(ob1.romanToInt("CDXLIII"))
```

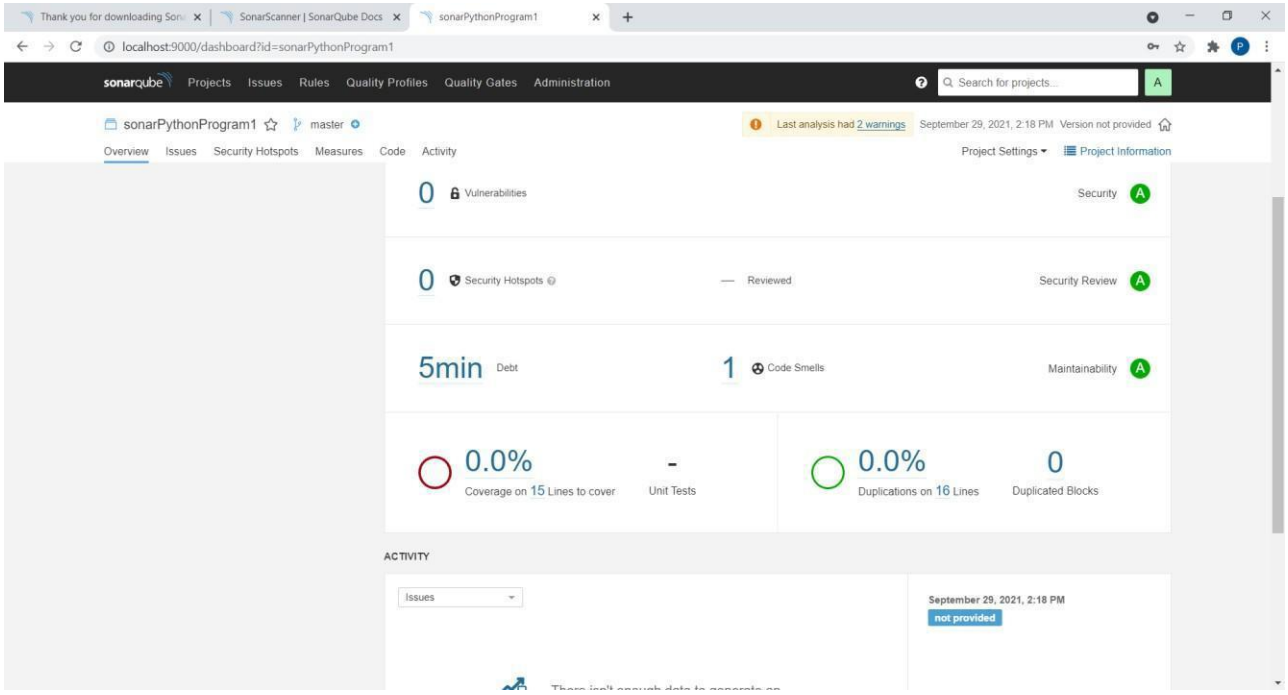
Open command prompt in this folder and Run program using copied command.
“sonar-scanner.bat -D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"

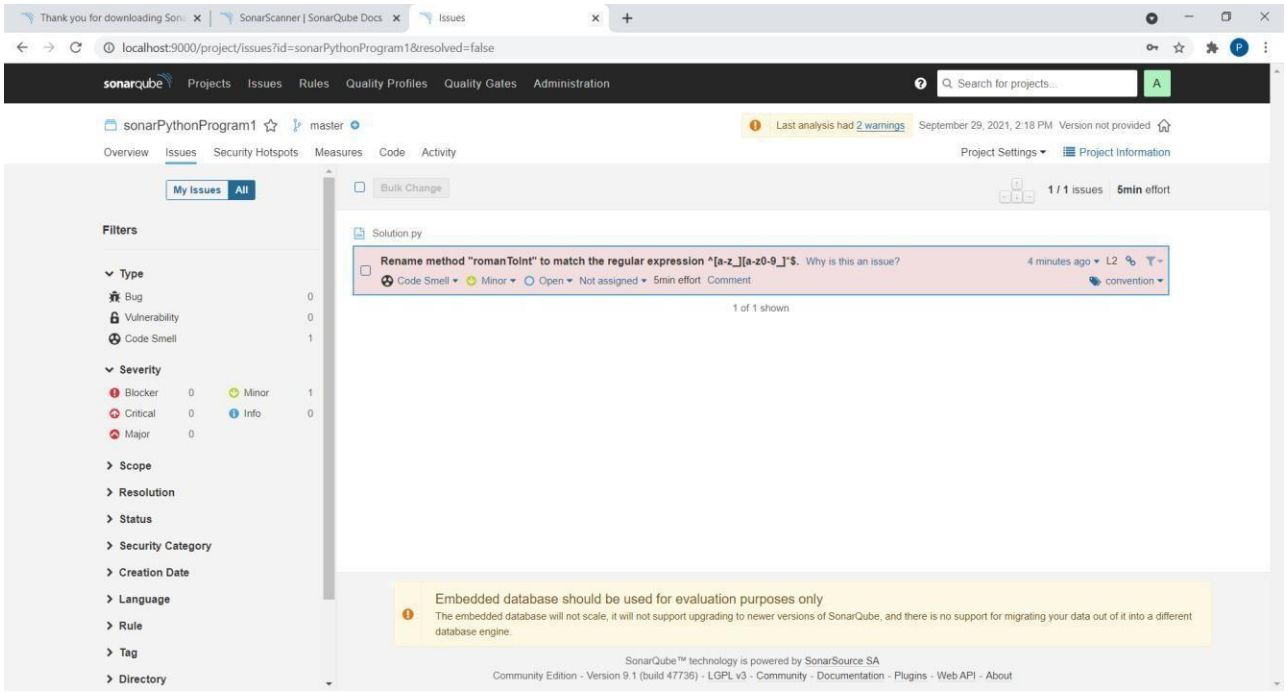




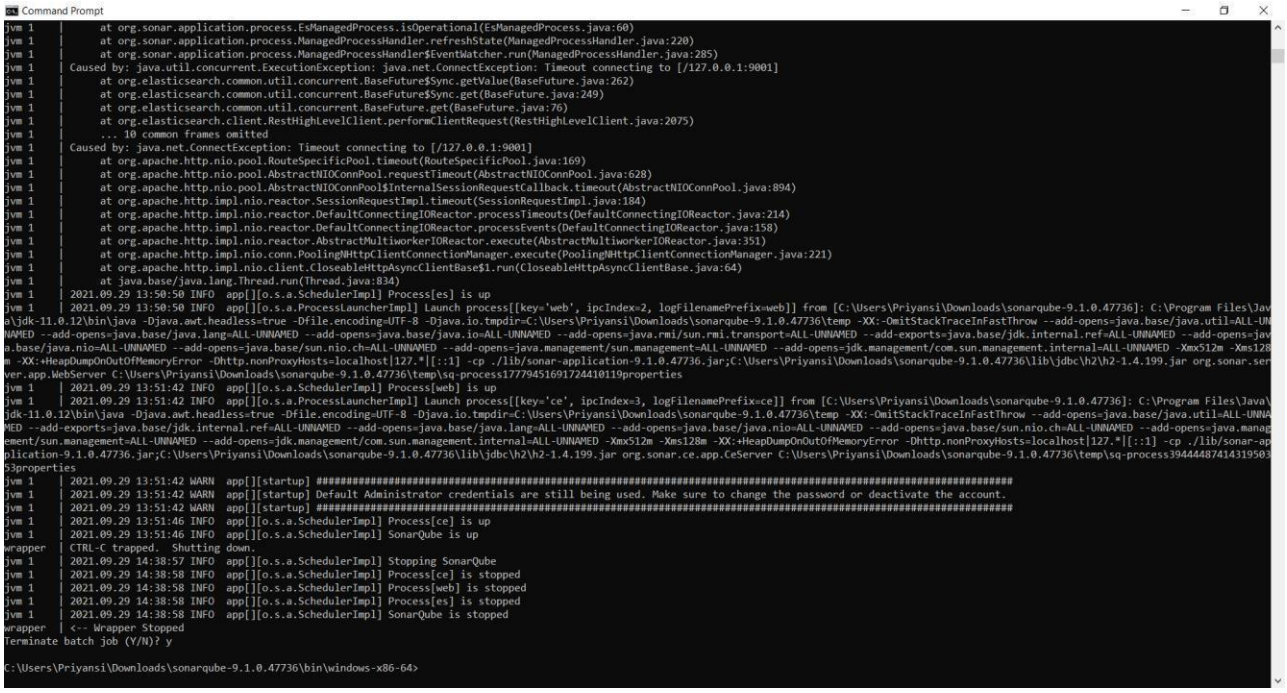
Given below is the inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.







Press “**Ctrl + C**” to stop the server.



CONCLUSION :-

We implemented static analysis using SonarCube.

Name :-Atharva Jadhav
Roll No:- 2105044
Batch:- T13
Date Of Performance :- 12/09/2023

Experiment 8

Aim:-

To understand continuous monitoring using Nagios

Theory:-

Nagios is an open-source monitoring system that provides monitoring of services, applications, and network resources. It is designed to alert system administrators about potential issues before they become critical problems. Nagios allows you to monitor the entire IT infrastructure, including servers, switches, applications, and services. It provides a comprehensive monitoring solution for both small and large organizations. Some key features and capabilities of Nagios include:

Monitoring Capabilities: Nagios can monitor a wide variety of network services including SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH, and many more.

Alerting and Notification: It provides alerting and notification functionalities to notify system administrators when something goes wrong. Nagios can send alerts via email, SMS, or other methods to ensure that the right people are notified in real-time.

Plugin Architecture: Nagios has a modular architecture that allows users to develop their plugins and addons to monitor specific devices and services that are not covered by default.

Customizable Dashboards and Reports: Nagios offers customizable dashboards and reporting capabilities that provide insights into the performance and health of the monitored resources.

Scalability and Flexibility: Nagios can scale to monitor complex, large-scale IT infrastructures. It is highly flexible and can be customized to meet specific monitoring and alerting requirements.

Extensibility: Nagios can be extended through various addons and plugins, allowing it to integrate with other tools and services, and enabling the monitoring of a wide range of devices and applications.

Historical Monitoring and Trend Analysis: Nagios can store historical data and provide trend analysis, allowing system administrators to identify patterns and plan for future infrastructure needs.

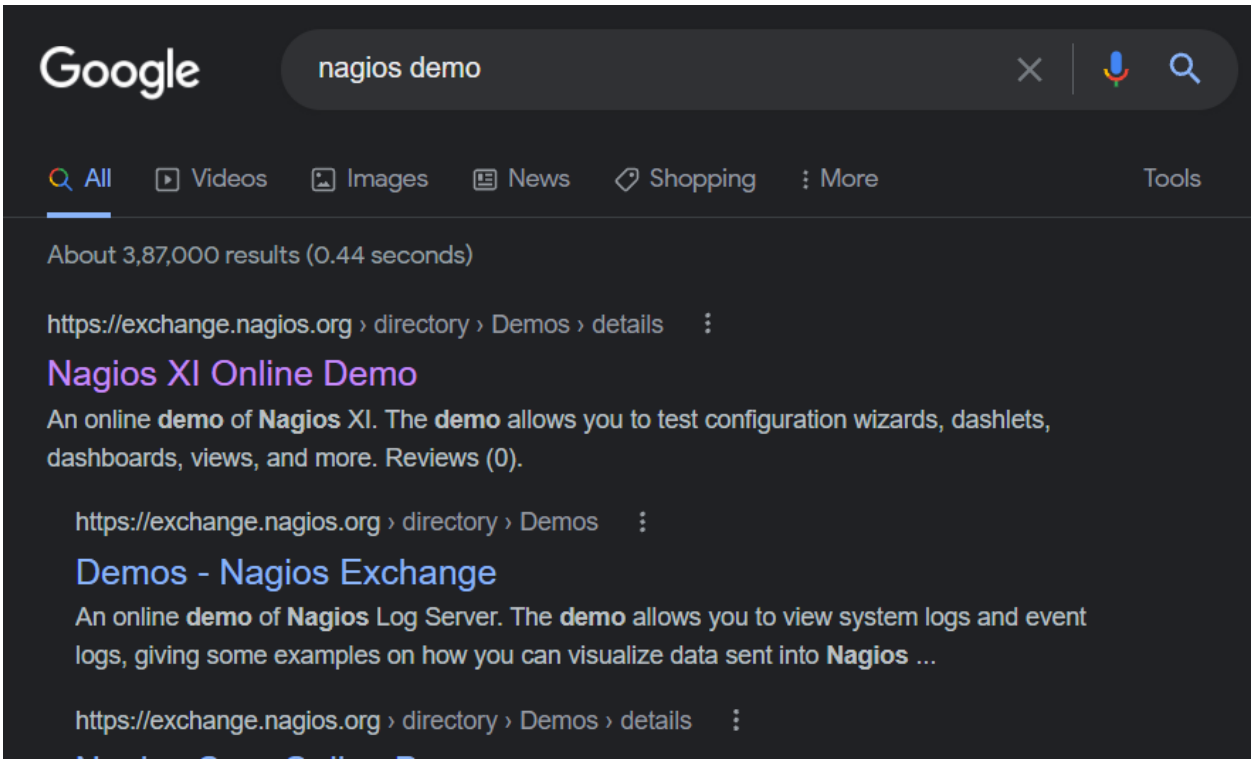
Community Support and Active Development: Being an open-source project, Nagios has a vibrant community that contributes to its development and support. This community-driven approach ensures that the software remains updated and robust.

Centralized Monitoring: Nagios provides a centralized view of the entire IT infrastructure, allowing administrators to have a comprehensive overview of the health and performance of all monitored resources from a single location.

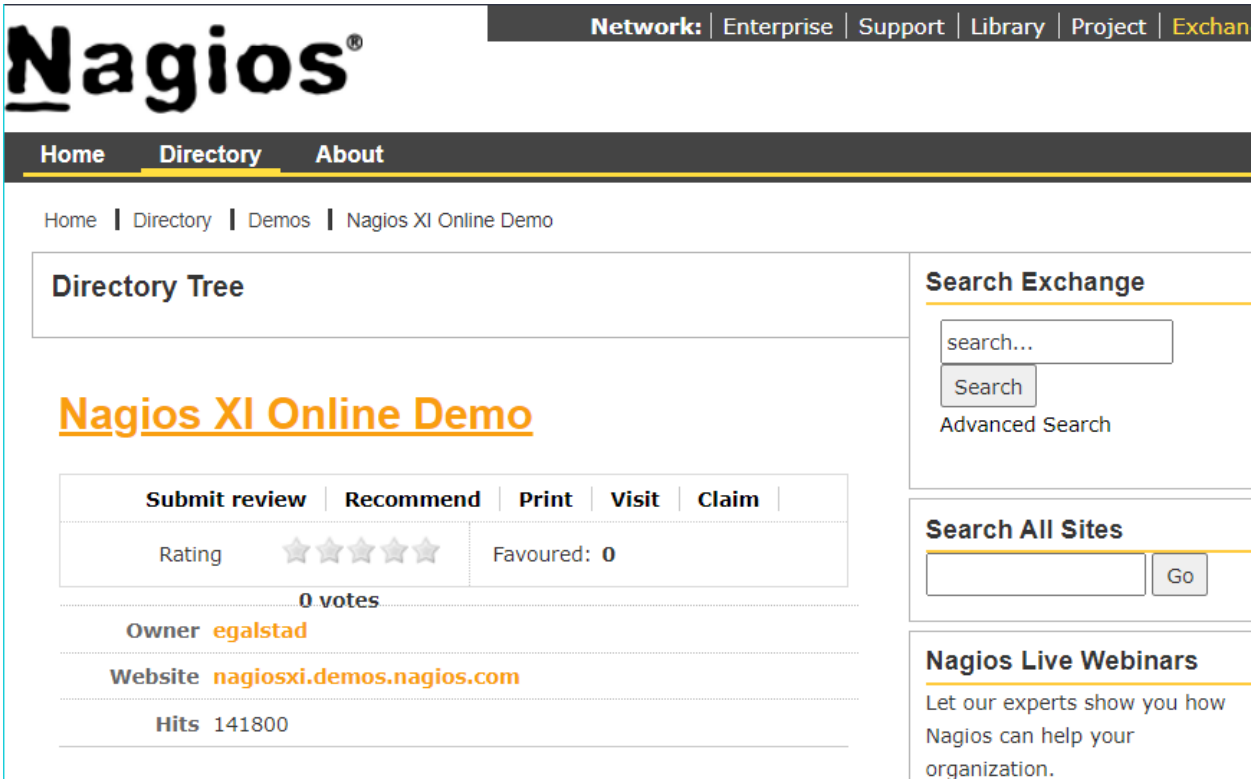
Integration with Third-Party Tools: Nagios can integrate with various third-party tools and services, making it a versatile monitoring solution that can fit into different IT ecosystems and workflows.

Steps :-

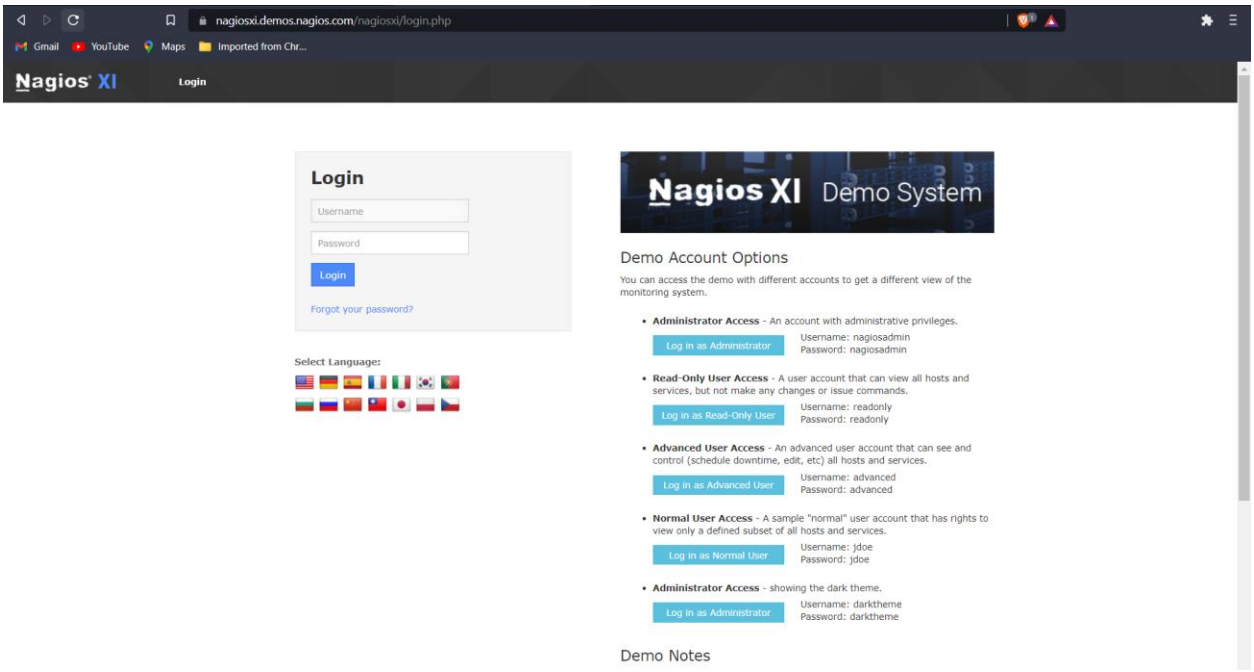
1) Go to google.com, Search Nagios Demo
Click on the first link shown below

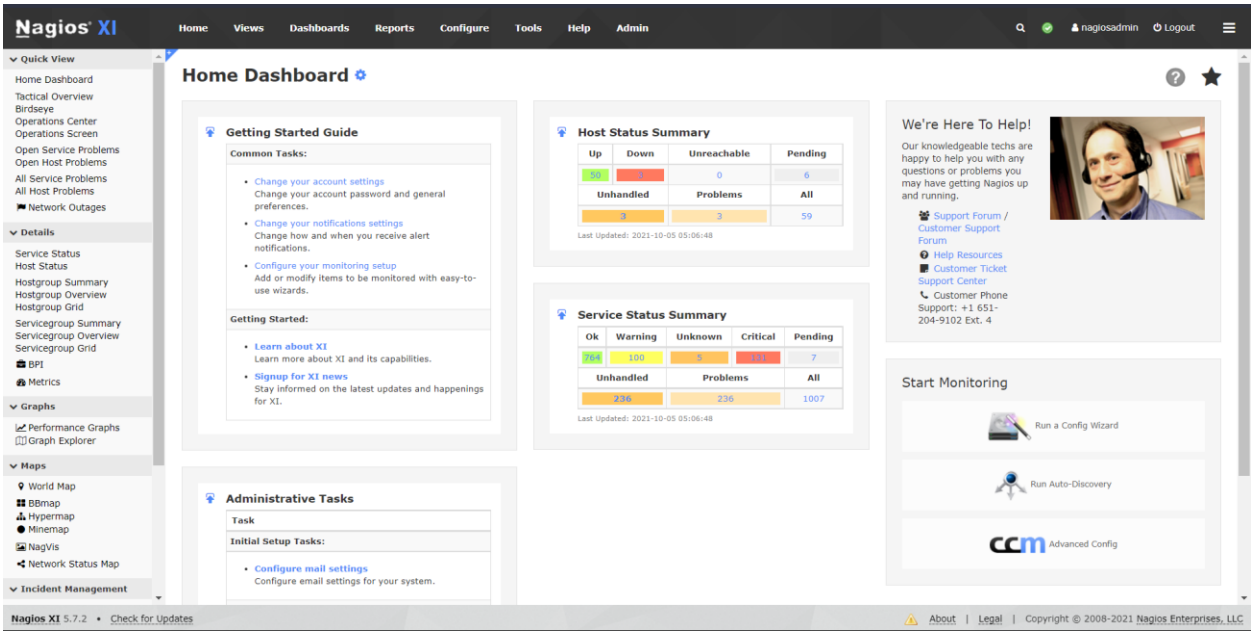


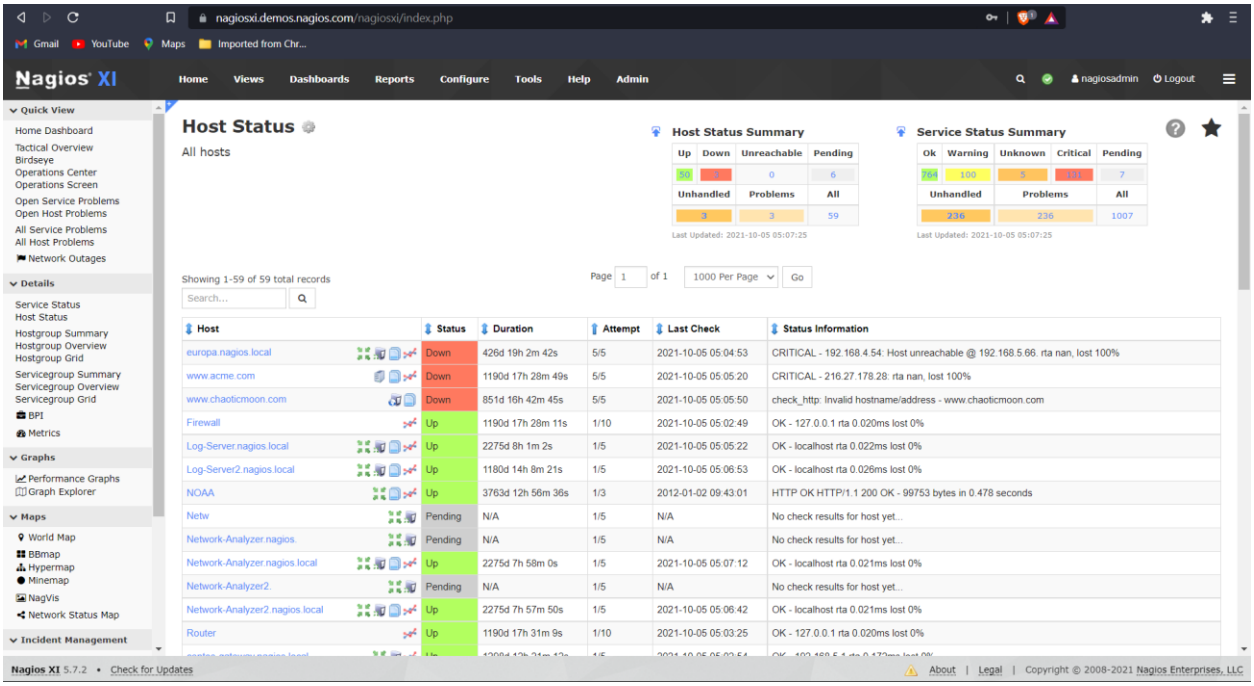
2) Now click on the website-



3) Now click on login as administrator

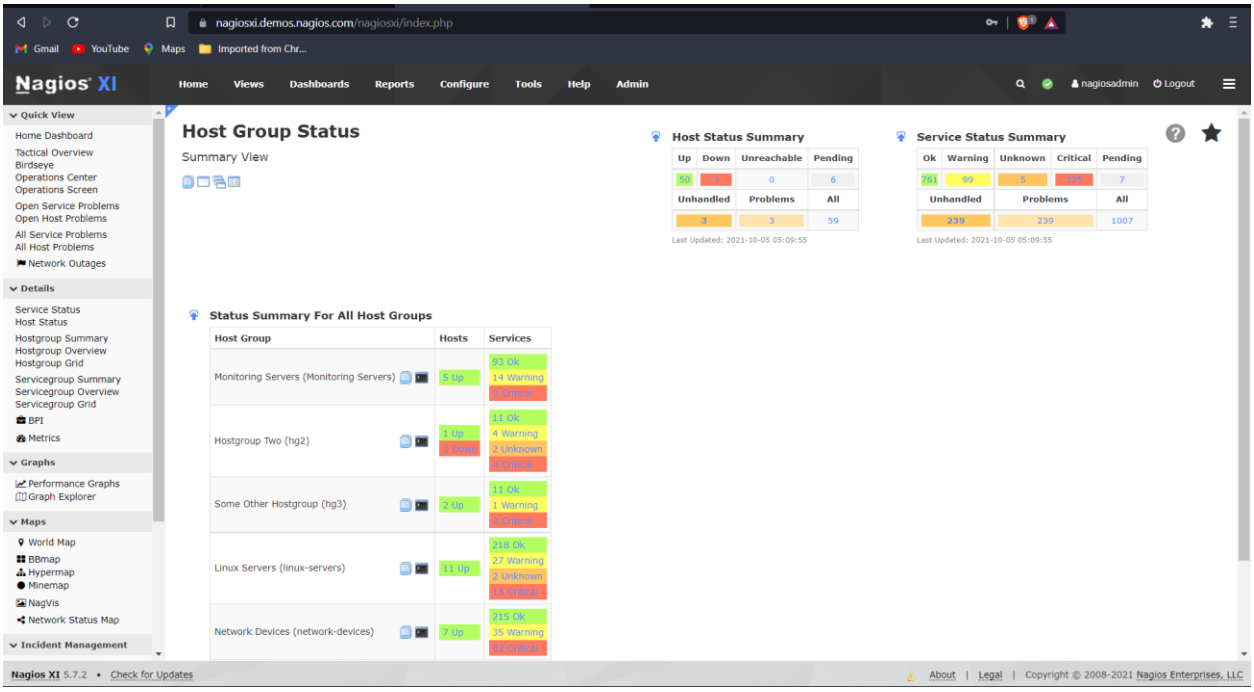






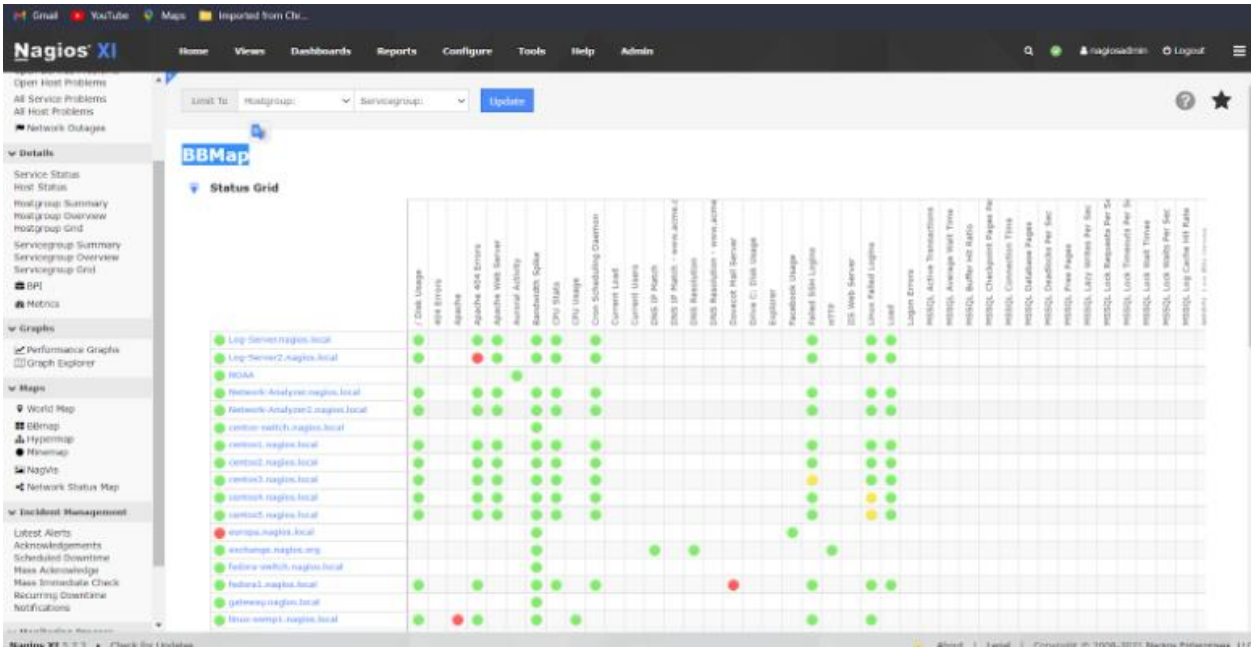
In the above image one can see Host Status Summary and Service Status Summary also how many host are up, down and also errors in detail

5) Now click on Host Group Status.



6) Now we click on BBMap

In this we can see status of following stuff in each host-



7) Now we have Network status map which is graphical representation of the network status

Conclusion:-

Continuous monitoring with Nagios enables proactive detection of system issues, ensuring minimal downtime and enhanced operational efficiency. Through customizable alerts and comprehensive reporting, Nagios empowers administrators to maintain optimal performance across diverse IT environments. Its scalable architecture and robust community support make it an invaluable tool for streamlined and centralized monitoring.

Roll No.:44
Name: Atharva Jadhav
Batch: T13
DATE: 13/10/23

ADVANCE DEVOPS ASSIGNMENT-9

AIM:- To understand AWS Lambda functions

LO MAPPED: LO1, LO5 THEORY:

AWS Lambda is a serverless compute service provided by Amazon Web Services (AWS) that enables you to run code in response to various events without the need to manage servers. It's a key component of AWS's serverless computing offerings. To understand the theory behind AWS Lambda functions, let's break it down into its core concepts:

Serverless Computing:

Serverless computing is a cloud computing model where cloud providers (like AWS) manage the infrastructure for you, allowing you to focus solely on your code.

Lambda Function:

A Lambda function is the fundamental unit of execution in AWS Lambda. It is a piece of code that can be executed in response to events such as HTTP requests, file uploads, database changes, etc. Lambda supports multiple programming languages, including Node.js, Python, Java, and more.

Event Sources:

Lambda functions are triggered by events. These events can come from various AWS services, like Amazon S3, Amazon DynamoDB, Amazon API Gateway, or custom events from your applications. Lambda listens to these event sources and automatically executes the code you've configured.

Stateless Execution:

Lambda functions are stateless, meaning they don't maintain any server-specific state between invocations. Each invocation of a Lambda function is independent and isolated from the others.

Scaling and Concurrency:

AWS Lambda automatically scales based on the number of incoming events. If there are more events, AWS will create more instances of your Lambda function to handle the load, and if there are fewer events, it will scale down accordingly. You pay only for the compute time your code consumes.

Execution Environment:

Each Lambda function runs in an execution environment provided by AWS. You can't control or manage this environment, but you can specify its configuration, including the amount of memory allocated to the function.

Function Versioning and Aliases:

You can create multiple versions of a Lambda function. This is useful for deploying and managing different versions of your code. You can also create aliases to point to specific versions, allowing you to easily switch between them.

IAM Roles:

Lambda functions can assume AWS Identity and Access Management (IAM) roles. These roles define what AWS services and resources the function can interact with, ensuring proper security and access control.

Logging and Monitoring:

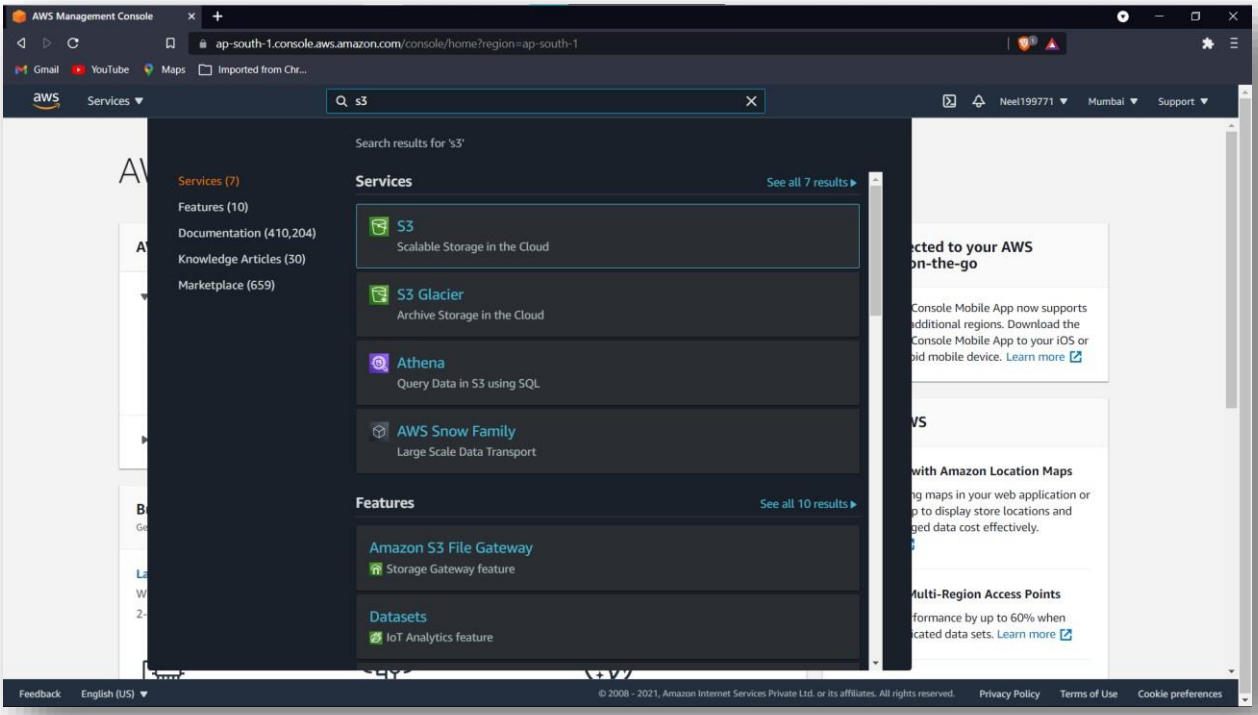
AWS Lambda provides built-in logging to capture function execution details. You can also integrate it with AWS CloudWatch for monitoring and creating custom metrics.

Triggers and Destinations:

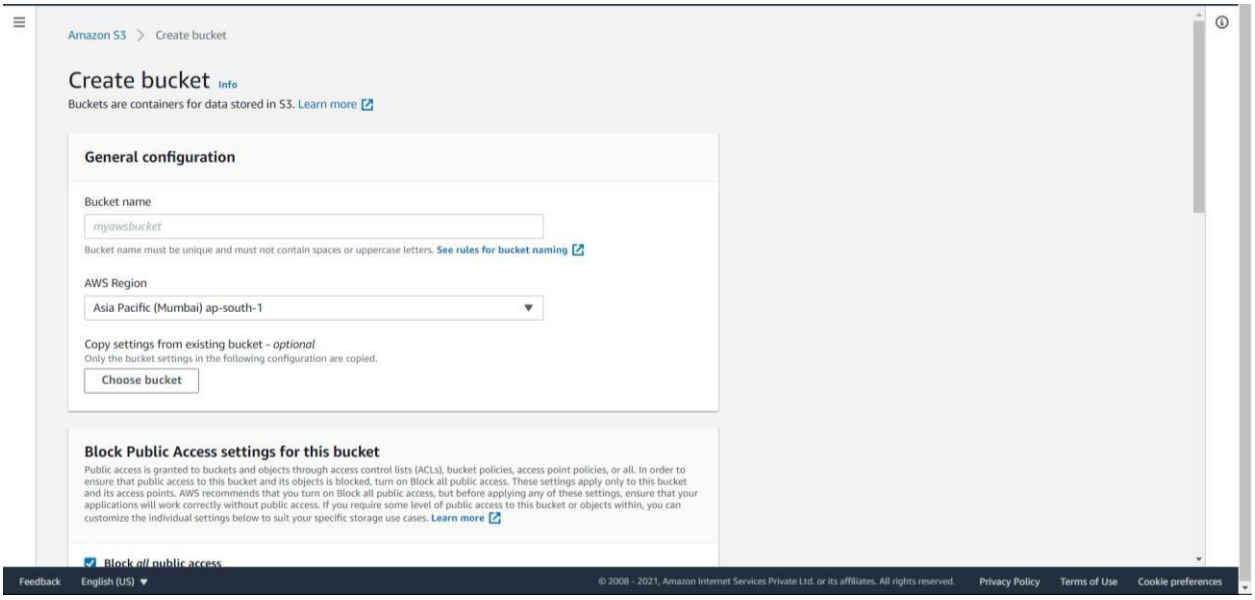
Lambda can be triggered by various event sources and can send the results to destinations such as other AWS services, like S3, DynamoDB, SNS, and more.

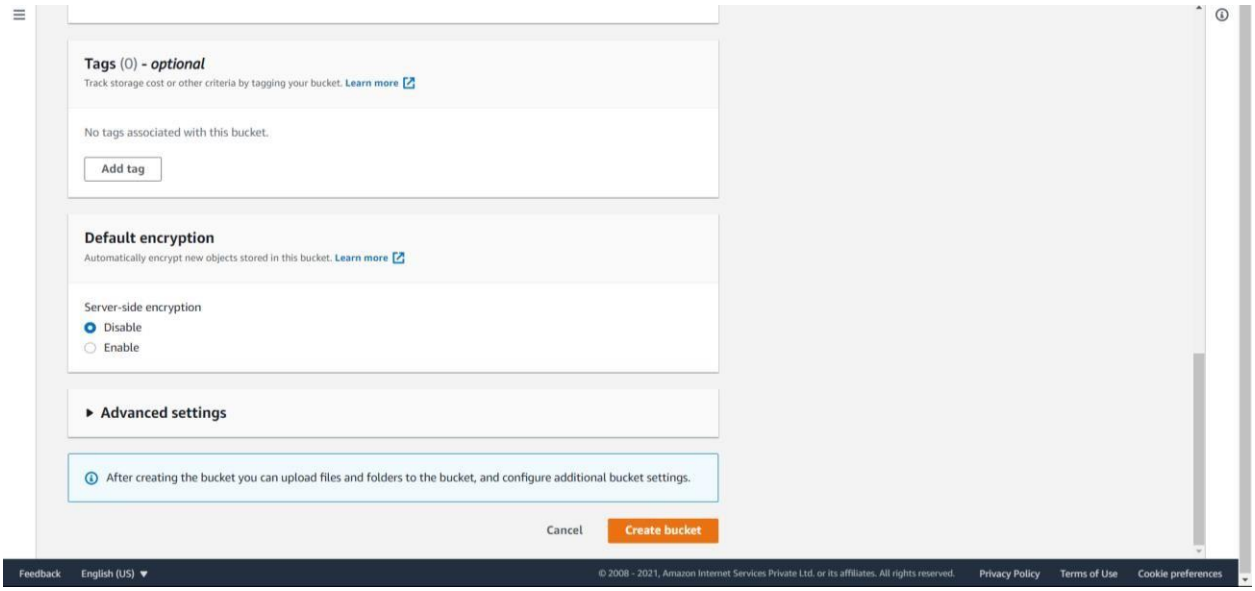
STEPS:

Login to Aws account- Search S3 ,click on the option below shown-

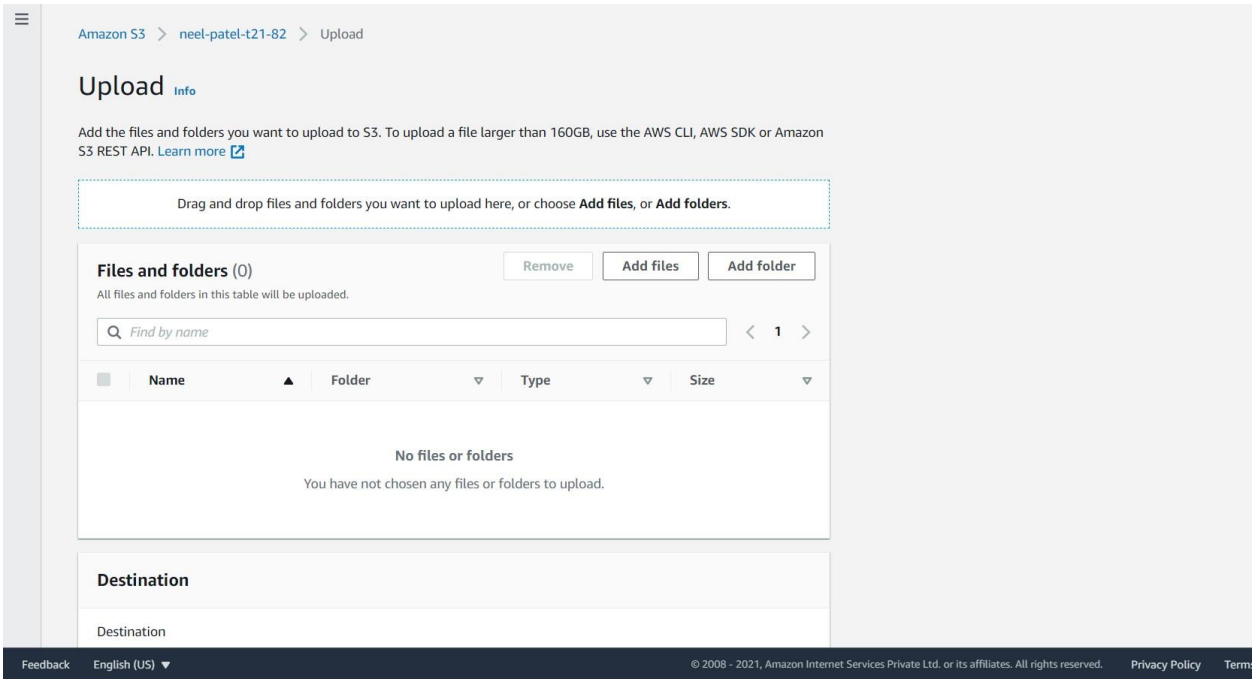


Create an S3 bucket by giving it a name

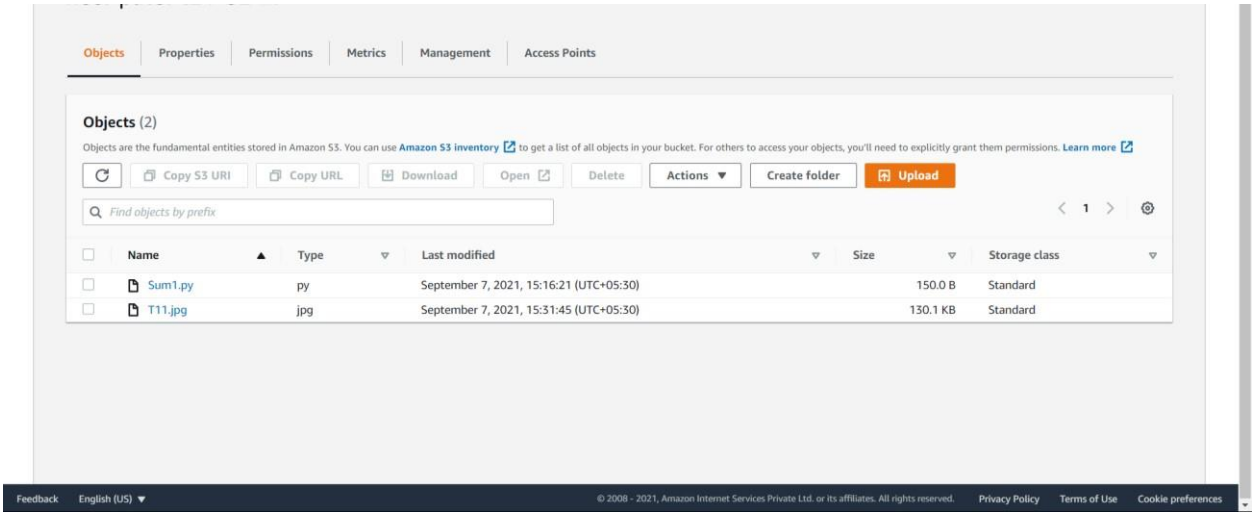




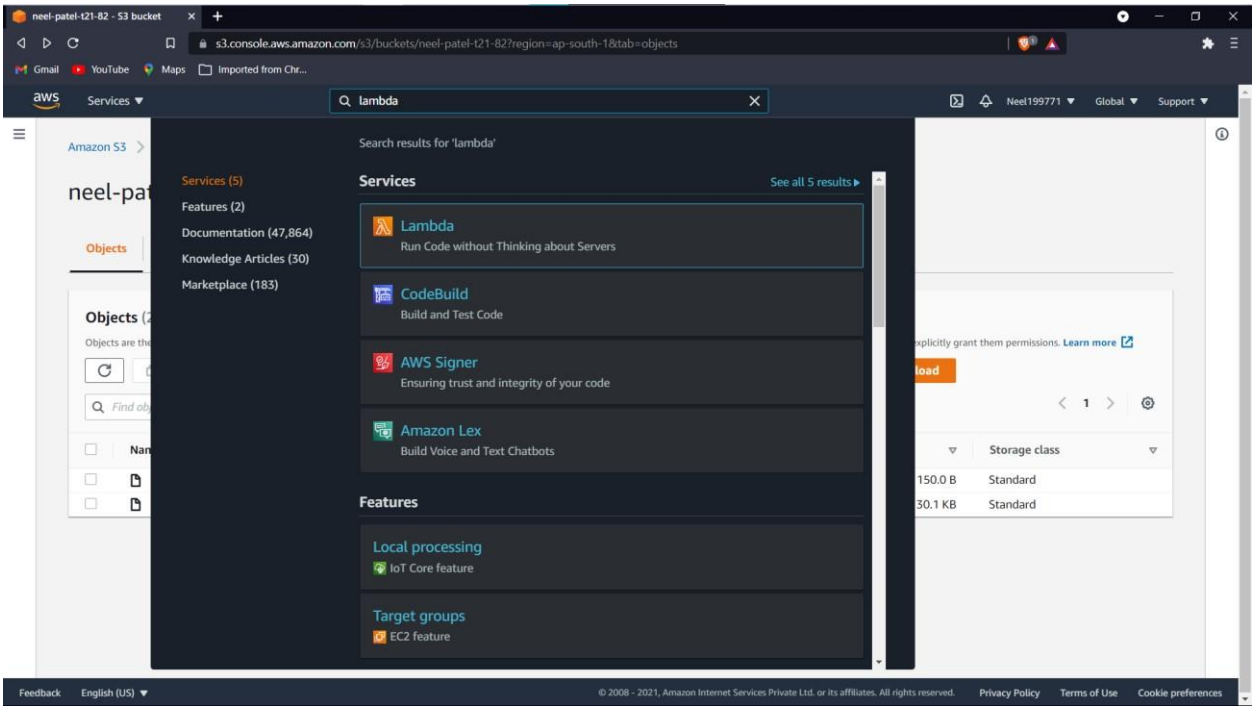
Click on upload button after the s3 bucket is created in the object section



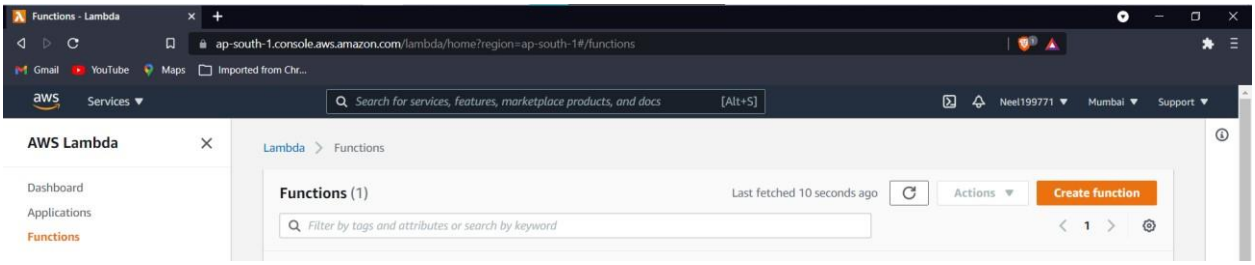
Add any .py or .java extenstion file and click on upload



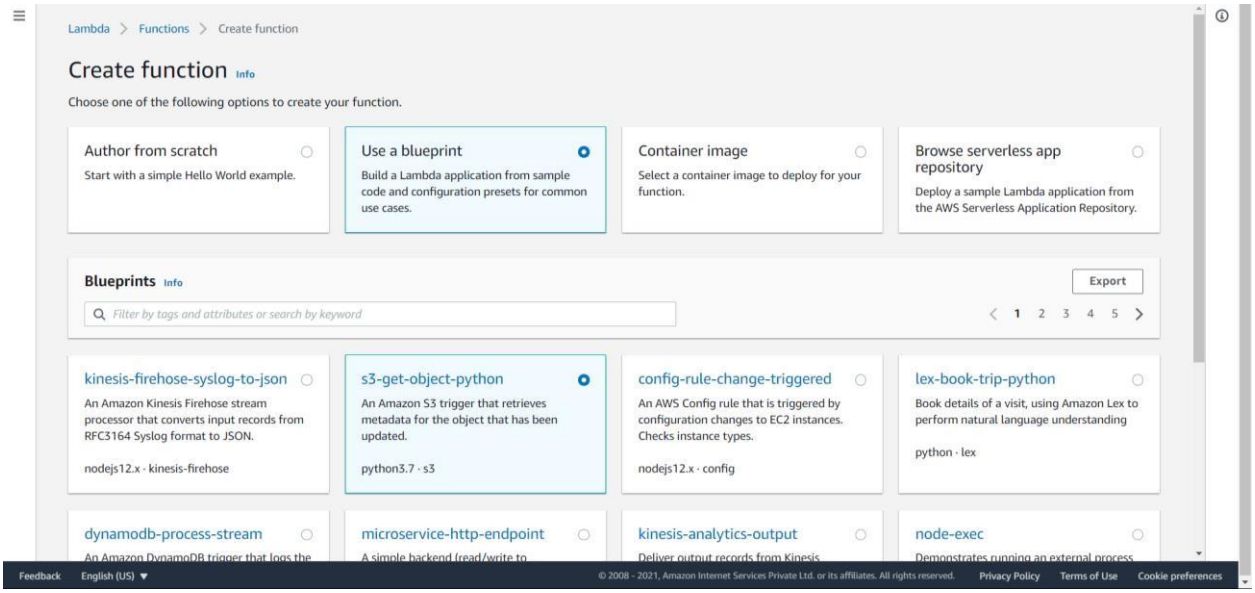
Now search Lamda

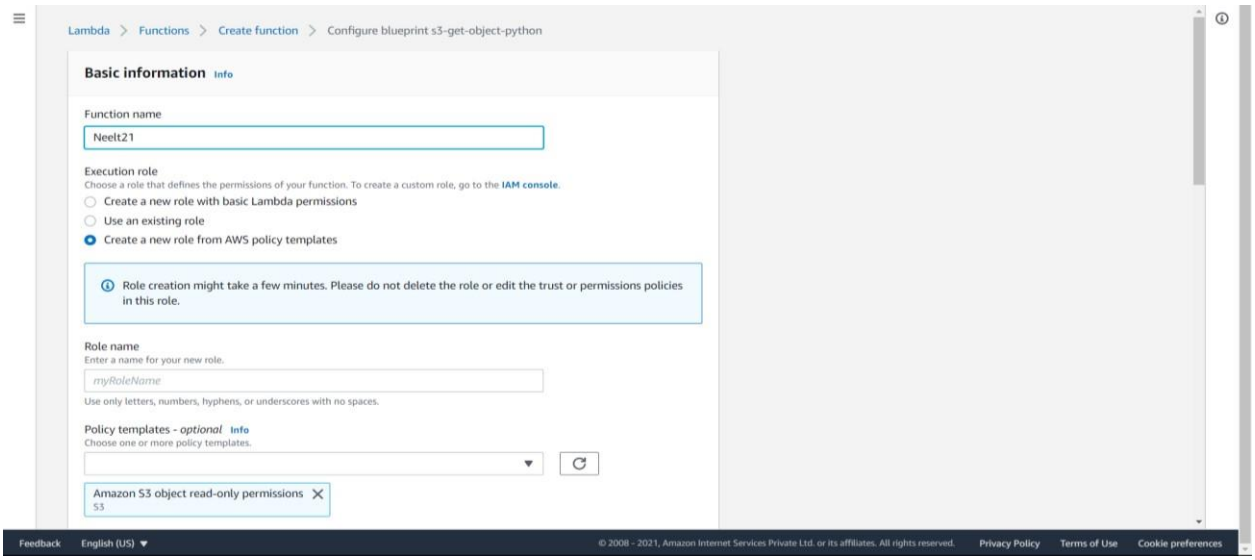


Click create function

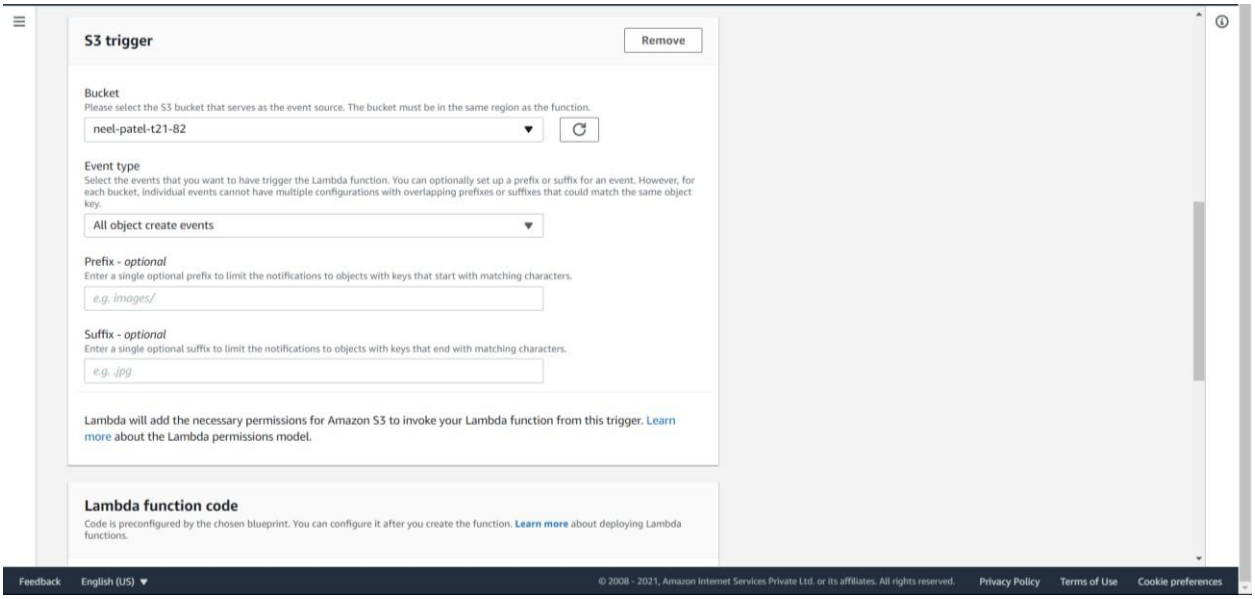


Click on below options and click on configure

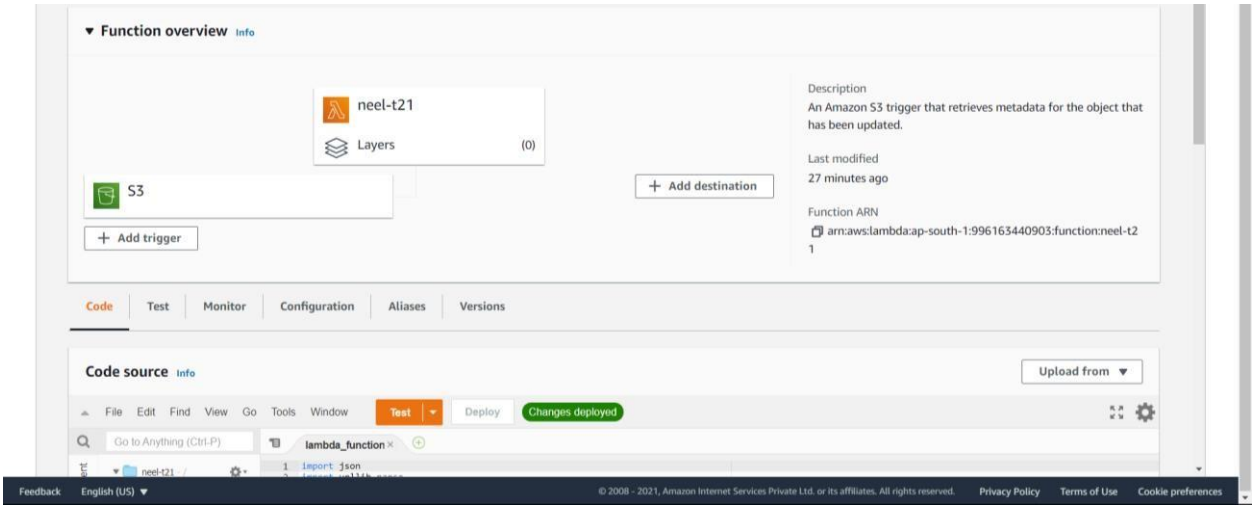




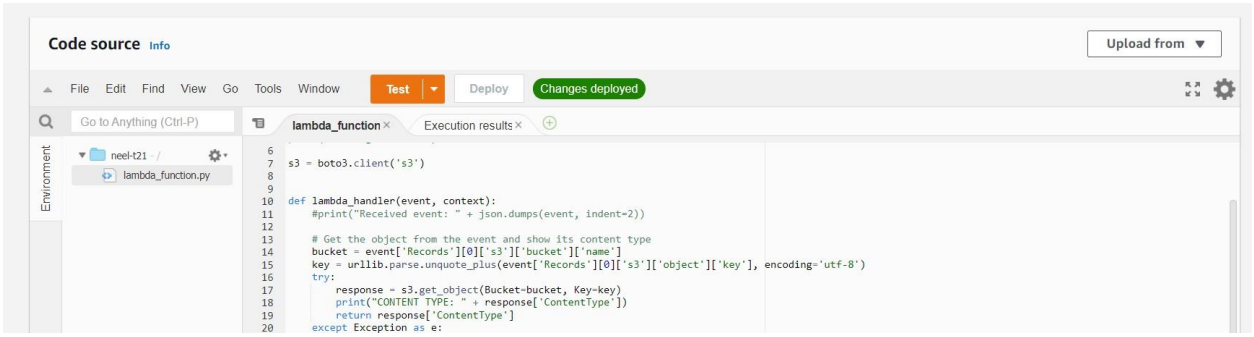
Select the bucket created and create trigger ,click on create function-



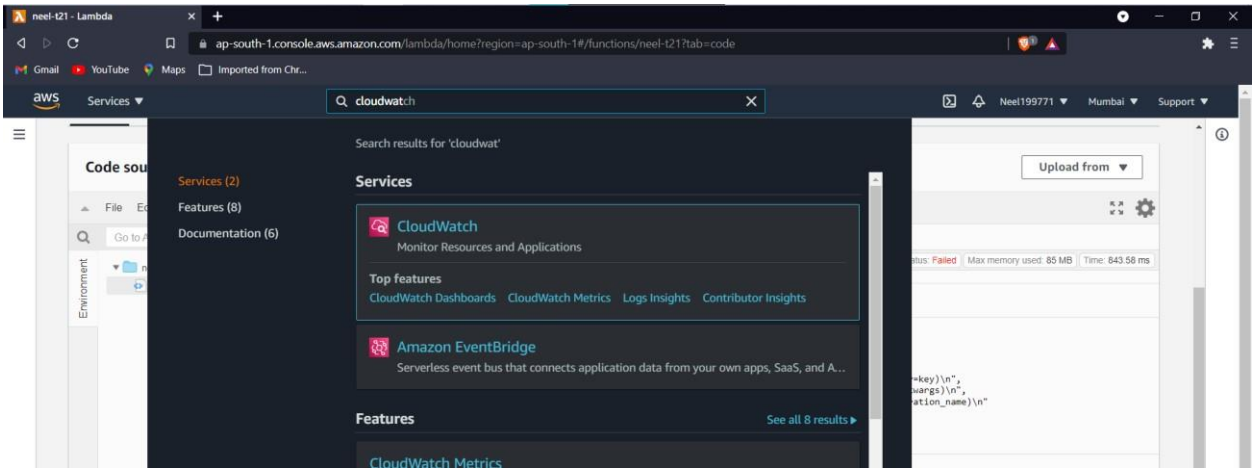
Check the given trigger is created



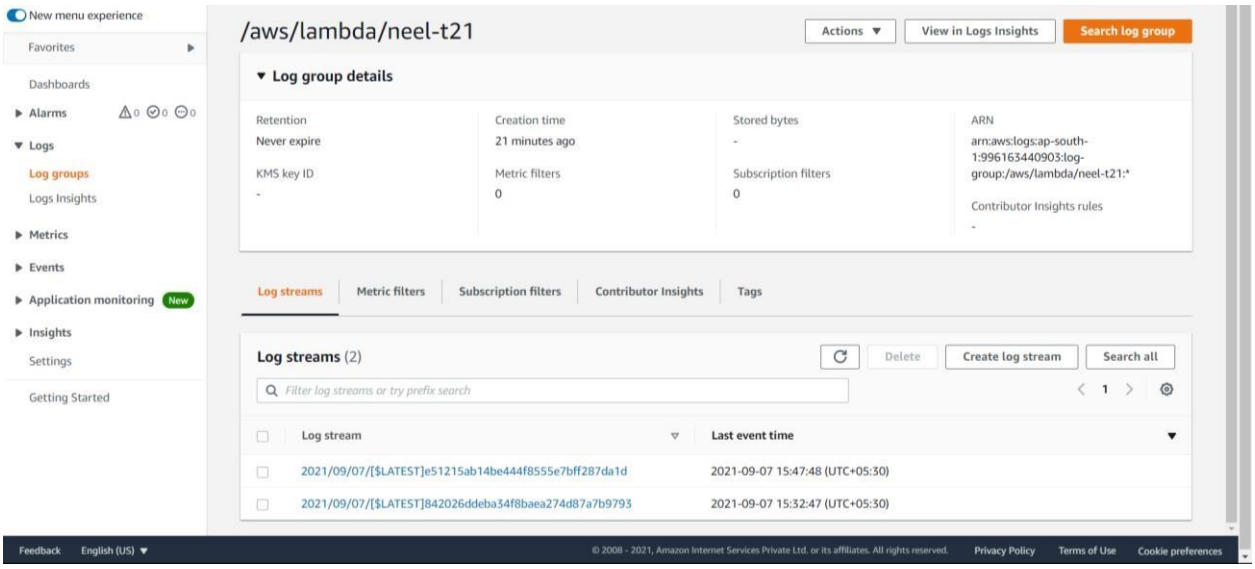
Click on the orange test button-



Now,

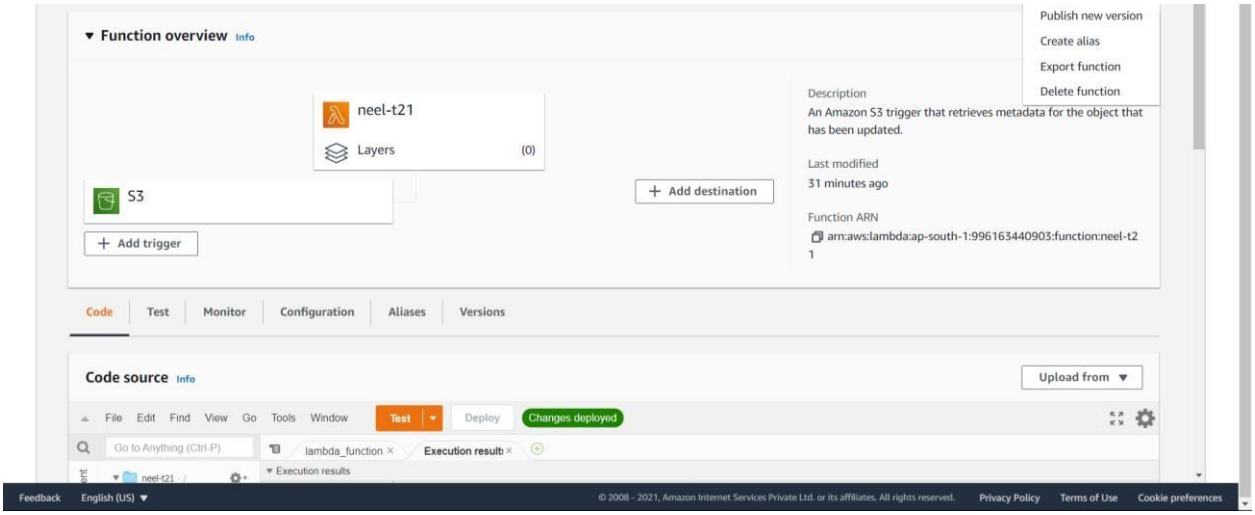


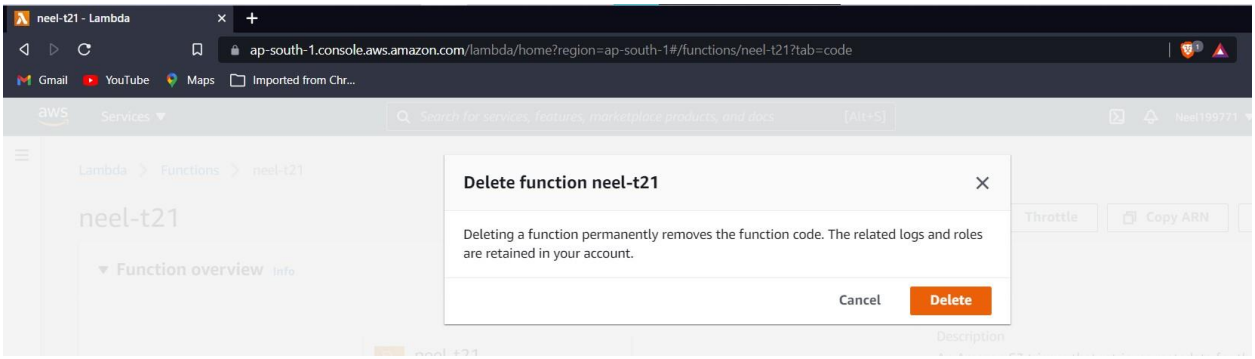
Check the logs of the test-



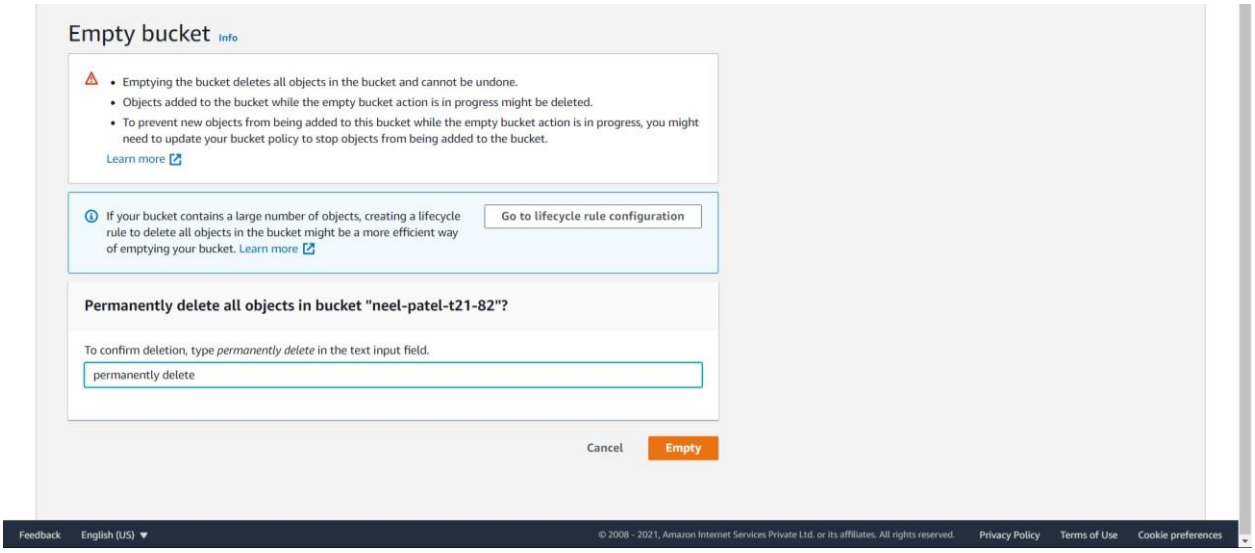
Now terminate-

Click on delete function.

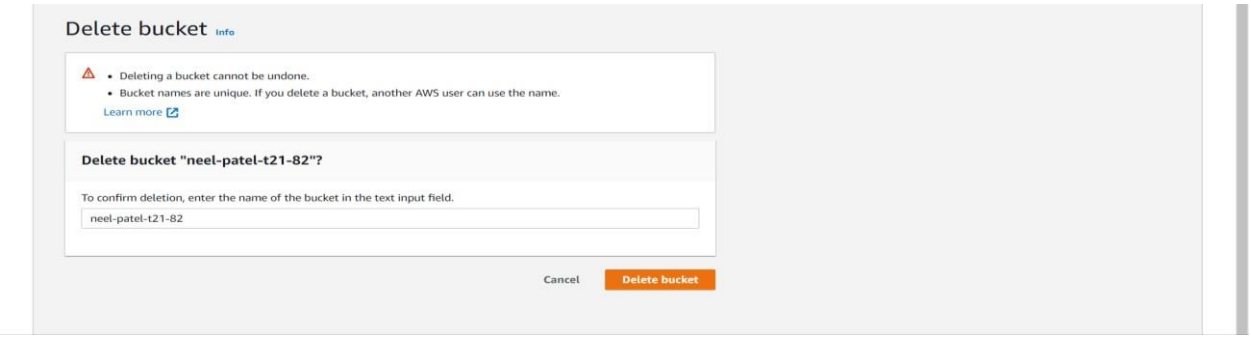




Empty bucket



Delete bucket-



CONCLUSION:

In this assignment, we covered the core concepts and terminologies of AWS Lambda, explored its practical applications, and gained an understanding of how it integrates with various AWS services.

Name :- Atharva Jadhav
Roll No:- 2105044
Batch:- T13
Date Of Performance :- 05/09/2023

Experiment 10

Aim:-

To create a Lambda function using Python for adding data to Dynamo DB database.

Theory:-

DYNAMO DB

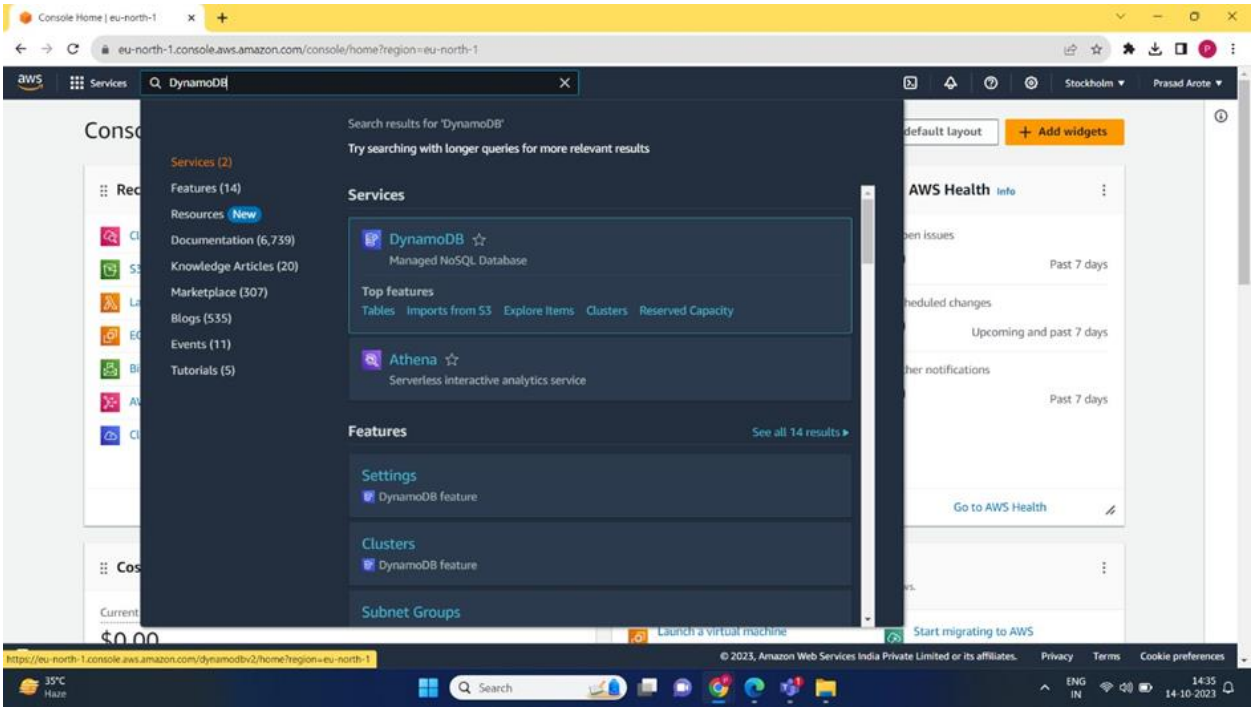
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data.

With DynamoDB, you can create database tables that can store and retrieve any amount of data and serve any level of request traffic. You can scale up or scale down your tables' throughput capacity without downtime or performance degradation. You can use the AWS Management Console to monitor resource utilization and performance metrics

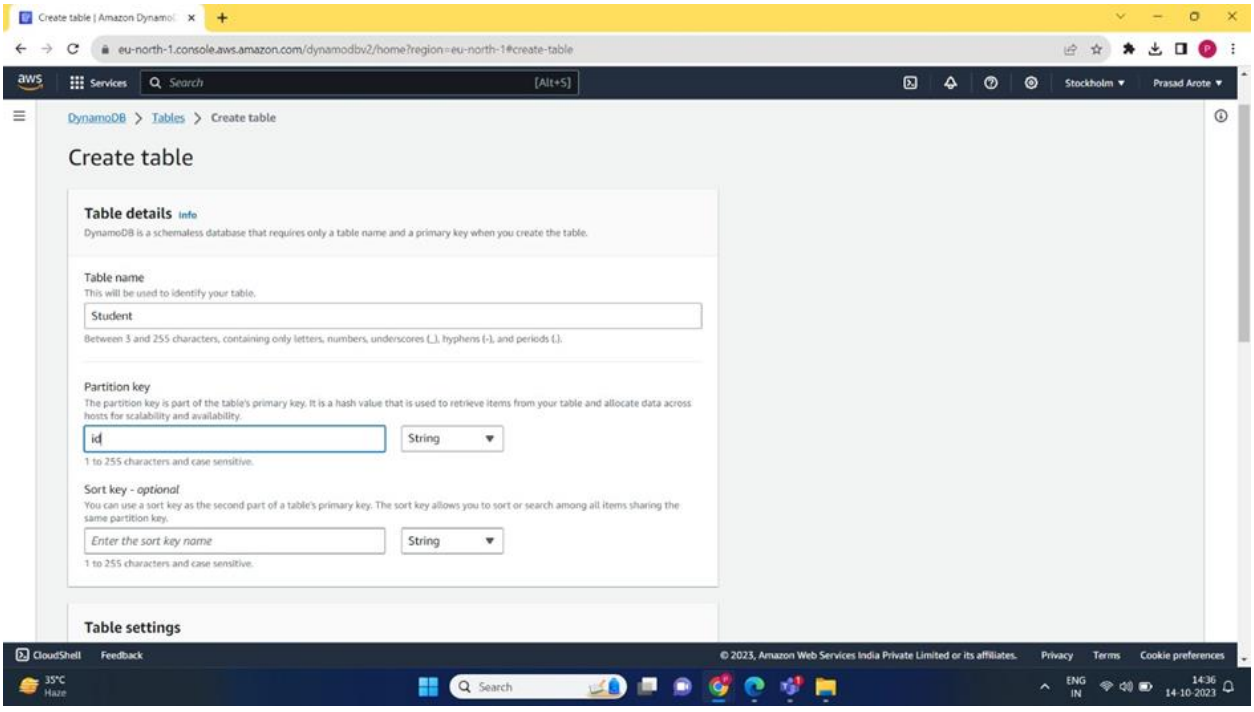
DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Steps :-

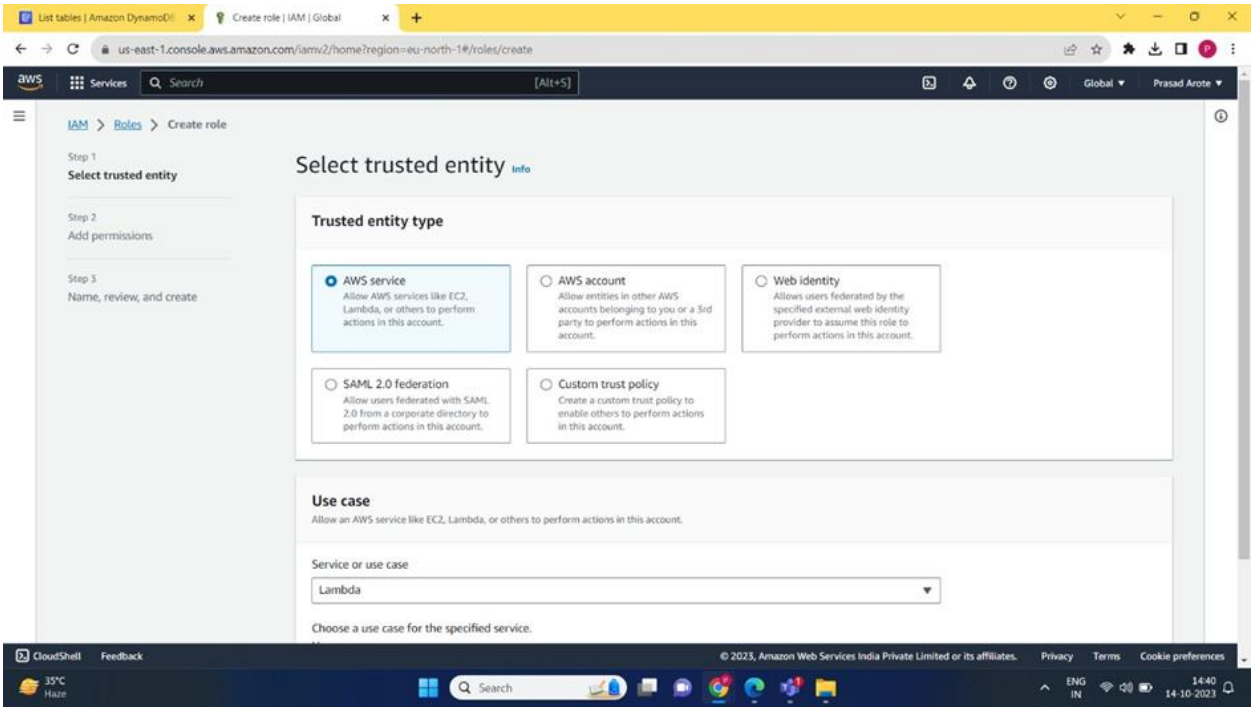
1) Login to AWS account and search for DynamoDB in search bar

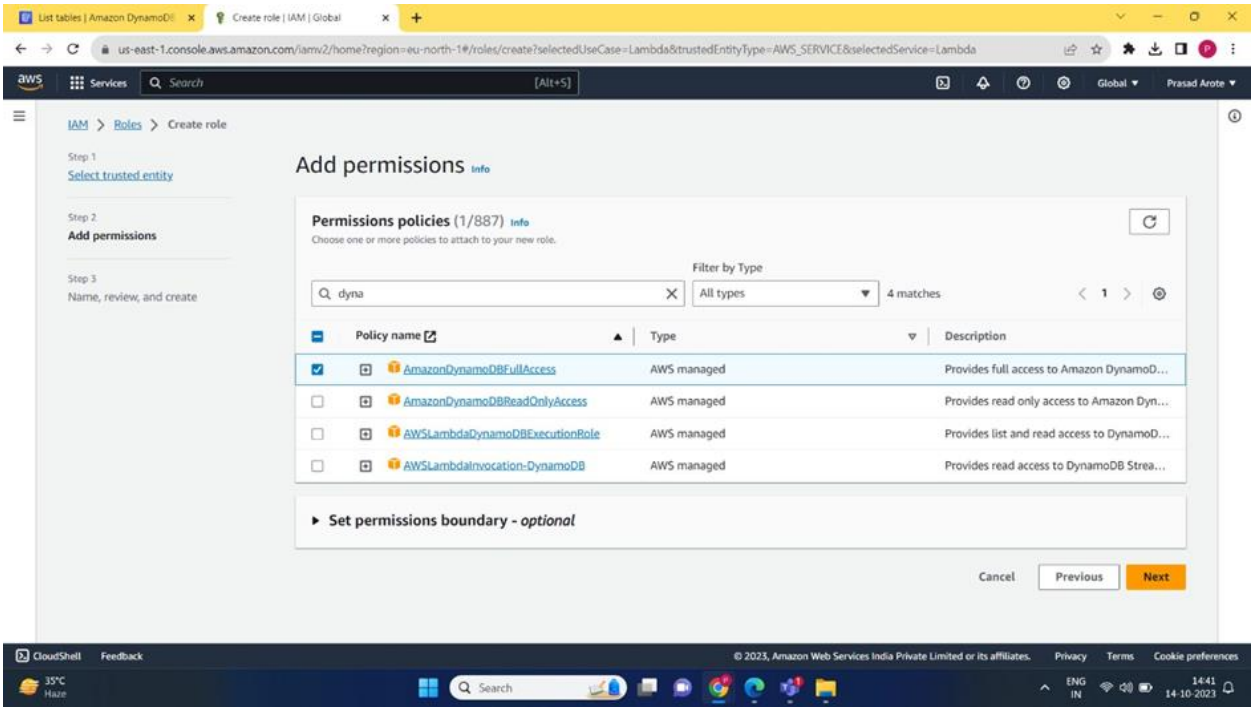


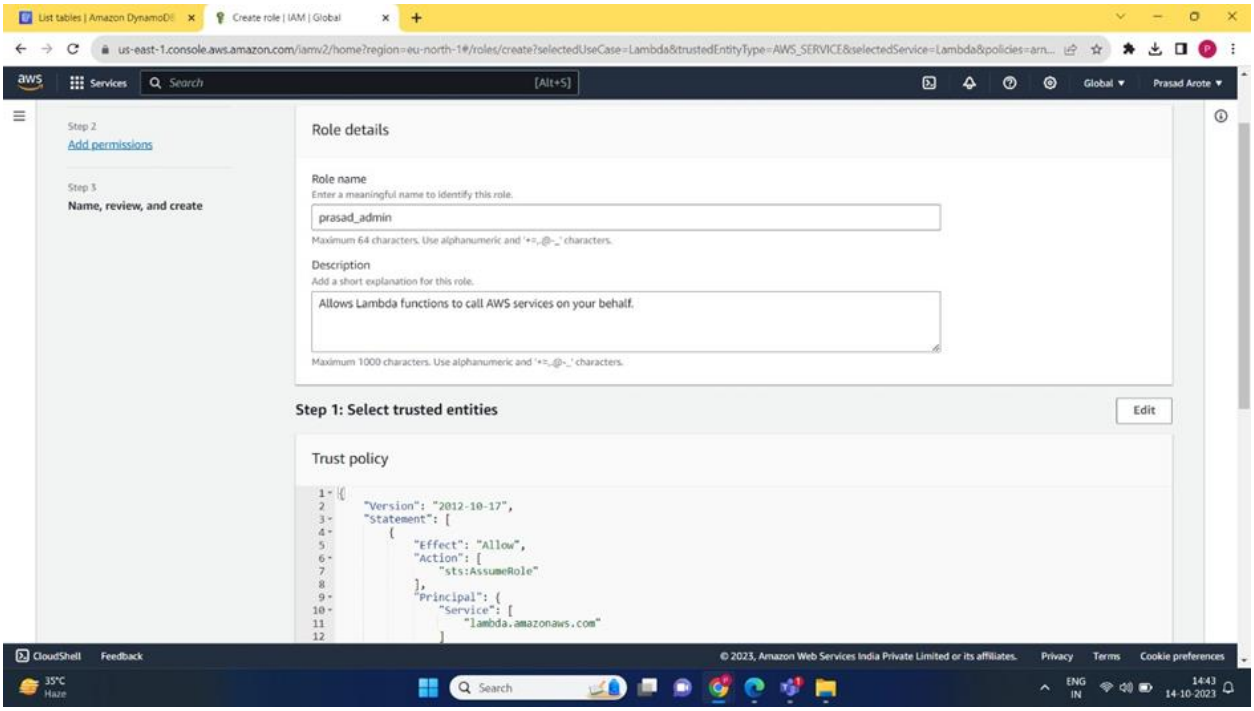
2) Click on DynamoDB option shown above and then click on create table

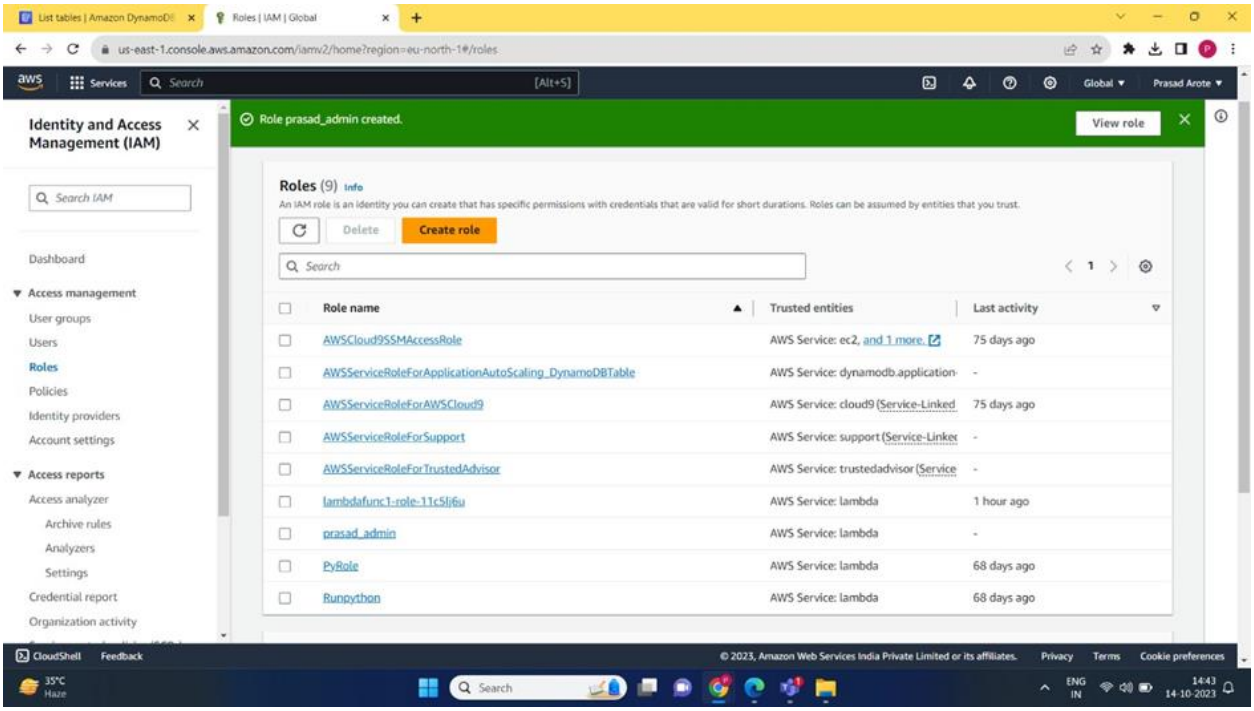


3)Then search IAM in the search box above and create a new role , give AmazonDynamoFullAccess permission to created user

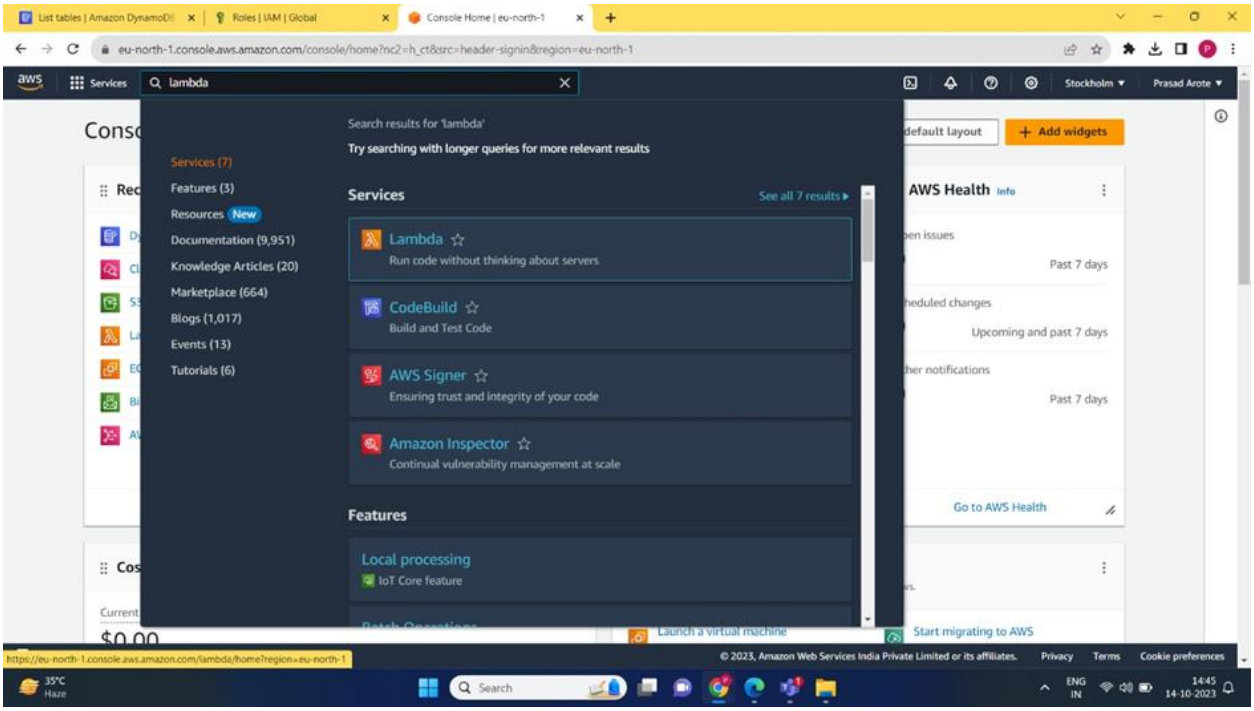

List tables | Amazon DynamoDB
Create role | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/roles/create
Services
Search
[Alt+S]
Global
Prasad Arote
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Select trusted entity Info
Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
Web identity
Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
SAML 2.0 federation
Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
Custom trust policy
Create a custom trust policy to enable others to perform actions in this account.
Use case
Allow an AWS service like EC2, Lambda, or others to perform actions in this account.
Service or use case
Lambda
Choose a use case for the specified service.
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Haze
Search
ENG
IN
14:40
14-10-2023

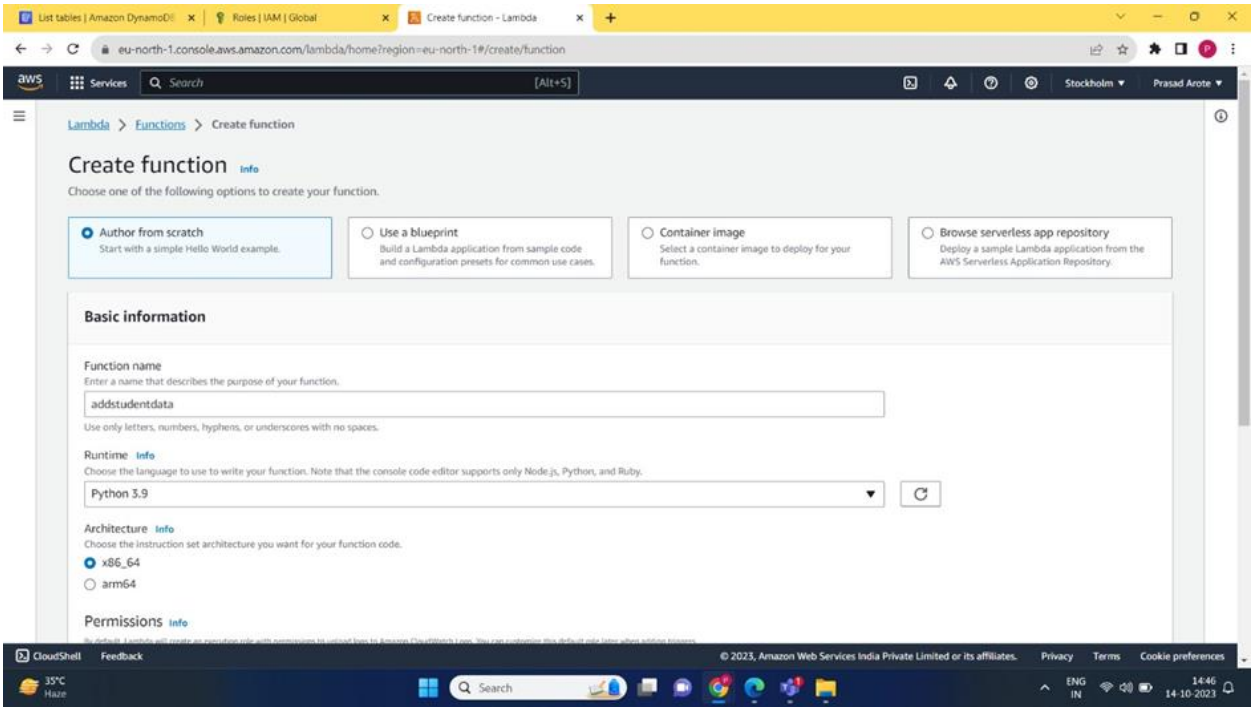

List tables | Amazon DynamoDB
Create role | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/roles/create?selectedUseCase=Lambda&trustedEntityType=AWS_SERVICE&selectedService=Lambda
Services
Search
[Alt+S]
Global
Prasad Arote
IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (1/887) Info
Choose one or more policies to attach to your new role.
Filter by Type
dyna
All types
4 matches
1
Policy name
Type
Description
AmazonDynamoDBFullAccess
AWS managed
Provides full access to Amazon DynamoD...
AmazonDynamoDBReadOnlyAccess
AWS managed
Provides read only access to Amazon Dyn...
AWSLambdaDynamoDBExecutionRole
AWS managed
Provides list and read access to DynamoD...
AWSLambdaInvocation-DynamoDB
AWS managed
Provides read access to DynamoDB Strea...
Set permissions boundary - optional
Cancel
Previous
Next
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Haze
Search
ENG
IN
14:41
14-10-2023

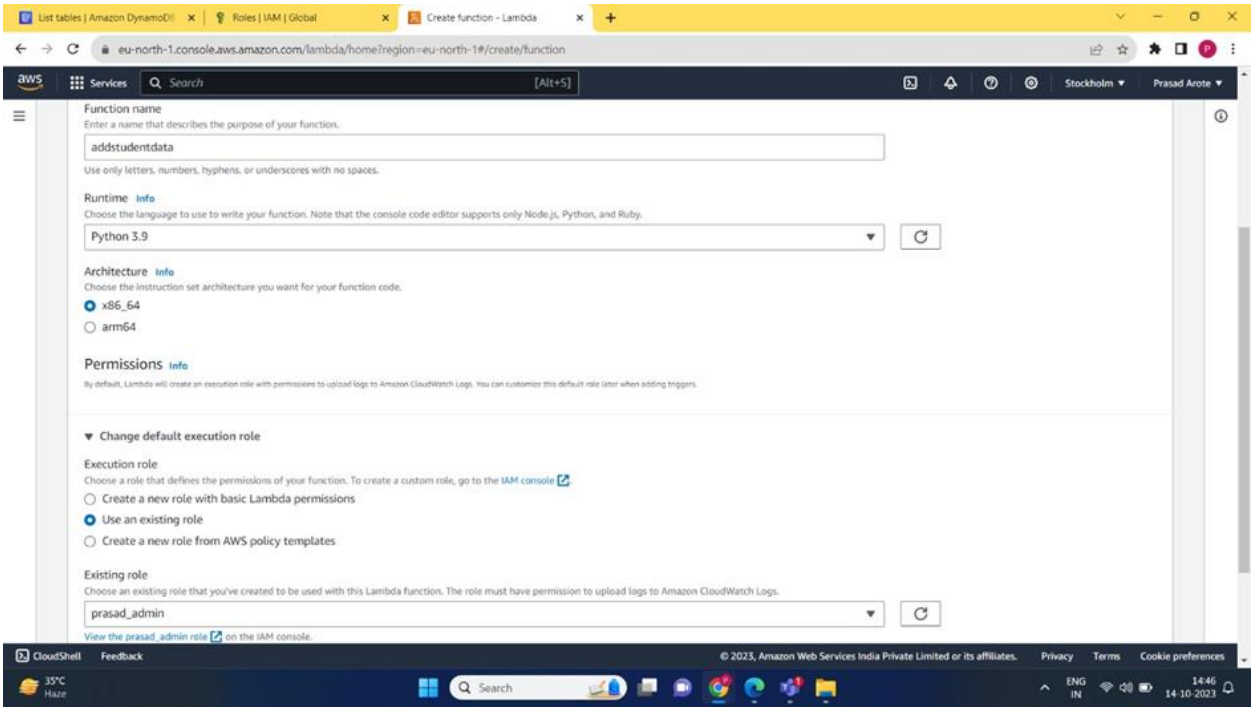


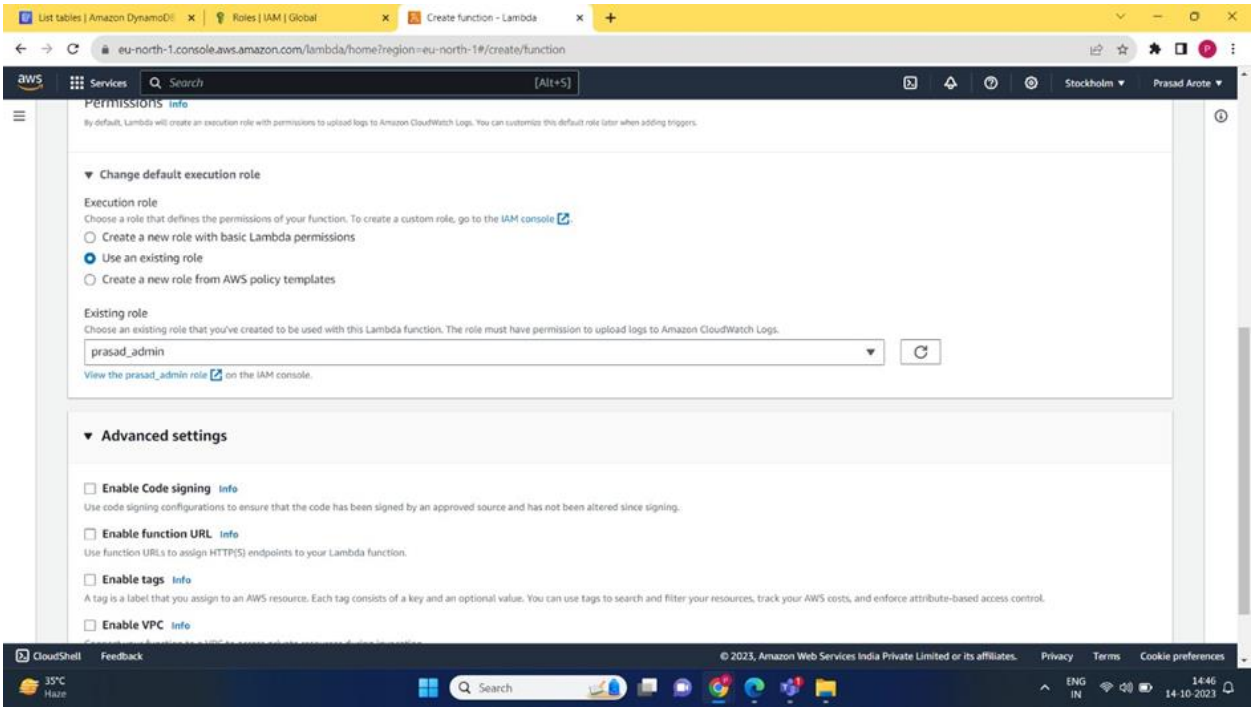


4) Search Lambda in search box and click on it , then create a new lambda function

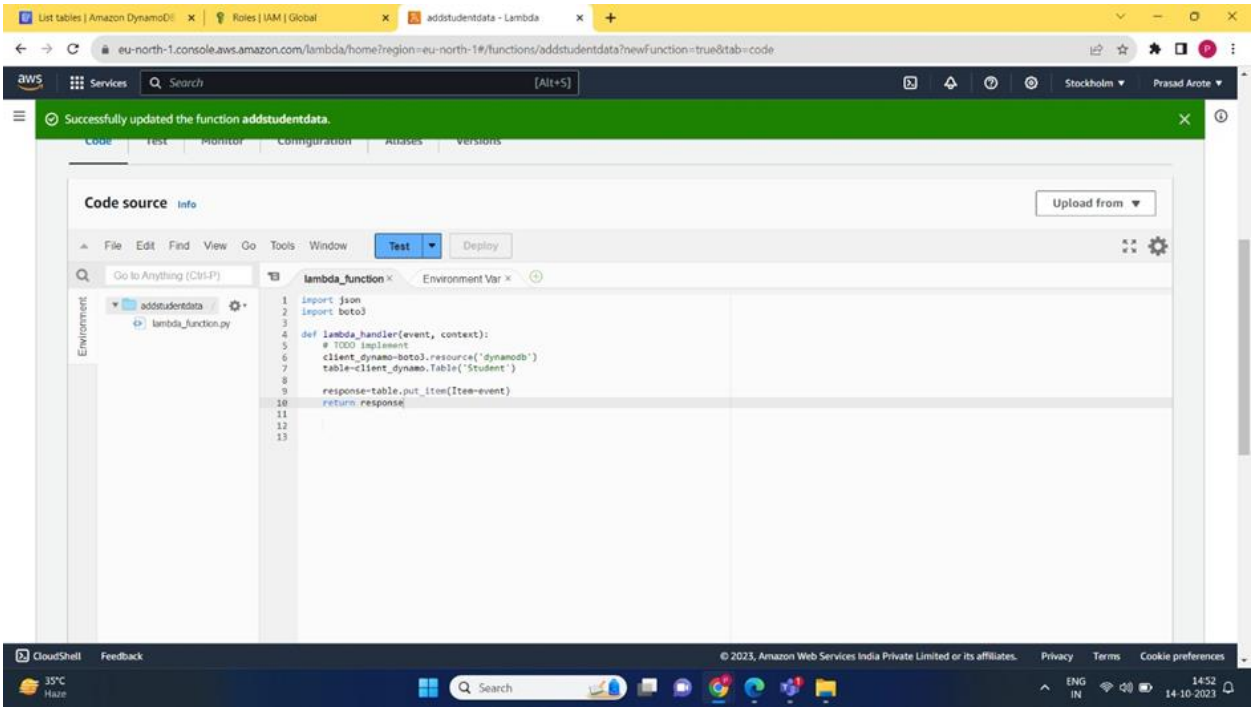

List tables | Amazon DynamoDB
Roles | IAM | Global
Console Home | eu-north-1
eu-north-1.console.aws.amazon.com/console/home?nc2=h_ct&src=header-signin&region=eu-north-1
aws
Services
lambda
Stockholm
Prasad Arote
Search results for 'lambda'
Try searching with longer queries for more relevant results
Services (7)
Features (3)
Resources New
Documentation (9,951)
Knowledge Articles (20)
Marketplace (664)
Blogs (1,017)
Events (13)
Tutorials (6)
Services
See all 7 results
Lambda
Run code without thinking about servers
CodeBuild
Build and Test Code
AWS Signer
Ensuring trust and integrity of your code
Amazon Inspector
Continual vulnerability management at scale
Features
Local processing
IoT Core feature
default layout
Add widgets
AWS Health Info
Past 7 days
Upcoming and past 7 days
Go to AWS Health
Launch a virtual machine
Start migrating to AWS
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Haze
Search
ENG
IN
14:45
14-10-2023


List tables | Amazon DynamoDB
Roles | IAM | Global
Create function - Lambda
eu-north-1.console.aws.amazon.com/lambda/home?region=eu-north-1#/create/function
aws
Services
Search
[Alt+S]
Stockholm
Prasad Arote
Lambda > Functions > Create function
Create function Info
Choose one of the following options to create your function.
Author from scratch
Start with a simple Hello World example.
Use a blueprint
Build a Lambda application from sample code and configuration presets for common use cases.
Container image
Select a container image to deploy for your function.
Browse serverless app repository
Deploy a sample Lambda application from the AWS Serverless Application Repository.
Basic information
Function name
Enter a name that describes the purpose of your function.
addstudentdata
Use only letters, numbers, hyphens, or underscores with no spaces.
Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.9
Architecture Info
Choose the instruction set architecture you want for your function code.
x86_64
arm64
Permissions Info
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
35°C
Haze
Search
ENG
IN
14:46
14-10-2023

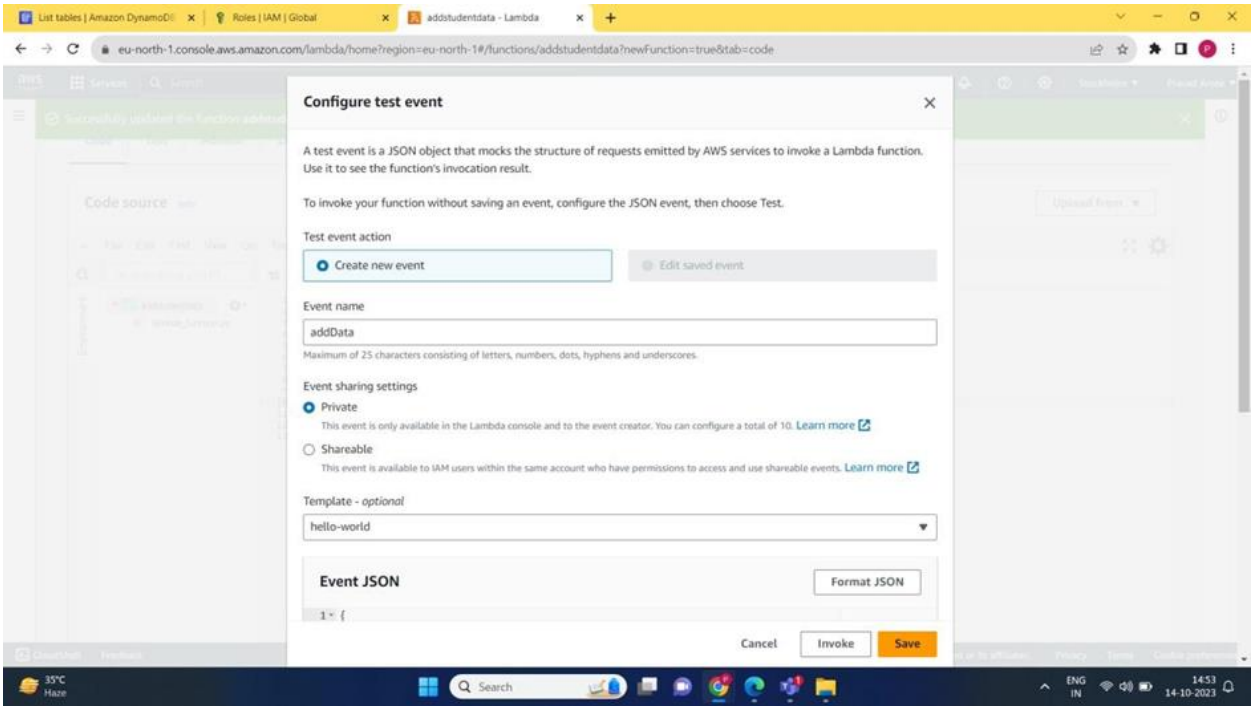


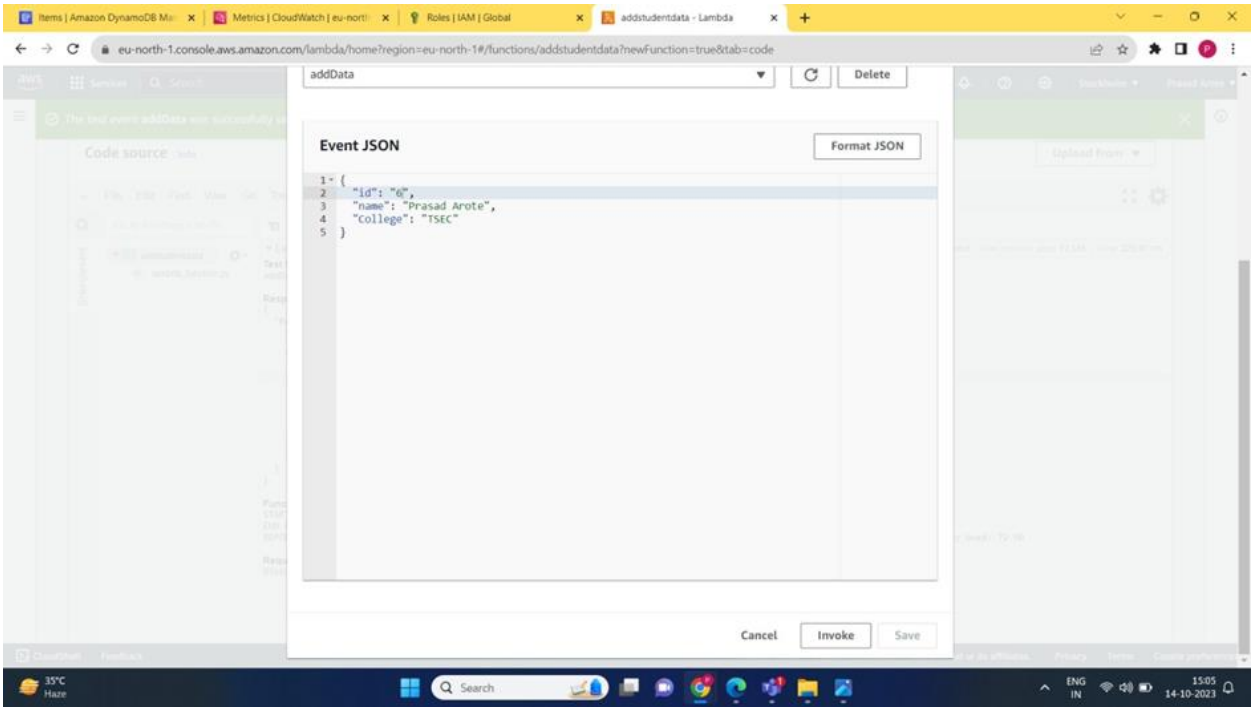


5) Write the following code in code source

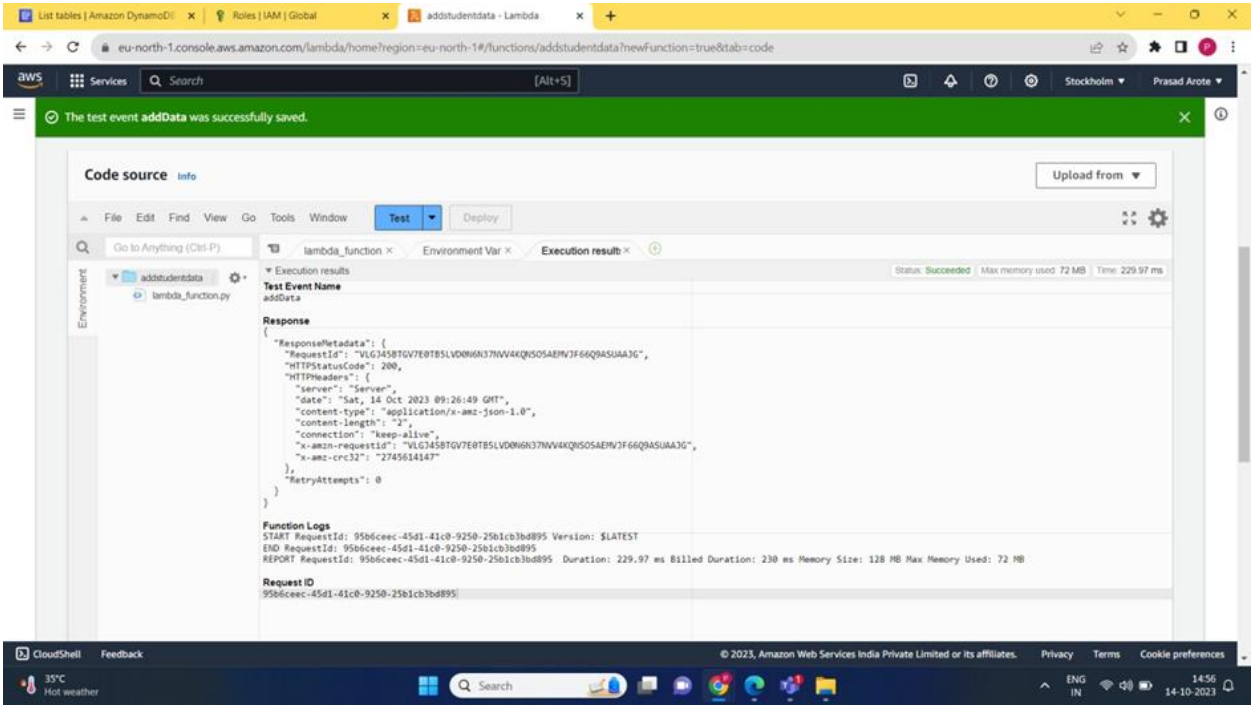


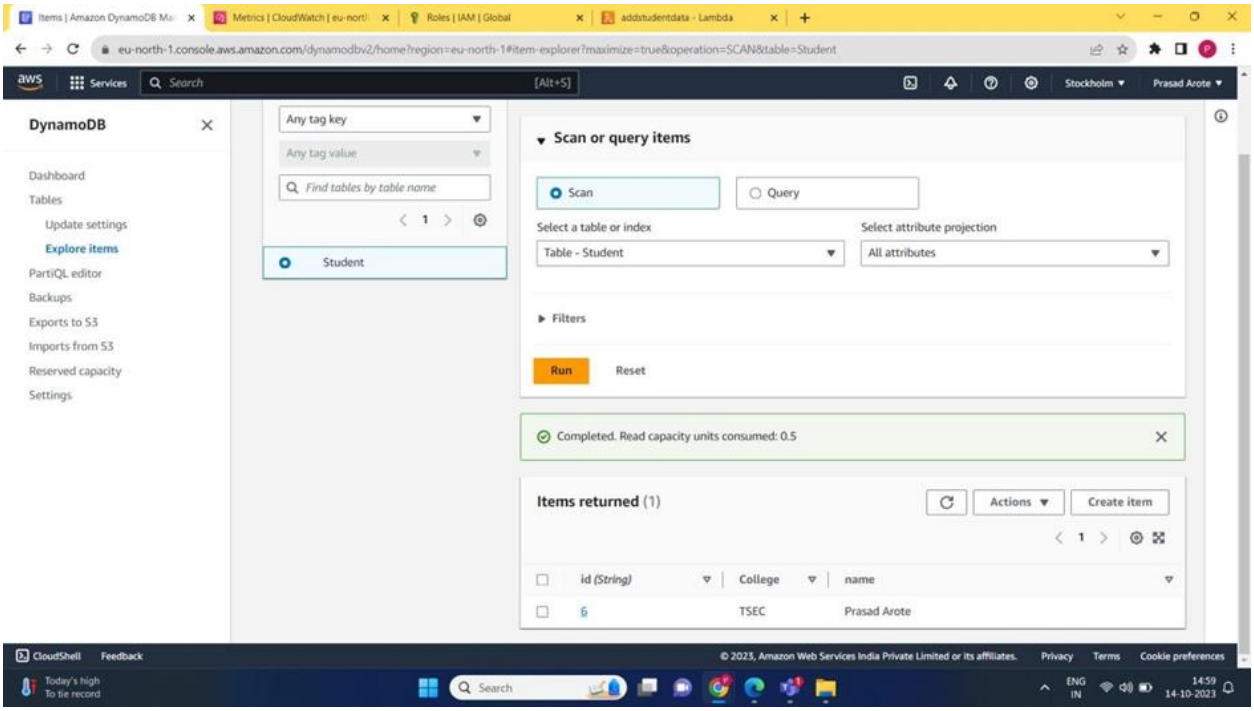
6) Configure the test event and save





7) Run the test and afterwards go to the DynamoDB>Explore items> Student where you can see the record inserted using lambda function.





Conclusion:-

Learnt about Amazon DynamoDB database service and inserted data into DynamoDB database by creating a new user , granting him permissions and then using a lambda function

Name :- Atharva Jadhav
RollNo :- 2105044
Batch :- T13
Date Of Performance :- 13/10/2023

Written Assignment 1:Study of Kubernetes

Q1) What security measures can be taken while using kubernetes ?

There are a number of security measures that can be taken while using Kubernetes. Some of the most important include:

1)Use Role-Based Access Control (RBAC): RBAC allows you to define who has access to the Kubernetes API and what permissions they have. This is essential for preventing unauthorized access to your cluster and its resources.
2)Use third-party authentication for the API server: This allows you to integrate Kubernetes with an existing identity provider, such as GitHub or Okta. This can make it easier to manage user accounts and permissions.
3)Protect etcd with TLS and a firewall: etcd is a distributed key-value store that stores the state of your Kubernetes cluster. It is a critical component, so it is important to protect it from unauthorized access.
4)Isolate Kubernetes nodes: Kubernetes nodes are the machines that run your containerized applications. It is important to isolate them from the rest of your network to prevent attackers from gaining access to your cluster.
5)Monitor network traffic to limit communications: Kubernetes workloads can communicate with each other over the network. It is important to monitor this traffic and limit it to only the necessary communication paths.
6)Use process whitelisting: Process whitelisting allows you to define which processes are allowed to run on your Kubernetes nodes. This can help to prevent attackers from running malicious code on your cluster.
7)Turn on audit logging: Audit logging records all activity on your Kubernetes cluster. This can help you to detect and investigate security incidents.
8)Keep Kubernetes up to date: The Kubernetes team regularly releases security updates. It is important to keep your Kubernetes cluster up to date to protect against known vulnerabilities.
9)Lock down the Kubelet: The Kubelet is a service that runs on each Kubernetes node and is responsible for managing pods. It is important to lock down the Kubelet to prevent attackers from gaining control of your nodes.

In addition to these general security measures, there are a number of other specific things you can do to secure your Kubernetes cluster, such as:

10)Use Kubernetes namespaces to isolate your workloads: Namespaces allow you to group your workloads together and isolate them from each other. This can help to prevent attackers from moving laterally between your workloads.
11)Use Pod Security Policies (PSPs) to restrict the privileges of your pods: PSPs allow you to define what resources and privileges your pods are allowed to use. This can help to prevent attackers from gaining access to sensitive data or running malicious code on your cluster.
12)Use a service mesh to manage network traffic between your workloads: A service mesh can help you to secure and manage the network traffic between your workloads. This can help to prevent attackers from communicating with your workloads or eavesdropping on their traffic.
13)Use a security scanner to scan your container images for vulnerabilities: A security scanner can help you to identify and fix vulnerabilities in your container images before they are deployed to your cluster.
14)Implement a security incident response plan: A security incident response plan will help you to respond to security incidents in a timely and effective manner.

By following these security measures, you can help to protect your Kubernetes cluster and its workloads from attack.

Q2) What are the three security techniques used to protect data ?

The three most important security techniques used to protect data are:

1) **Encryption**
2) **Access control**
3) **Backup and recovery**

Encryption is the process of converting data into a format that cannot be read without a secret key. This makes data unreadable to unauthorized individuals, even if they have access to it. Encryption can be used to protect data at rest, in transit, and in use.

Access control is the process of restricting access to data to authorized individuals. This can be done using a variety of methods, such as passwords, multi-factor authentication, and role-based access control (RBAC). Access control is important for preventing unauthorized individuals from accessing and modifying data.

Backup and recovery is the process of creating copies of data and storing them in a secure location. This is important for protecting data from loss or corruption. Backup and recovery plans should be regularly tested to ensure that they are working properly.

In addition to these three core security techniques, there are a number of other security measures that can be used to protect data, such as network security, physical security, and security awareness training.

Here are some examples of how these three security techniques can be used to protect data:

Encryption: A company can encrypt its customer database to protect it from unauthorized access, even if the database is compromised.
Access control: A hospital can use RBAC to restrict access to patient medical records to authorized personnel, such as doctors and nurses.
Backup and recovery: A government agency can regularly back up its financial data to a secure location in case of a cyberattack or natural disaster.

By using these security techniques, organizations can help to protect their data from unauthorized access, modification, and loss.

Q3) How do you expose a service using ingress in Kubernetes?

To expose a service using Ingress in Kubernetes, you need to create an **Ingress resource**. An Ingress resource specifies the rules for routing traffic to your services.

To create an Ingress resource, you can use the following command:
kubectl create ingress <ingress-name>

The Ingress resource must specify the following:

1) **The rules for routing traffic to your services.**
2) **The hostname or IP address that traffic will be routed to.**
3) **The port that traffic will be routed to.**

For example, the following Ingress resource exposes a service named my-service on port 80:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - http:
        paths:
        - path: /
        pathType: Prefix
```

```
      backend:
      service:
      name: my-service
      port: 80
```

Once you have created the Ingress resource, you can access the service at the hostname or IP address specified in the Ingress resource. For example, if the Ingress resource specifies the hostname my-service.example.com, you can access the service at my-service.example.com on port 80.

You can also use Ingress to expose multiple services on the same hostname or IP address. To do this, you can specify multiple rules in the Ingress resource. For example, the following Ingress resource exposes two services, my-service and my-other-service, on port 80:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - http:
      paths:
      - path: /
      pathType: Prefix
      backend:
      service:
      name: my-service
      port: 80
  - http:
      paths:
      - path: /other
      pathType: Prefix
      backend:
      service:
      name: my-other-service
      port: 80
```

Ingress is a powerful tool for exposing services in Kubernetes. It allows you to expose services on a specific hostname or IP address, and to expose multiple services on the same hostname or IP address

Q4) Which service protocol does Kubernetes ingress expose ?

Ingress is a Kubernetes resource that allows you to expose services running in a cluster to external traffic. **Ingress can expose services through either HTTP or HTTPS.**

To expose a service using Ingress, you need to create an Ingress resource. An Ingress resource specifies the rules for routing traffic to your services.

When you create an Ingress resource, you need to specify the following:

1) The rules for routing traffic to your services.
2) The hostname or IP address that traffic will be routed to.
3) The port that traffic will be routed to.

The Ingress resource also specifies the service protocol, which can be either HTTP or HTTPS.

To expose a service using HTTP, you need to specify the http protocol in the Ingress resource. For example, the following Ingress resource exposes a service named my-service on port 80 using HTTP:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - http:
        paths:
        - path: /
        pathType: Prefix
        backend:
        service:
        name: my-service
        port: 80
```

To expose a service using HTTPS, you need to specify the https protocol in the Ingress resource. For example, the following Ingress resource exposes a service named my-service on port 443 using HTTPS:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
metadata:
  name: my-ingress
spec:
  tls:
  - hosts:
        - my-service.example.com
        secretName: my-tls-secret
  rules:
  - http:
        paths:
        - path: /
        pathType: Prefix
        backend:
        service:
        name: my-service
        port: 443
```

Once you have created the Ingress resource, you can access the service at the hostname or IP address specified in the Ingress resource. For example, if the Ingress resource specifies the hostname my-service.example.com, you can access the service at my-service.example.com on port 80.

You can also use Ingress to expose multiple services on the same hostname or IP address. To do this, you can specify multiple rules in the Ingress resource.

Roll No:-44
Batch:- T13
Name :-Atharva Jadhav

ADVANCE DEVOPS WRITTEN ASSIGNMENT-2

Q.1 How to deploy Lambda function on AWS?

=>Deploying a Lambda function on AWS involves several steps. Here's a detailed guide on how to deploy a Lambda function:

Prerequisites:
An AWS account.
The AWS Command Line Interface (CLI) installed and configured with appropriate permissions.
Your Lambda function code packaged as a ZIP archive or uploaded to an Amazon S3 bucket.
Familiarity with the programming language and runtime you're using for your Lambda function (e.g., Node.js, Python, Java, etc.).

Step-by-Step Guide to Deploying a Lambda Function: Create or Prepare Your Lambda Function Code:

Write your Lambda function code or prepare it if you haven't already. Ensure it follows the AWS Lambda function structure, including the handler function. Package Your Code:

If your function code consists of multiple files or dependencies, package it as a ZIP archive. Make sure that the primary function handler is at the top level of the archive. Create an Execution Role (if needed):

Lambda functions often need permissions to interact with other AWS services. Create an AWS Identity and Access Management (IAM) role that grants the necessary permissions to your Lambda function. The role should have policies attached that allow access to AWS resources, like S3, DynamoDB, or others. Upload Code to Amazon S3 (if needed):

If your deployment package is larger than 3 MB, you will need to upload it to an Amazon S3 bucket. Make sure your Lambda function has permissions to access this bucket.

Deploy the Lambda Function:

You can deploy a Lambda function using the AWS Management Console, AWS CLI, AWS SDKs, or AWS CloudFormation. Here's how to deploy using the AWS CLI:

Open a terminal and run the following AWS CLI command to create your Lambda function:

aws lambda create-function --function-name MyFunctionName --runtime nodejs14.x role arn:aws:iam::123456789012:role/MyRole --handler index.handler --zip-file fileb://function.zip

Replace MyFunctionName with your desired function name.

Specify the correct runtime for your function (e.g., nodejs14.x for Node.js 14).

Use the --role option with the ARN of the IAM role you created. Specify the --handler option with the name of your handler function. Use --zip-file to reference your deployment package.

Test Your Lambda Function:

After creating your Lambda function, you can test it using the AWS Management Console, AWS CLI, or an SDK. Make sure it works as expected.

Configure Event Sources (if needed):

If your Lambda function is triggered by events from other AWS services (e.g., S3, SNS, API Gateway), configure these event sources in the AWS Management Console.

Set Up Environment Variables (if needed):

If your function relies on environment variables, configure these in the Lambda function's configuration.

Deploy Updates (if needed):

If you make changes to your function code or configuration, you can update the Lambda function by uploading a new deployment package, and the changes will be applied.

Monitor and Troubleshoot:

Use AWS CloudWatch and other monitoring tools to track the performance and behavior of your Lambda function. You can also view logs and error messages in CloudWatch Logs to troubleshoot issues.

Scale and Manage Your Function:

AWS Lambda automatically scales your function to handle incoming requests. You can configure concurrency limits, timeout settings, and other properties to manage its behavior.

Cost Management:

Keep an eye on the costs associated with your Lambda function, as you'll be billed based on the number of requests and duration of execution. Utilize cost management tools and set up billing alerts to avoid unexpected charges.

Deploying a Lambda function on AWS is a straightforward process, but it's important to pay attention to configuration, permissions, and monitoring to ensure your function operates as expected in a production environment.

Q.2 What are the deployment options for AWS Lambda?

AWS Lambda offers several deployment options, each suited for different use cases and development workflows. Here are the primary deployment options for AWS Lambda, explained in detail:

Upload Deployment Package:

This is the simplest deployment option. You create a ZIP archive that contains your function code and dependencies. Then, you manually upload it when creating or updating your Lambda function. This approach is suitable for small functions or when you need full control over your deployment process.

Steps:

Create a ZIP archive containing your function code and dependencies.

Use the AWS Management Console, AWS CLI, or AWS SDKs to create or update your Lambda function, providing the ZIP archive as the deployment package. S3 Bucket Deployment:

For larger deployment packages or for separating the deployment process from the Lambda function creation or update, you can store your deployment package in an Amazon S3 bucket. When you create or update a Lambda function, you specify the S3 bucket location for your deployment package.

Steps:

Upload your deployment package to an S3 bucket.

Use the AWS Management Console, AWS CLI, or AWS SDKs to create or update your Lambda function, specifying the S3 bucket location. AWS Serverless Application Model (SAM):

SAM is an open-source framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define the Amazon API Gateway APIs, AWS Lambda functions, and Amazon DynamoDB tables needed by your serverless application. You can define your Lambda function configurations and deployment details in a template.yaml file, which SAM uses to create and deploy the function. Steps:
Create a template.yaml file that defines your Lambda function and its dependencies.
Use the AWS SAM CLI to package and deploy your serverless application to AWS. This CLI tool simplifies packaging and deployment, including the creation of Amazon S3 buckets for deployment. Lambda Layers:

Lambda Layers allow you to separate your function code from its dependencies. You can create a custom Layer with your dependencies and then reference it in your Lambda function. When updating the dependencies, you only need to update the Layer, reducing the size and complexity of the deployment package. Steps:

Create a Lambda Layer containing your dependencies.
Reference the Layer in your Lambda function configuration.
When updating dependencies, update the Layer without changing your function code.
Container Images (AWS Lambda for Containers):

AWS Lambda added support for running functions in container images. With this option, you build a container image with your function code, dependencies, and any runtime environment you need. You can use your preferred container registry to store the image. Lambda manages the container execution, scaling, and resource allocation. Steps:

Build a Docker container image with your function code and dependencies.
Push the image to a container registry (e.g., Amazon ECR, Docker Hub).
Create a Lambda function using the image from your container registry. Continuous Deployment Tools:

You can integrate AWS Lambda deployment into your continuous deployment pipelines using tools like AWS CodePipeline, AWS CodeBuild, Jenkins, or any other CI/CD solution. This approach automates the deployment process, allowing you to push changes to your function code in a version-controlled manner. Steps:

Set up a CI/CD pipeline that monitors your code repository.
Configure the pipeline to build and deploy your Lambda function when changes are detected.
Each of these deployment options has its own advantages and is suitable for different scenarios. The choice depends on factors such as the size and complexity of your

function, your development workflow, and whether you require more granular control over your deployments.

Q.3 What are the 3 full deployment modes that can be used for AWS?

In the context of AWS, there are three primary full deployment modes, each offering a distinct approach to deploying and managing applications. These modes are designed to accommodate different use cases and operational requirements. Let's explore each of them in detail:

EC2-Based Deployment:

Description:
EC2-based deployment is the traditional deployment mode in AWS where you provision and manage virtual machines (EC2 instances) to run your applications. In this mode, you have full control over the underlying infrastructure, including the choice of instance types, operating systems, and configuration. This mode is ideal for applications that require a high degree of customization or when you have legacy systems that need to run in a traditional virtualized environment.

Use Cases:
Running legacy applications or software that can't be containerized.
Applications with complex network configurations or specific hardware requirements.
When you need to manage the entire stack from the operating system up.

Key Components:
Amazon Elastic Compute Cloud (EC2) instances.
Amazon Virtual Private Cloud (VPC) for network isolation.
Elastic Load Balancers for distributing traffic.
Amazon RDS or other database services for data storage.

Benefits:
Complete control over infrastructure.
Compatibility with a wide range of software. Ability to run non-containerized or legacy applications.

Challenges:
Manual scaling and management of EC2 instances.

More operational overhead compared to serverless or container-based solutions.
Limited automation compared to other deployment modes.

Serverless Deployment:

Description:

Serverless deployment is a modern cloud computing paradigm in which you focus solely on writing code (usually in the form of functions) and let the cloud provider manage all the underlying infrastructure. AWS Lambda is a key component of this approach, allowing you to run code in response to events without provisioning or managing servers. Serverless computing is highly scalable and event-driven.

Use Cases:

Building scalable, event-driven applications.
Microservices architecture.
Real-time data processing and analysis.
Reducing operational overhead by offloading infrastructure management.

Key Components:

AWS Lambda for running code in response to events.
Amazon API Gateway for exposing APIs.
Various AWS services for data storage and processing. Amazon EventBridge or Amazon S3 event triggers for event-driven applications.

Benefits:

Auto-scaling and high availability.
Minimal operational overhead.
Pay-per-use pricing.
Easy integration with other AWS services.

Challenges:

Stateless execution, which may require workarounds for stateful applications.
Limited runtime options compared to EC2 instances. Function duration and resource limits.

Container-Based Deployment:

Description:

Container-based deployment leverages containerization technology to package applications and their dependencies into a consistent and portable format. AWS offers Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS) for managing containers in a scalable, automated, and highly available manner.

This deployment mode is ideal for containerized applications, microservices, and orchestrating container workloads.

Use Cases:
Microservices architecture.
Porting and running containerized applications.
Managing applications that need to scale and have dependencies isolated in containers.

Key Components:
Amazon ECS or Amazon EKS for managing containers.
Amazon ECR for container registry.
Docker for building and running containers.
Kubernetes for container orchestration (EKS).
Benefits:
Scalability and flexibility of containerization.
Portability and consistency of containers.
Advanced orchestration and management features in Kubernetes.

Challenges:
Container management complexity.
Learning curve for orchestrators like Kubernetes. Ongoing operational overhead.

These three full deployment modes represent different approaches to deploying and managing applications in AWS. Your choice should be based on your specific requirements, including the nature of your applications, your scalability needs, and your desired level of operational control. It's not uncommon for organizations to use a combination of these deployment modes, depending on their application portfolio and use cases.

Q.4 What are the 3 components of AWS Lambda?

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run code in response to events without the need to manage servers. AWS Lambda has three core components that work together to enable serverless compute capabilities:

Lambda Function:

Description: A Lambda function is the core unit of execution in AWS Lambda. It represents your code, which can be written in various programming languages such as Node.js, Python, Java, Go, and more. A Lambda function is a small, self-contained piece of code that can perform a specific task when triggered by an event. It can be as simple as a few lines of code or more complex, and it typically follows a specific structure, including a handler function that AWS Lambda invokes when an event occurs.

Use Cases: Lambda functions are used for a wide range of purposes, including data processing, automation, real-time file processing, API endpoints, and more. They are particularly well-suited for building serverless applications and microservices.

Key Characteristics:

Small, single-purpose code.
Stateless (no persistent storage of data between invocations).
Event-driven execution.
Automatic scaling and resource allocation.

Event Source:

Description: Event sources are triggers that initiate the execution of a Lambda function. These sources can be various AWS services or external systems that generate events. When an event occurs, AWS Lambda is automatically invoked, and the event data is passed to the Lambda function for processing. AWS Lambda supports a wide range of event sources, including AWS services like Amazon S3, Amazon DynamoDB, AWS SNS, and custom event sources using AWS Step Functions.

Use Cases: Event sources enable Lambda functions to respond to changes in data, incoming requests, system events, and more. This makes them suitable for building event-driven applications and automating workflows.

Key Characteristics:

Diverse sources, including AWS services and custom events.
Real-time event triggering.
Integration with various AWS services.

Execution Environment:

Description: The execution environment is the runtime environment where Lambda functions run. AWS Lambda manages and provisions these environments dynamically as needed, and it abstracts the underlying infrastructure from developers. The execution

environment includes the compute resources (CPU, memory) and the network configuration necessary for the function's execution. The environment automatically scales with incoming event load.

Use Cases: The execution environment is responsible for ensuring that Lambda functions can run in a scalable and highly available manner without the need for manual provisioning or management. It enables the on-demand execution of code.

Key Characteristics:

Automatic provisioning and scaling.
Abstracts infrastructure management.
Resource allocation (memory and CPU) defined per function. Isolation between concurrent executions.

Together, these three components form the foundation of AWS Lambda. A Lambda function processes events from various event sources within an execution environment provided by AWS Lambda. The serverless nature of Lambda, where you focus on code rather than infrastructure, allows you to build applications that are highly scalable and responsive to real-time events with minimal operational overhead. This serverless model is particularly valuable for organizations looking to optimize resource utilization and reduce infrastructure management complexities.