Thadomal Shahani Engineering College

Bandra (W.), Mumbai - 400 050.

CERTIFICATE

Certify that Mr./Miss ANIMESH NARAYAN PARAB

of I-T Department, Semester V with

Roll No. 88 has completed a course of the necessary

experiments in the subject Advance Devops Lab under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In- Charge

Head of the Department

Date 21/10/23

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
1	To understand the benefits of cloud infrastructure and setup Aws cloud IDE, Launch AWS cloud 9 IDE		19-7-23	
2	To Build your application using Aws codebuild and deploy on S3		19-7-23	
3	To understand the kubernets cluster Architecture install and spin up a cluster on linux		26-7-23	
4	To install kubectl and execute kubectl command to manage the kubernets cluster		2-8-23	
5	To understand terraform lifecycle core concepts and install it or Linux Machine		11-10-23	
6	To build change and destroy Aws / GCP Azure digital ocean		23-8-23	
7	To understand static Analysis SAST process and learn to integrate to sonarqube cube		13-9-23	
8	To create a lambad function which will log 'An Image has been Added'		2-8-23	
9	To understand continous monitoring and configuration of Nagious on Linux		27-9-23	

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	PAGE NO.	DATE	TEACHERS SIGN.
10	To understand AWS Lamda it works flow various functions and create function		07-9-23	
11	Assignment 1		26-7-23	
12	Assignment 2		11-10-23	

Animesh Parab 88 T2-T21

Assignment 1

Aim:- Study and create an AWS EC2 instance.

Lop mapped:-
LO1- To understand the fundamentals of Cloud Computing to be fully proficient with Cloud-based DevOps solution deployment options to meet your business requirements.

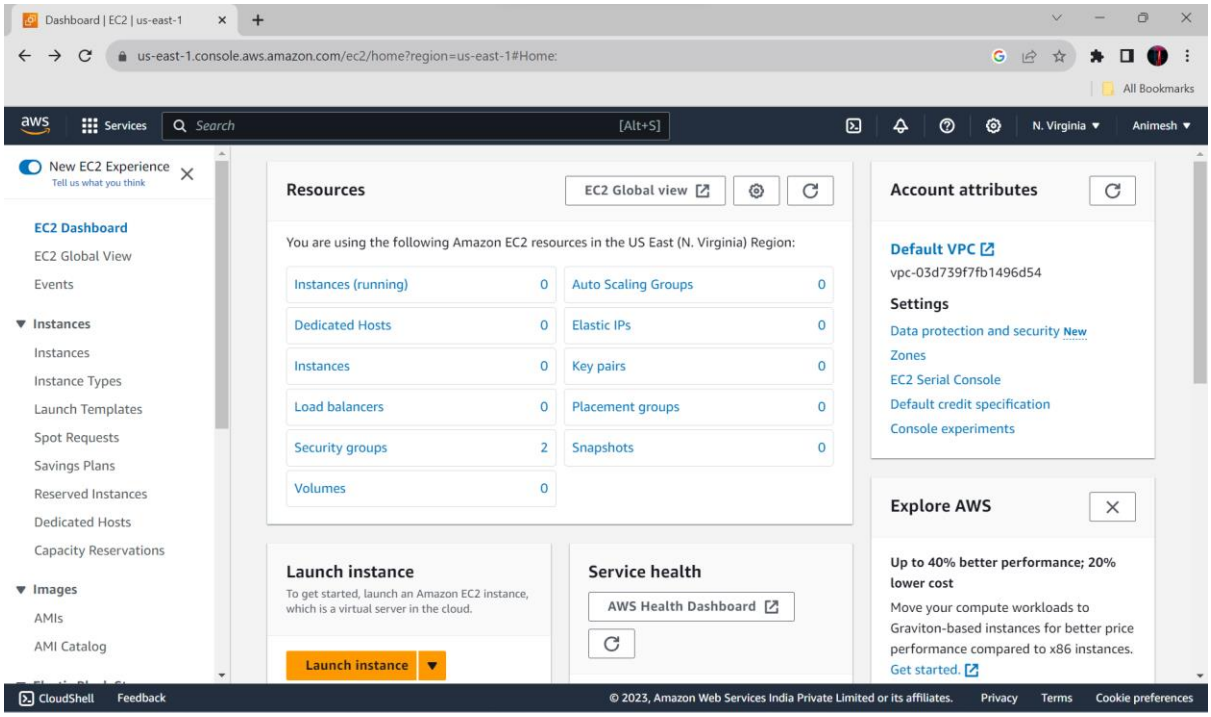
Theory:-
Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable computing capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.
Here's a detailed explanation of an AWS EC2 instance:
1. Virtual Server:
An EC2 instance is a virtualized computing environment that mimics a physical server. It runs an operating system of your choice and supports a wide range of applications, databases, and services.
2. Scalability:
One of the key benefits of EC2 is its scalability. You can easily scale your instances horizontally (adding more instances) or vertically (resizing an instance) to meet changing workloads. This elasticity helps you optimize performance while controlling costs.
3. Variety of Instance Types:
EC2 provides a variety of instance types optimized for different use cases. These instance types vary in terms of CPU, memory, storage, and network capabilities, allowing you to choose the best fit for your application's requirements.

4. Amazon Machine Image (AMI):
An AMI is a pre-configured template that contains the information required to launch an instance. It includes the operating system, application software, and any additional configuration you've applied. You can choose from a wide range of public AMIs or create your own custom AMIs.
5. Networking and Security:
Each EC2 instance is associated with a security group, which acts as a virtual firewall controlling inbound and outbound traffic. You can also assign Elastic IP addresses for consistent IP assignments, create Virtual Private Clouds (VPCs) for isolated networking, and configure network settings like subnets, routing tables, and network access control lists.
6. Cost Flexibility:
EC2 instances are available in different pricing models, including On-Demand (pay-as-you-go), Reserved Instances (upfront payment for long-term usage), and Spot Instances (bid-based pricing). This allows you to choose the most cost-effective option for your workload.

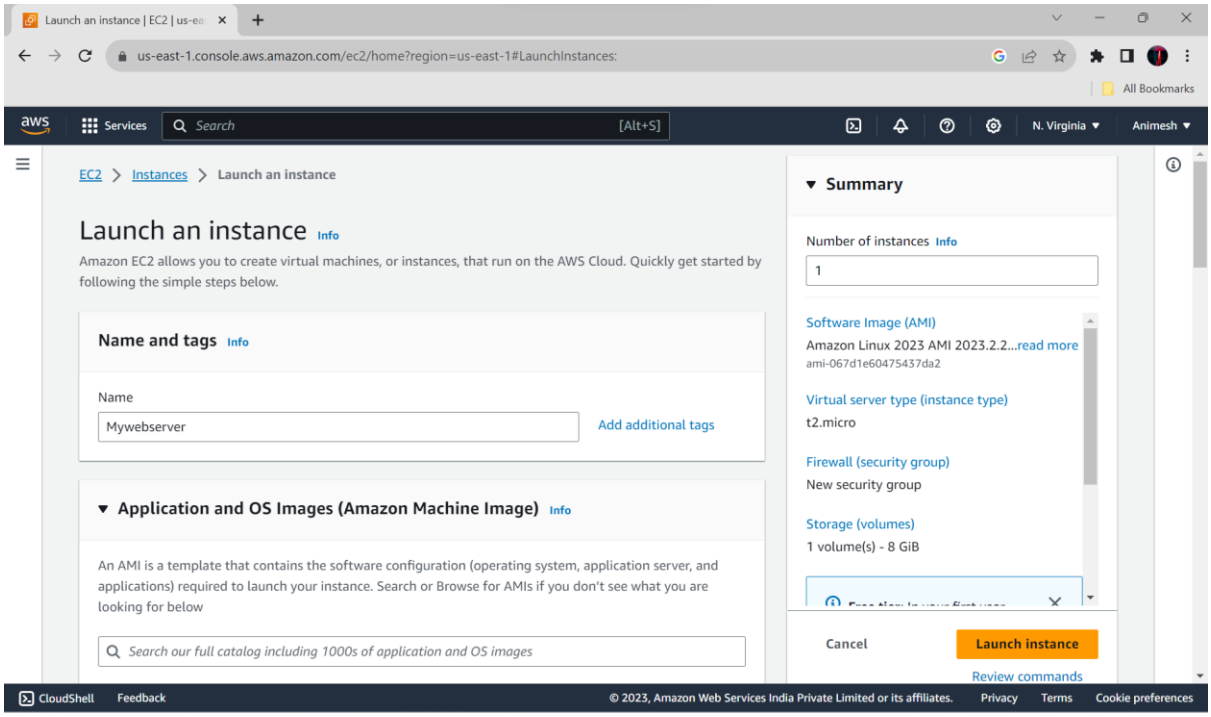
Output:-

LOGIN TO AWS ACCOUNT,

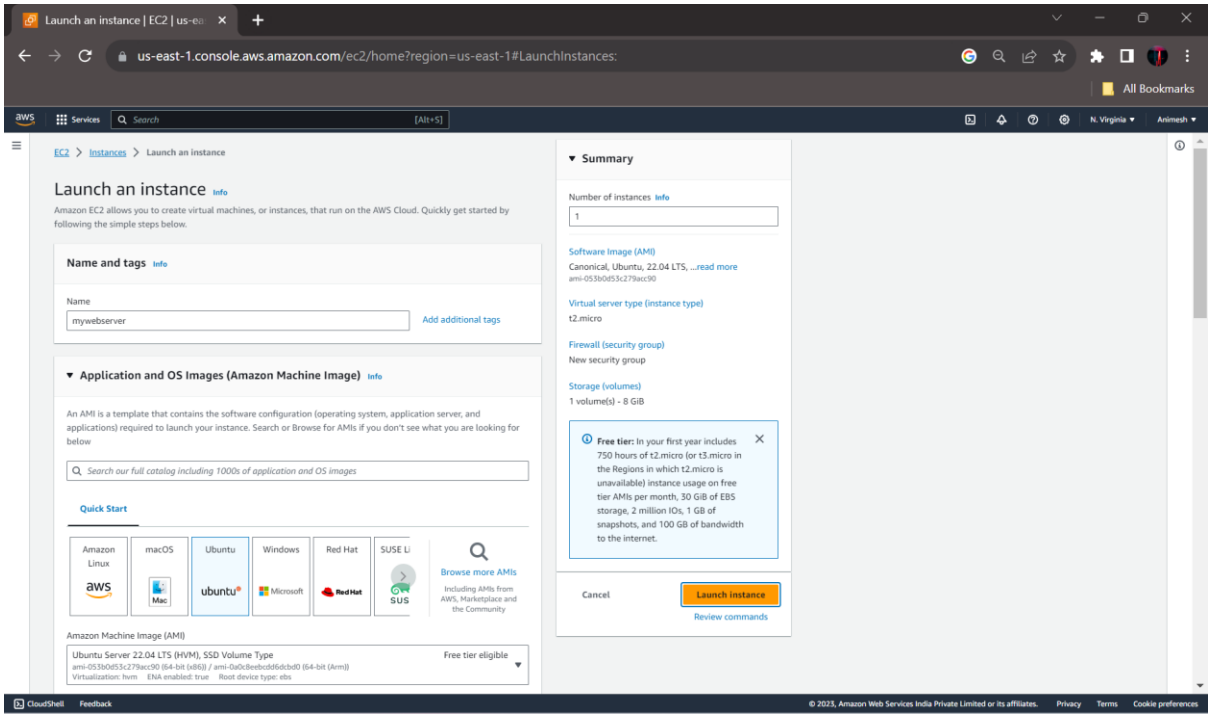
THEN SEARCH EC2.

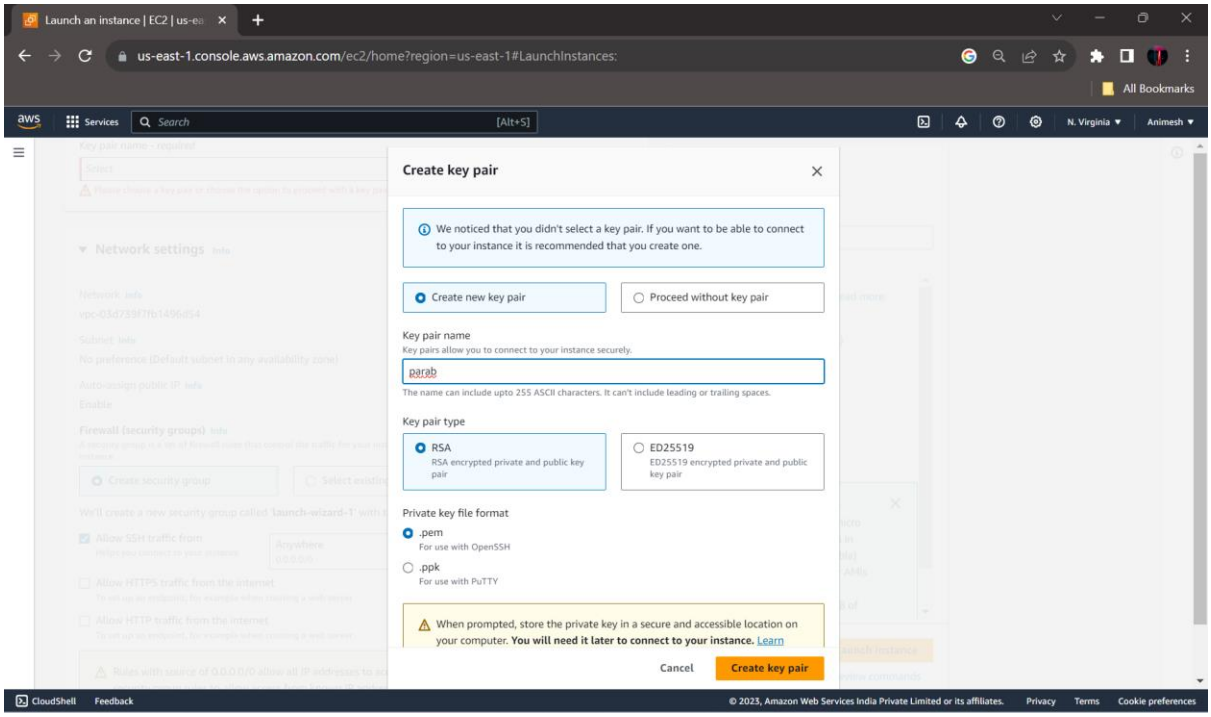


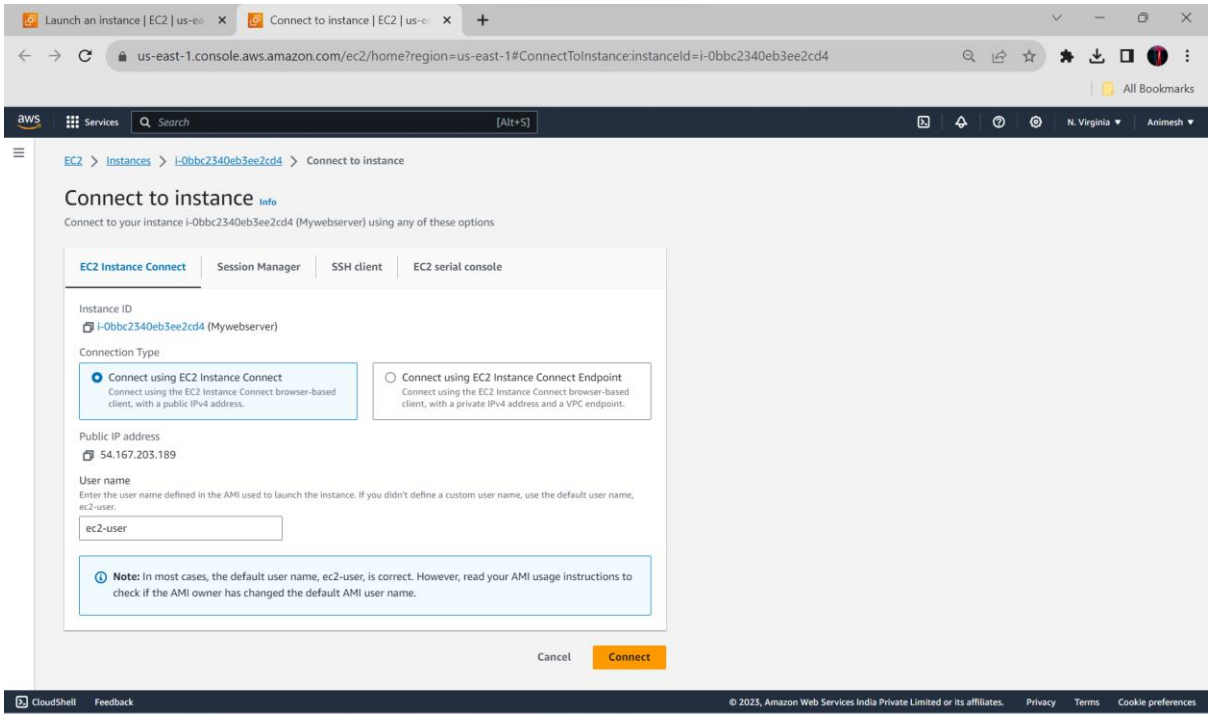
NOW CLICK ON LAUNCH / CREATE NEW INSTANCES.

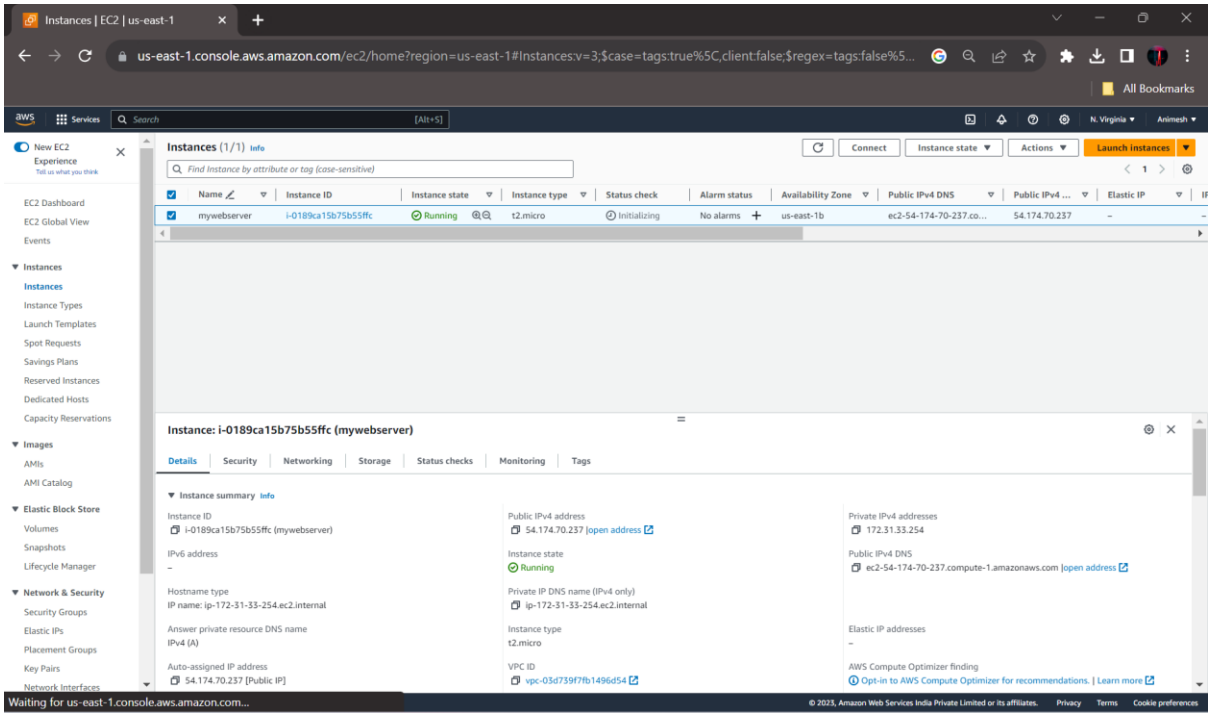


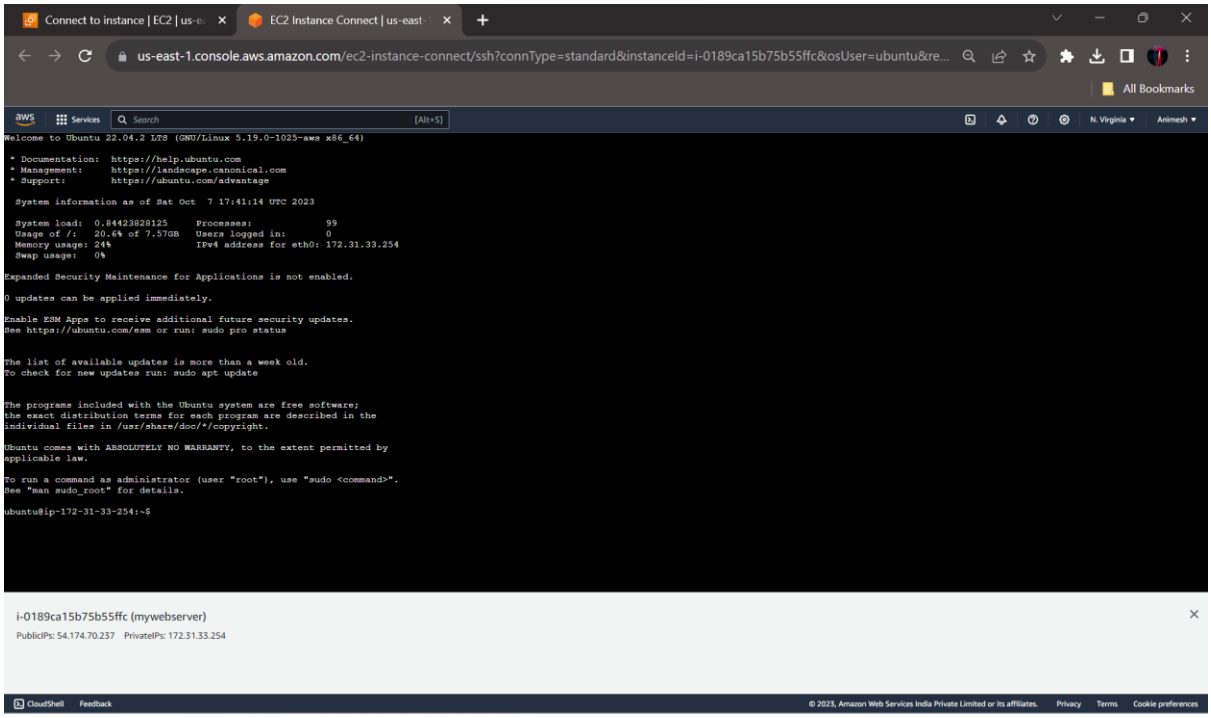
Choose any machine you want to create here I am creating UBUNTU(free tier).




Launch an instance | EC2 | us-ea
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
All Bookmarks
Services
Search
[Alt+S]
N. Virginia
Animesh
Create key pair
We noticed that you didn't select a key pair. If you want to be able to connect to your instance it is recommended that you create one.
Create new key pair
Proceed without key pair
Key pair name
Key pairs allow you to connect to your instance securely.
parab
The name can include upto 255 ASCII characters. It can't include leading or trailing spaces.
Key pair type
RSA
RSA encrypted private and public key pair
ED25519
ED25519 encrypted private and public key pair
Private key file format
.pem
For use with OpenSSH
.ppk
For use with PuTTY
When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. Learn
Cancel
Create key pair
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences


Launch an instance | EC2 | us-ea
Connect to instance | EC2 | us-e
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-0bbc2340eb3ee2cd4
All Bookmarks
Services
Search
[Alt+S]
N. Virginia
Animesh
EC2 > Instances > i-0bbc2340eb3ee2cd4 > Connect to instance
Connect to instance Info
Connect to your instance i-0bbc2340eb3ee2cd4 (Mywebserver) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Instance ID
i-0bbc2340eb3ee2cd4 (Mywebserver)
Connection Type
Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 address.
Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IP address
54.167.203.189
User name
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ec2-user.
ec2-user
Note: In most cases, the default user name, ec2-user, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.
Cancel
Connect
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences





Conclusion:- In this, I have created an Amazon web services account and after creating the account I created instances successfully as shown above also I have run a virtual Unix environment on the created instance and lab outcome is also achieved.

Animesh Parab T2-T21 88

ASSIGNMENT 2 CLOUD9

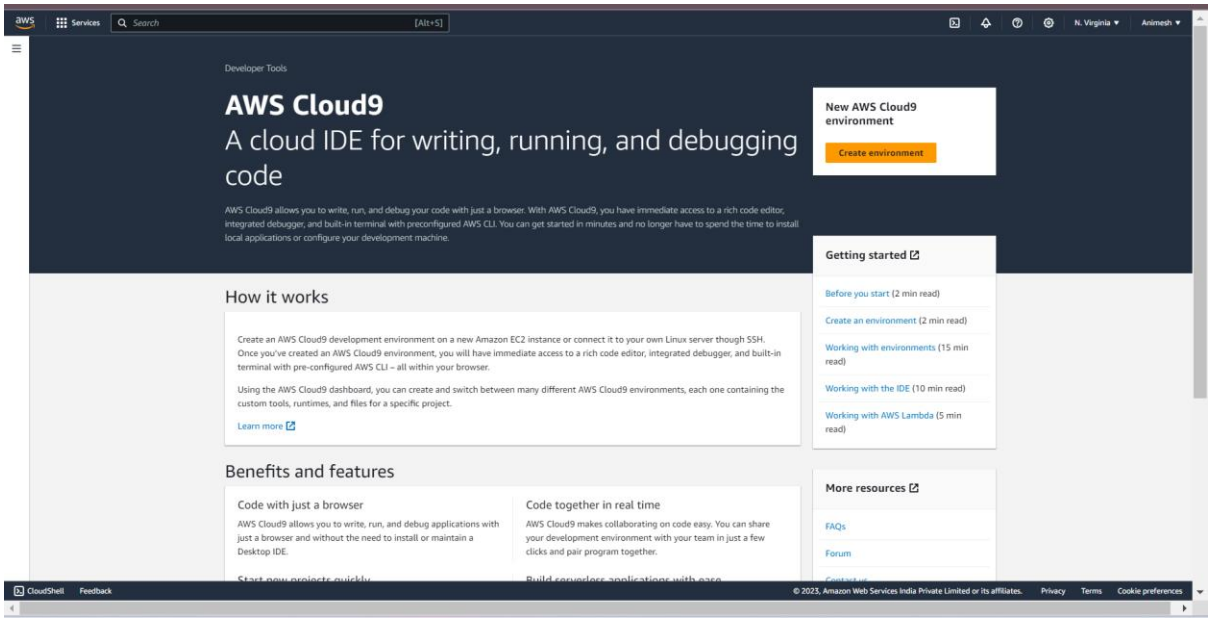
AIM- To create aws Cloud9 IDE account and execute python application development.

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-

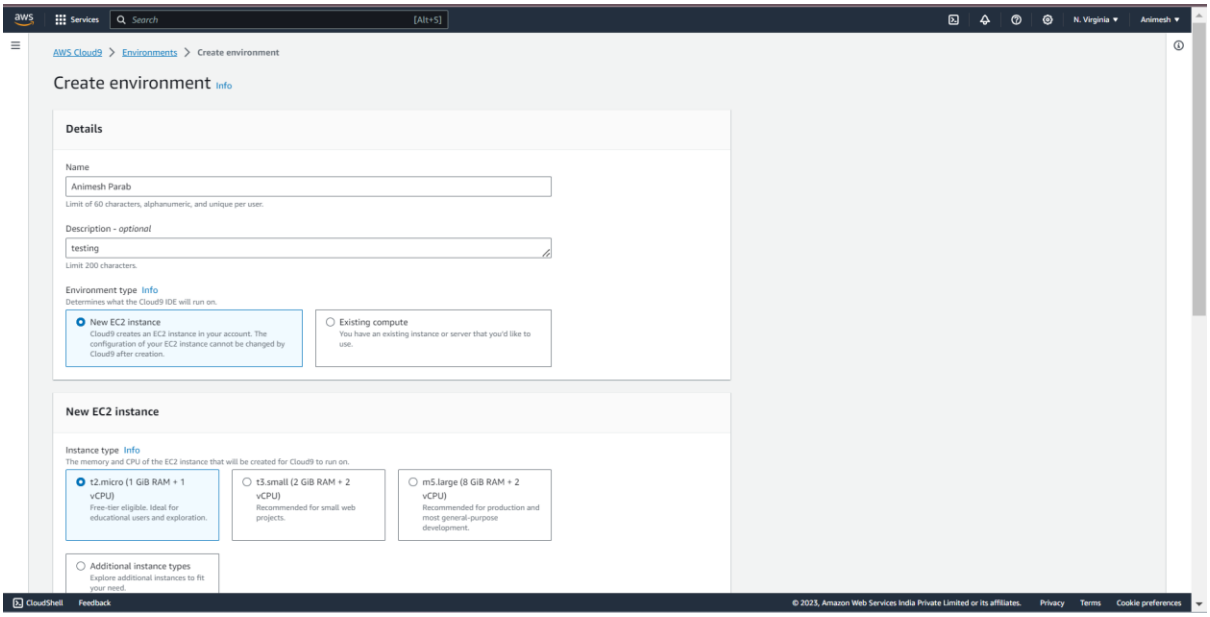
LOG IN TO YOUR AWS ACCOUNT,

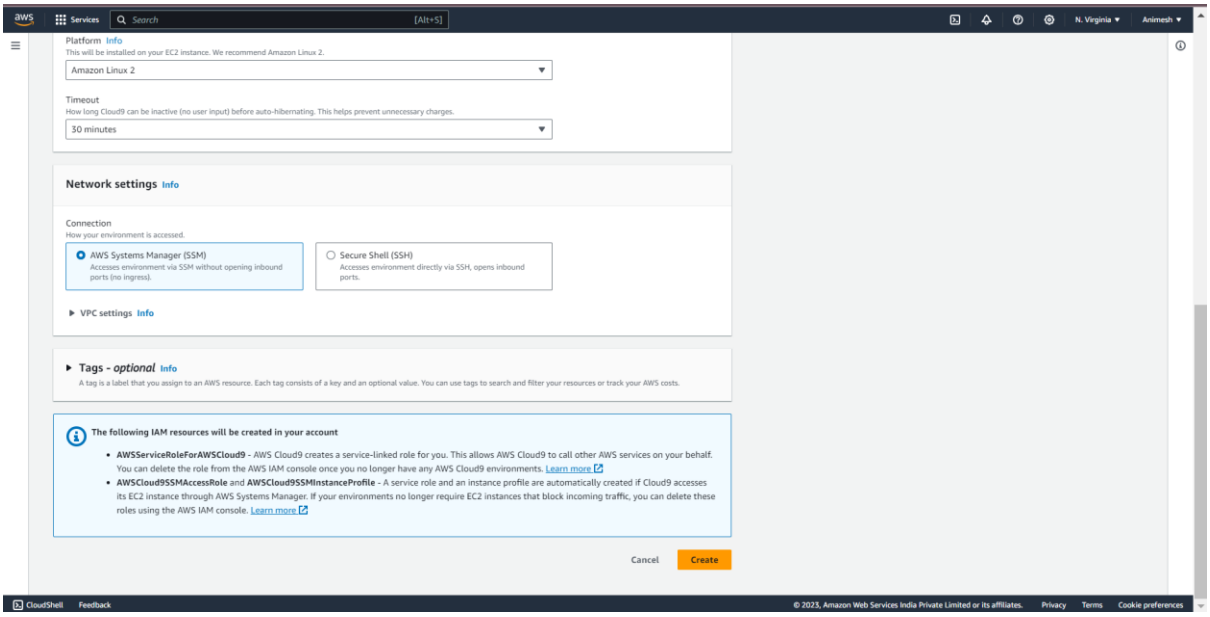
SEARCH FOR CLOUD 9 IN THE SEARCH BAR

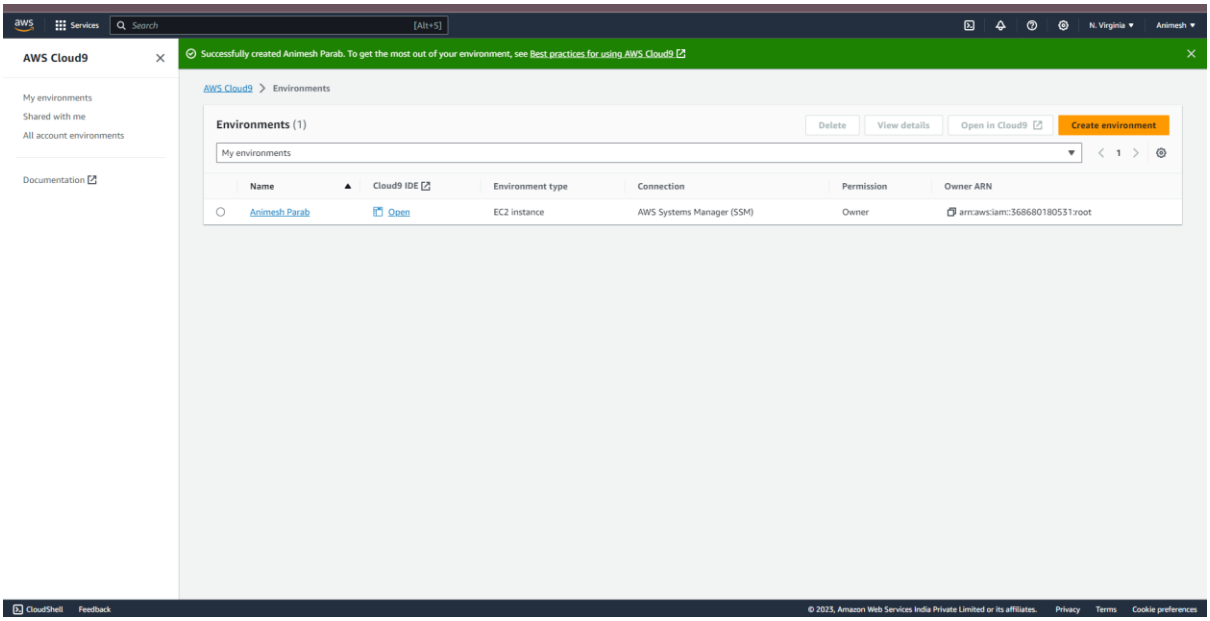


CLICK ON CREATE ENVIRONMENT,

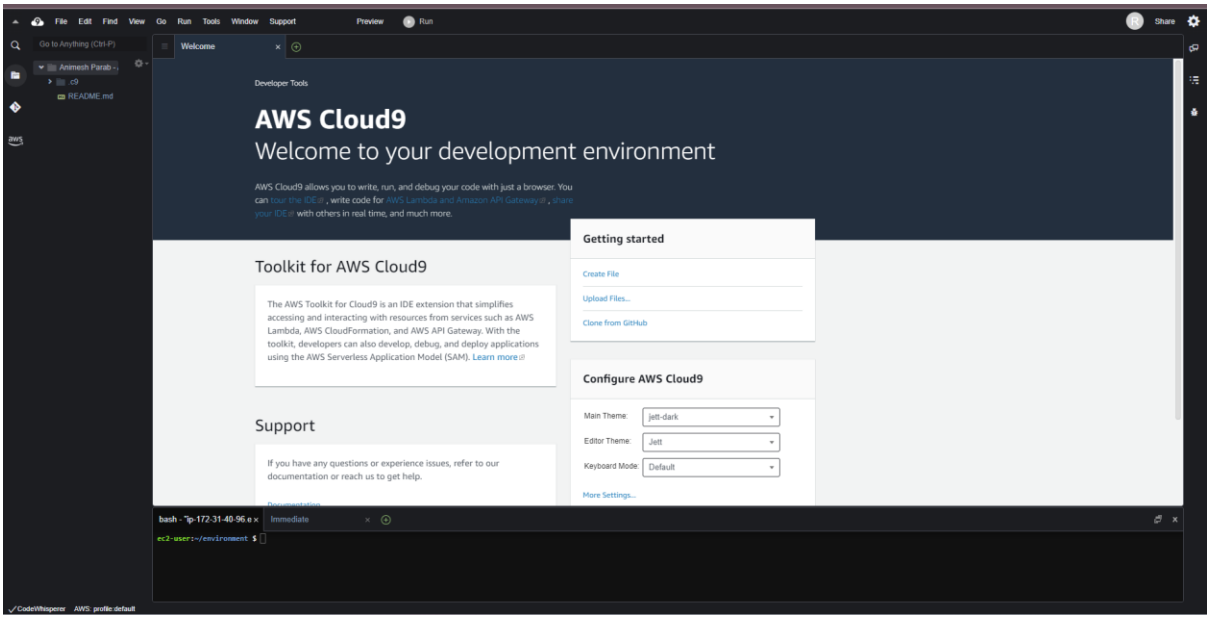
NAME THE ENVIRONMENT

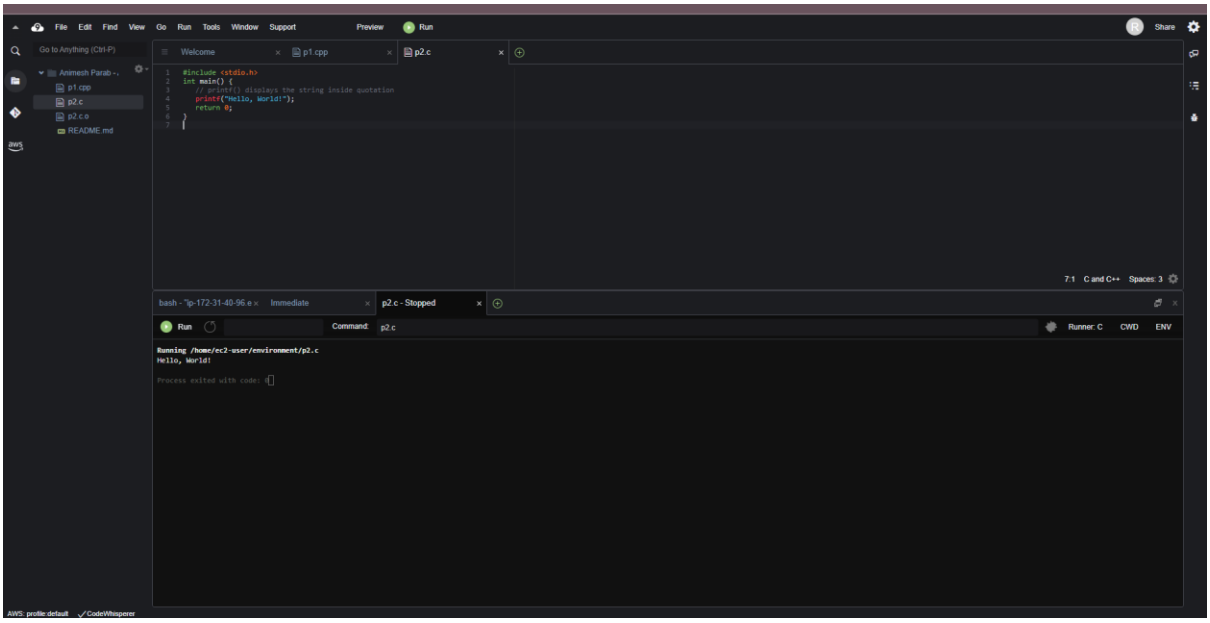

aws
Services
Search
[Alt+S]
N. Virginia
Animesh
AWS Cloud9 > Environments > Create environment
Create environment Info
Details
Name
Animesh Parab
Limit of 60 characters, alphanumeric, and unique per user.
Description - optional
testing
Limit 200 characters.
Environment type Info
Determines what the Cloud9 IDE will run on.
New EC2 instance
Cloud9 creates an EC2 instance in your account. The configuration of your EC2 instance cannot be changed by Cloud9 after creation.
Existing compute
You have an existing instance or server that you'd like to use.
New EC2 instance
Instance type Info
The memory and CPU of the EC2 instance that will be created for Cloud9 to run on.
t2.micro (1 GiB RAM + 1 vCPU)
Free-tier eligible. Ideal for educational users and exploration.
t3.small (2 GiB RAM + 2 vCPU)
Recommended for small web projects.
m5.large (8 GiB RAM + 2 vCPU)
Recommended for production and most general-purpose development.
Additional instance types
Explore additional instances to fit your need.
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences

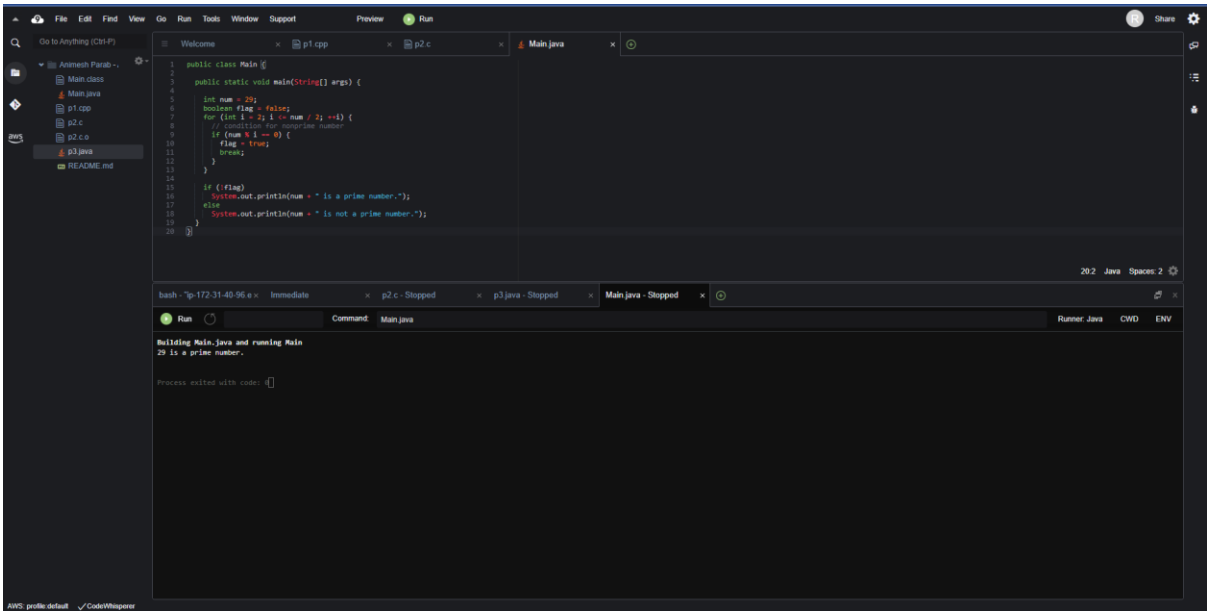

aws
Services
Search
[Alt+S]
N. Virginia
Animesh
Platform Info
This will be installed on your EC2 instance. We recommend Amazon Linux 2.
Amazon Linux 2
Timeout
How long Cloud9 can be inactive (no user input) before auto-hibernating. This helps prevent unnecessary charges.
30 minutes
Network settings Info
Connection
How your environment is accessed.
AWS Systems Manager (SSM)
Accesses environment via SSM without opening inbound ports (no ingress).
Secure Shell (SSH)
Accesses environment directly via SSH, opens inbound ports.
VPC settings Info
Tags - optional Info
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
The following IAM resources will be created in your account
AWSServiceRoleForAWSCloud9 - AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. Learn more
AWSCloud9SSMAccessRole and AWSCloud9SSMInstanceProfile - A service role and an instance profile are automatically created if Cloud9 accesses its EC2 instance through AWS Systems Manager. If your environments no longer require EC2 instances that block incoming traffic, you can delete these roles using the AWS IAM console. Learn more
Cancel
Create
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences



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

Animesh Parab T2-T21 88

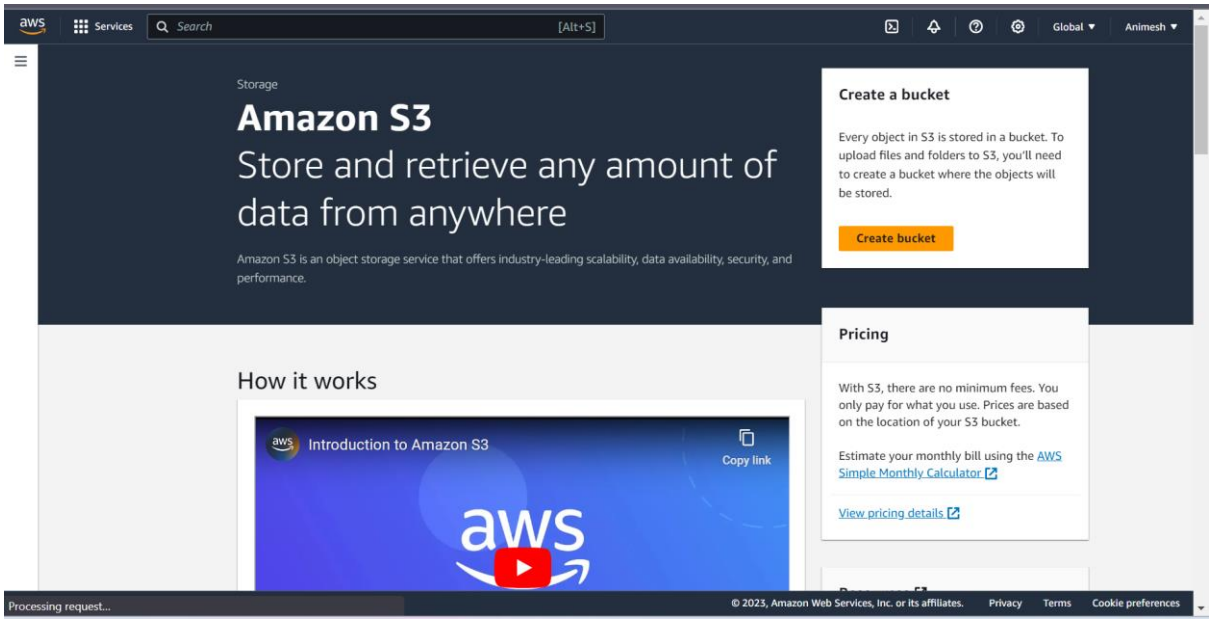
ASSIGNMENT-3

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.

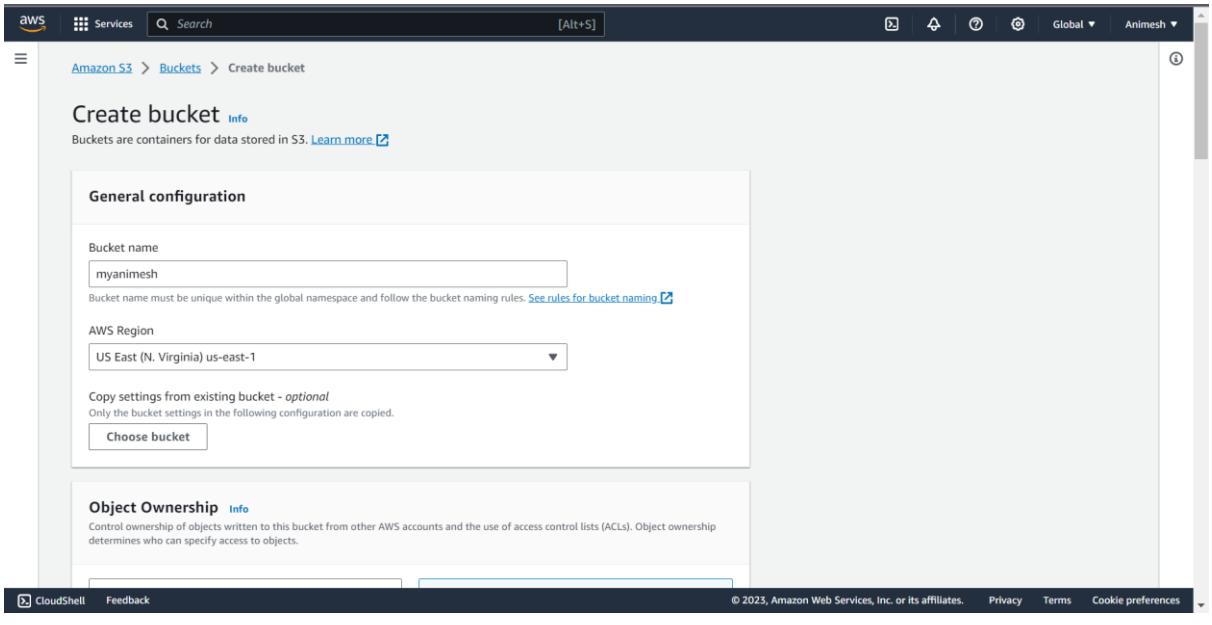
STEPS:
Step 1: Creating a Bucket
1. First, we have to launch our S3 instance. Follow these steps for creating a Bucket

2. Open the Amazon S3 console by logging into the AWS Management Console at

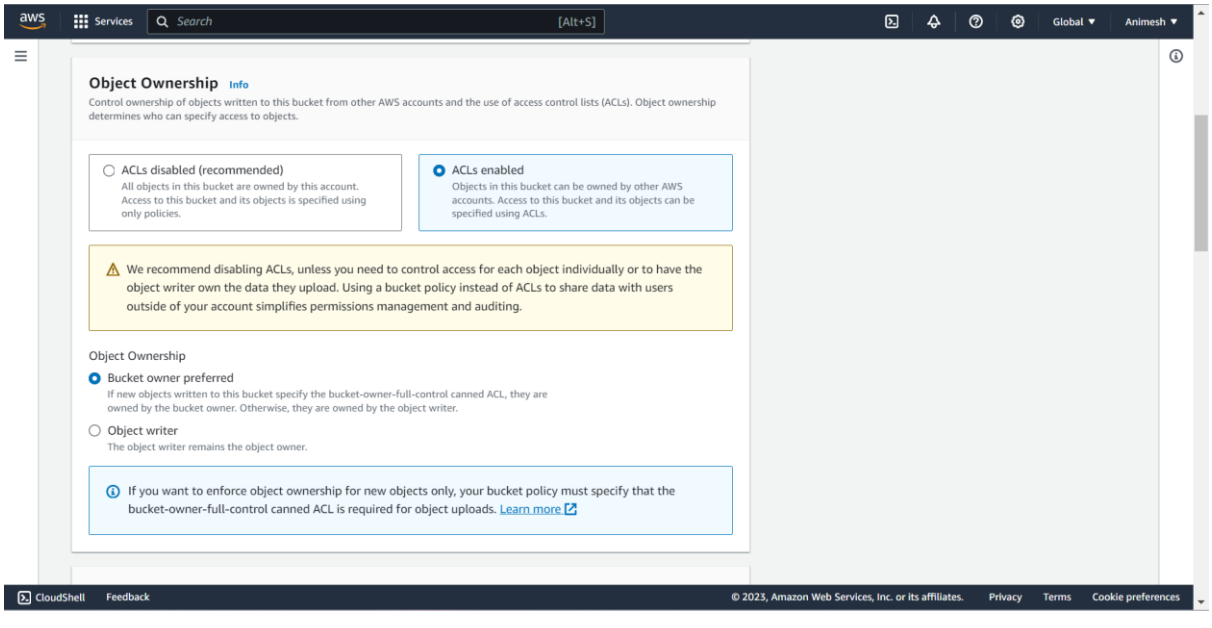


Step 2: Block Public Access settings for the bucket

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

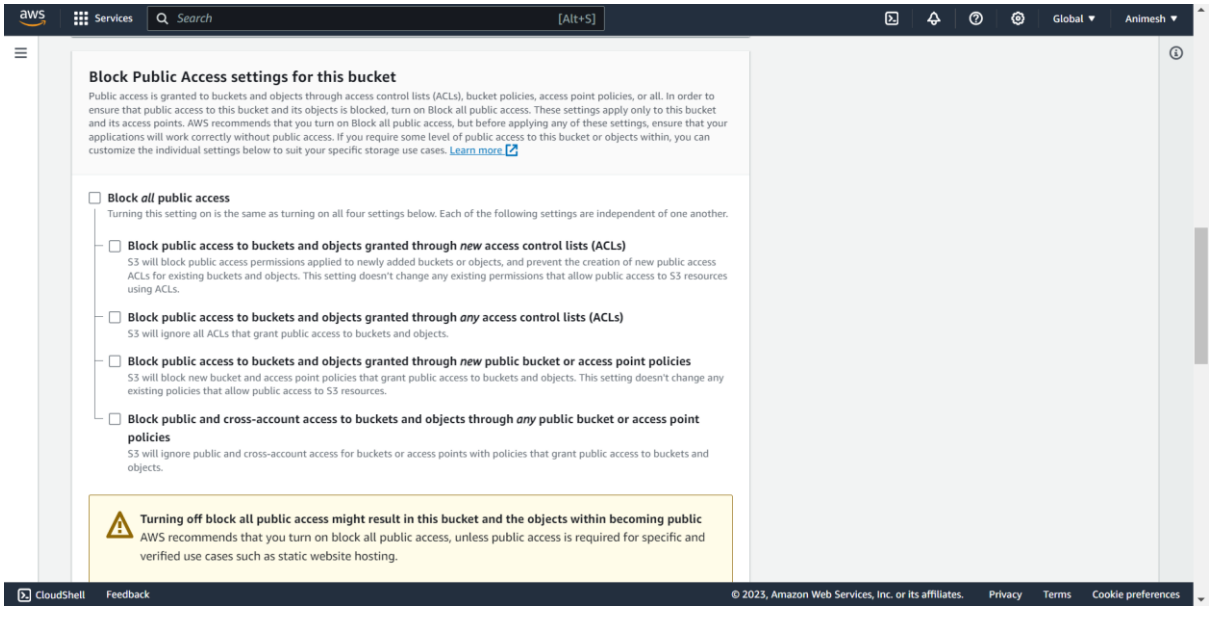


4. Choose Bucket Name – Bucket Name Should be Unique
5. Object Ownership – Enable for making Public, Otherwise disable

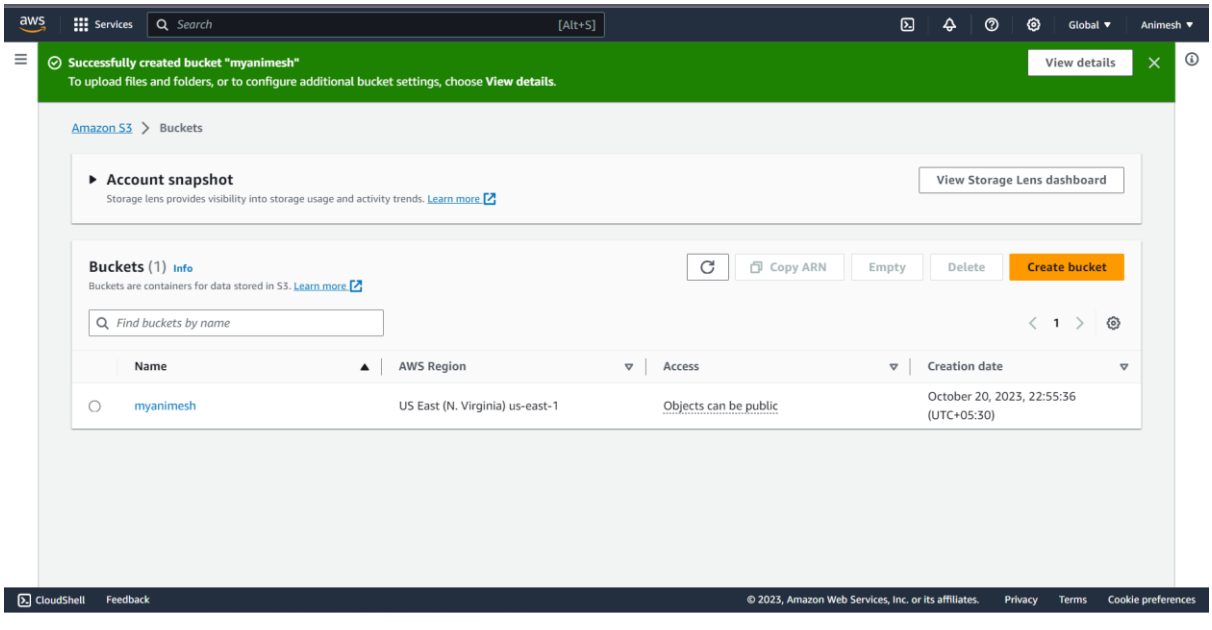


Step 2: Block Public Access settings for the bucket

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



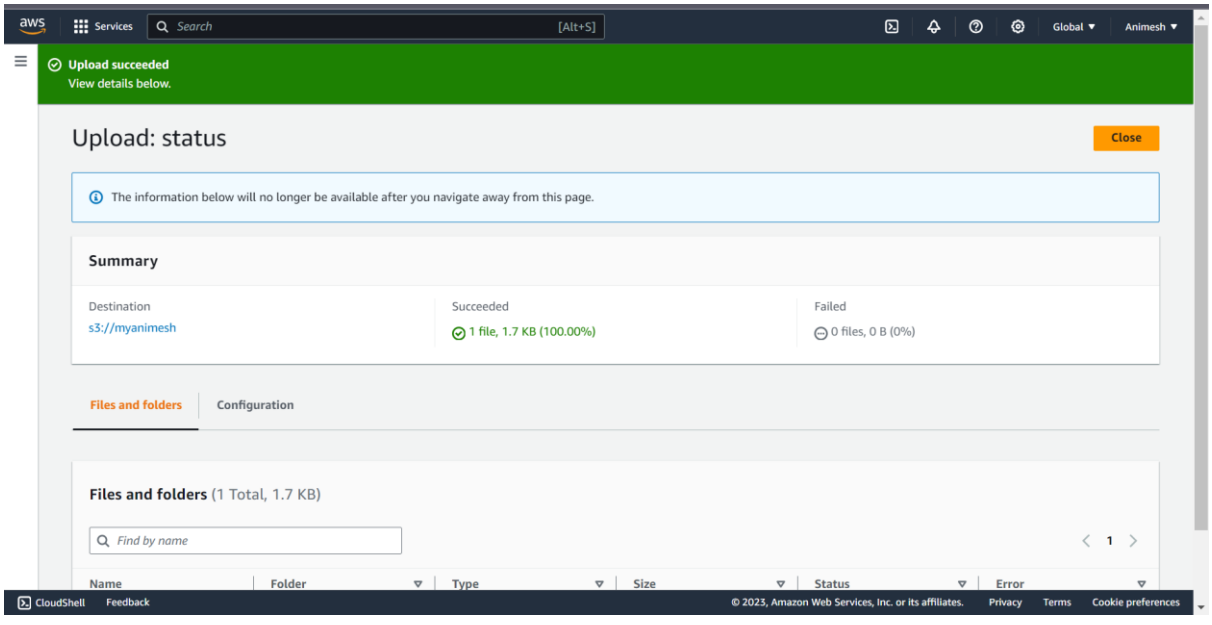
2. Now click on create bucket
Select Bucket and Click your Bucket Name.
3. Bucket is created



Step 3: Now upload code files

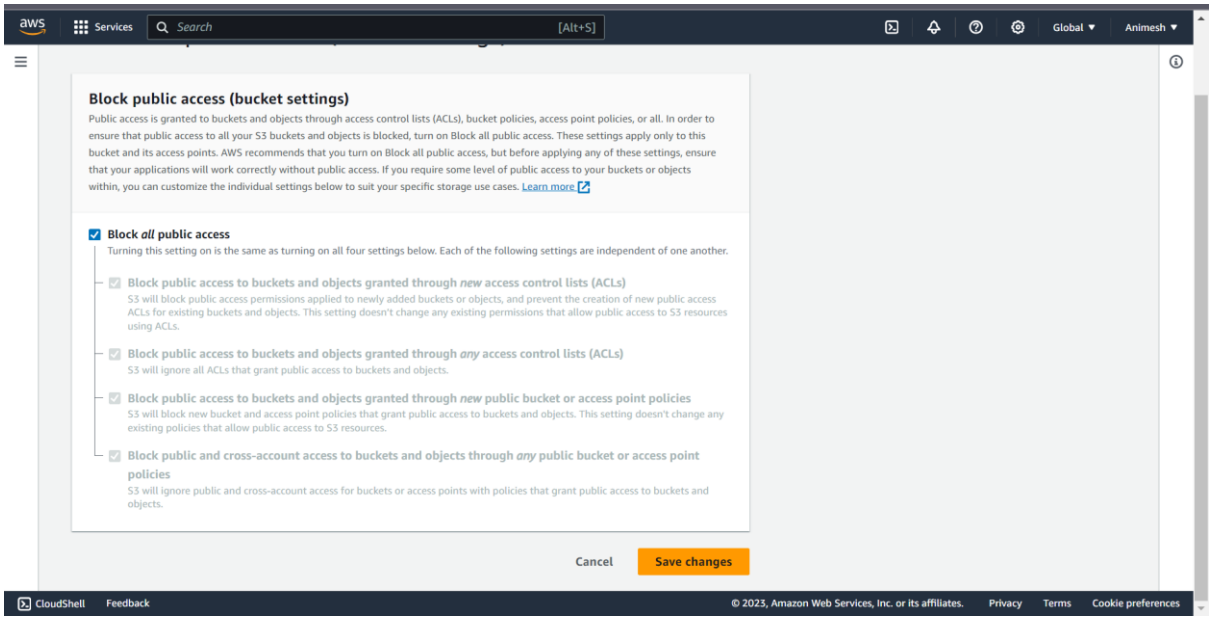
Select Bucket and Click your Bucket Name.

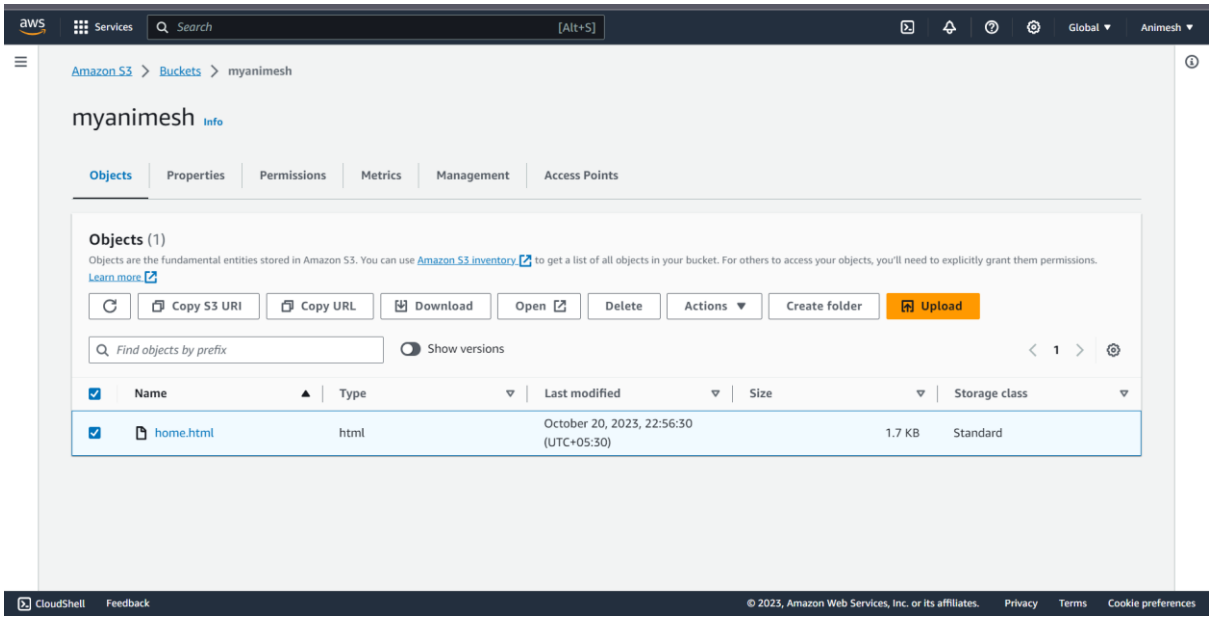
Now, click on upload (then click add File/folder) and select your HTML code file from your PC/Laptop.



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

- Block public access
- Object Ownership
- Make public Object





Step 5: Copy your Object URL
Now, click on your HTML File Object Name.
Copy the Object URL.
Step 6
Check out your Website!

CONCLUSION This experiment demonstrated how to
utilize AWS S3, a powerful andcloud storage solution, to host a static web application. S3's ability to serve Directly Paste this URL into the Other Tab or your other System. 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 costeffective and efficient solution for hosting various types of web applications, contributing to the agility and scalability of web development projects.

Animesh Parab T2-T21 88

ASSIGNMENT-4

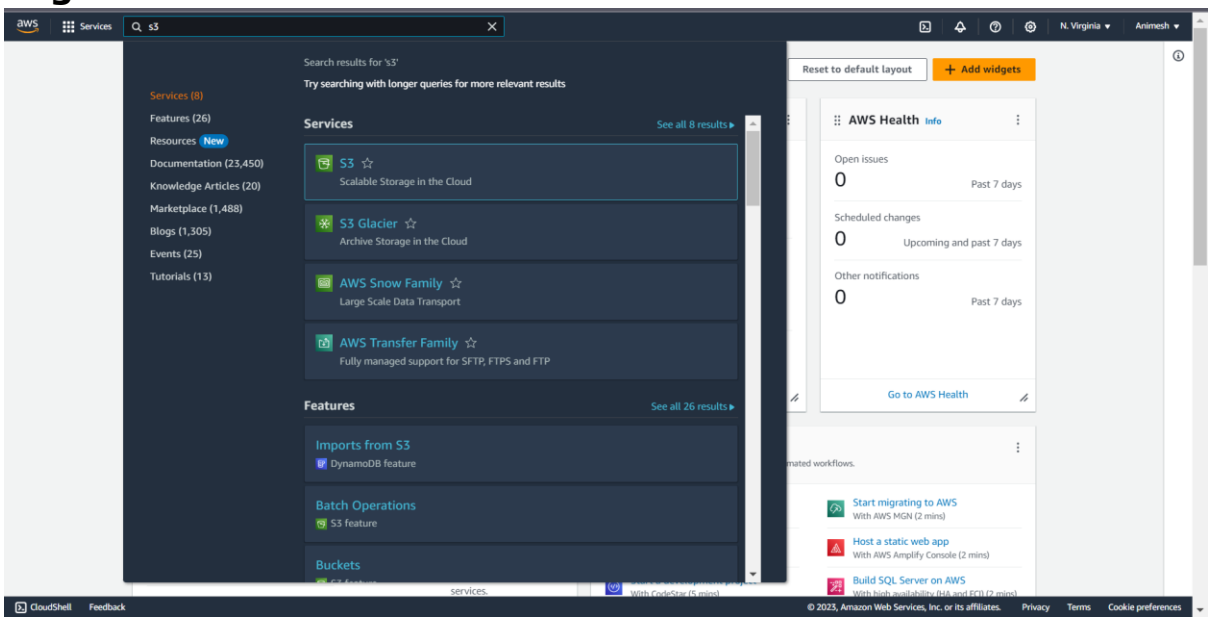
AIM: To study AWS CodePipeline and deploy web application using Code

LO MAPPED: LO1 , LO2

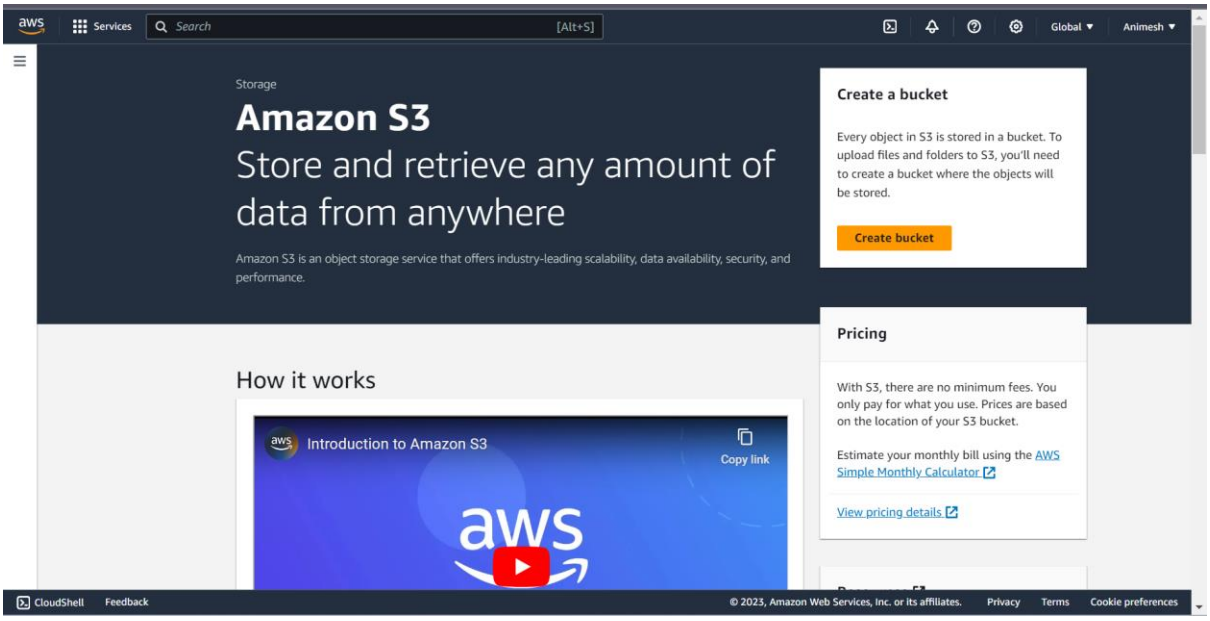
THEORY:
Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native 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:

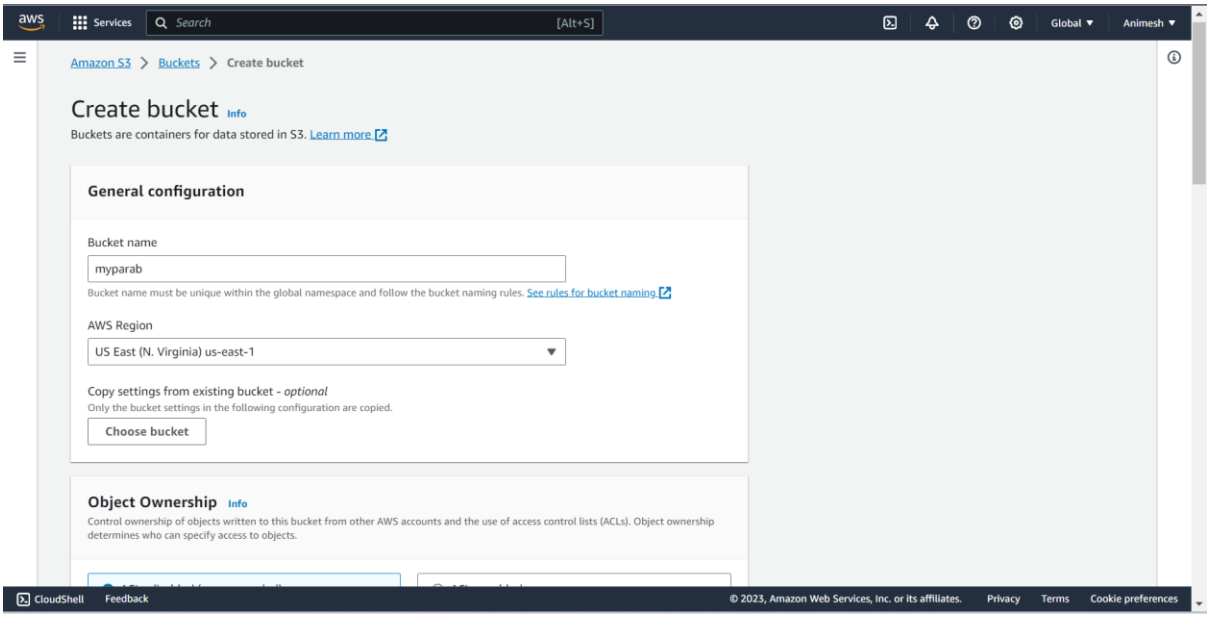
1. **Log in as IAM User. Search for S3 in console**

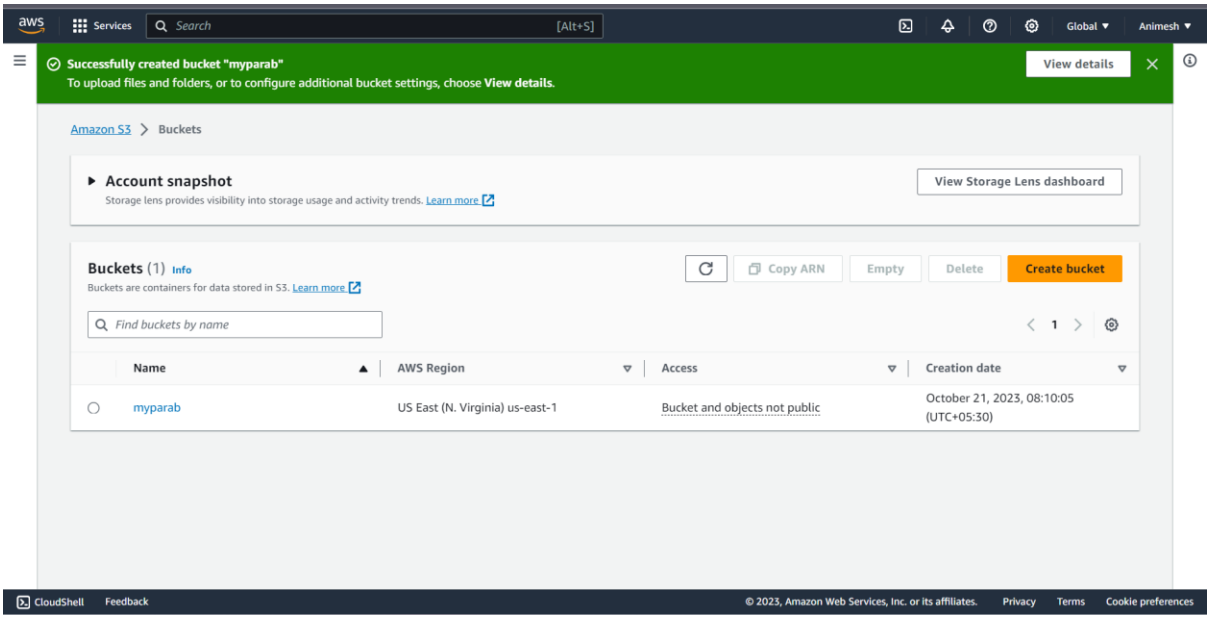


2. **Create a new bucket by clicking Create Bucket Button**

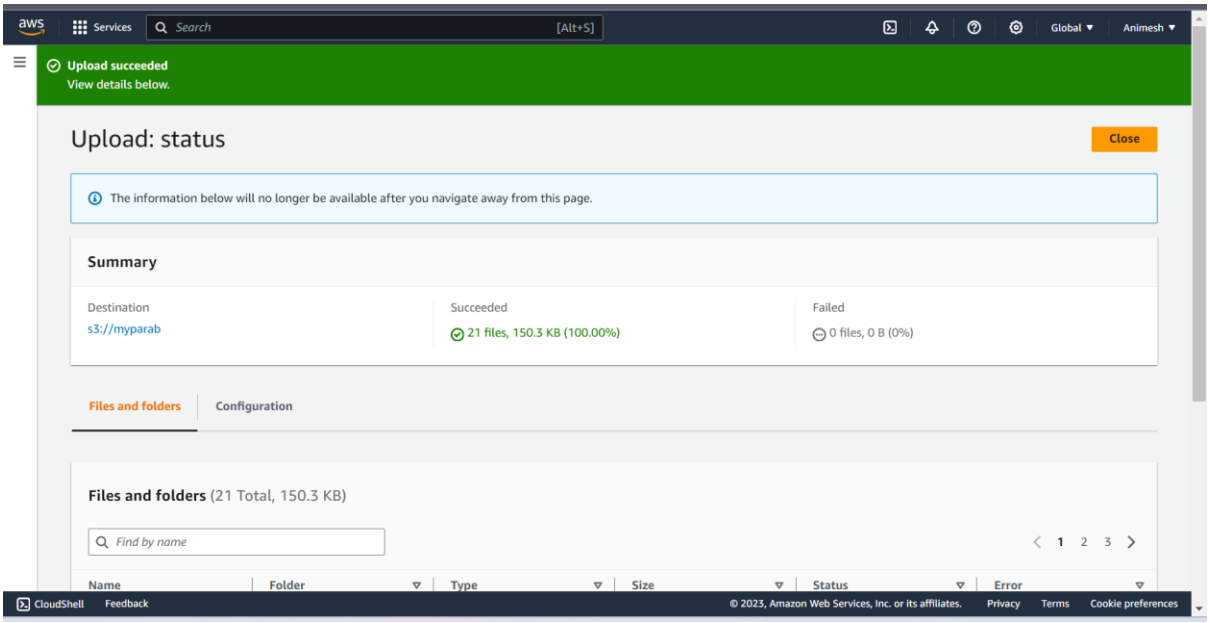


3. Give the bucket a name and click on Create Bucket to make a new bucket

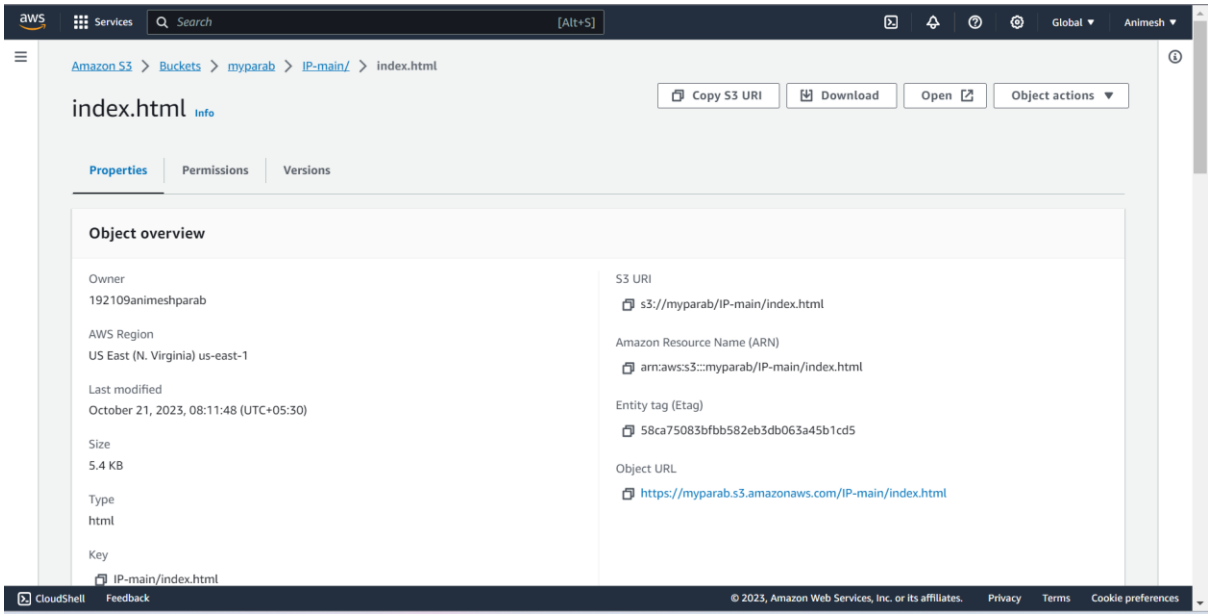




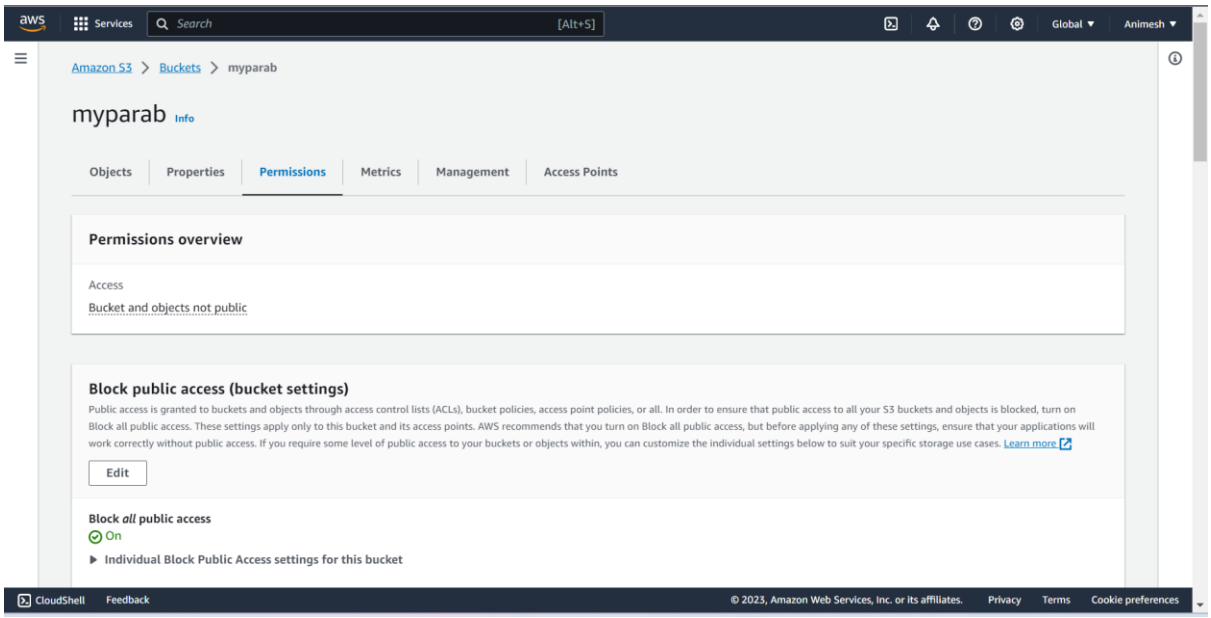
4. Click on Upload button to add files and folders of your projects.



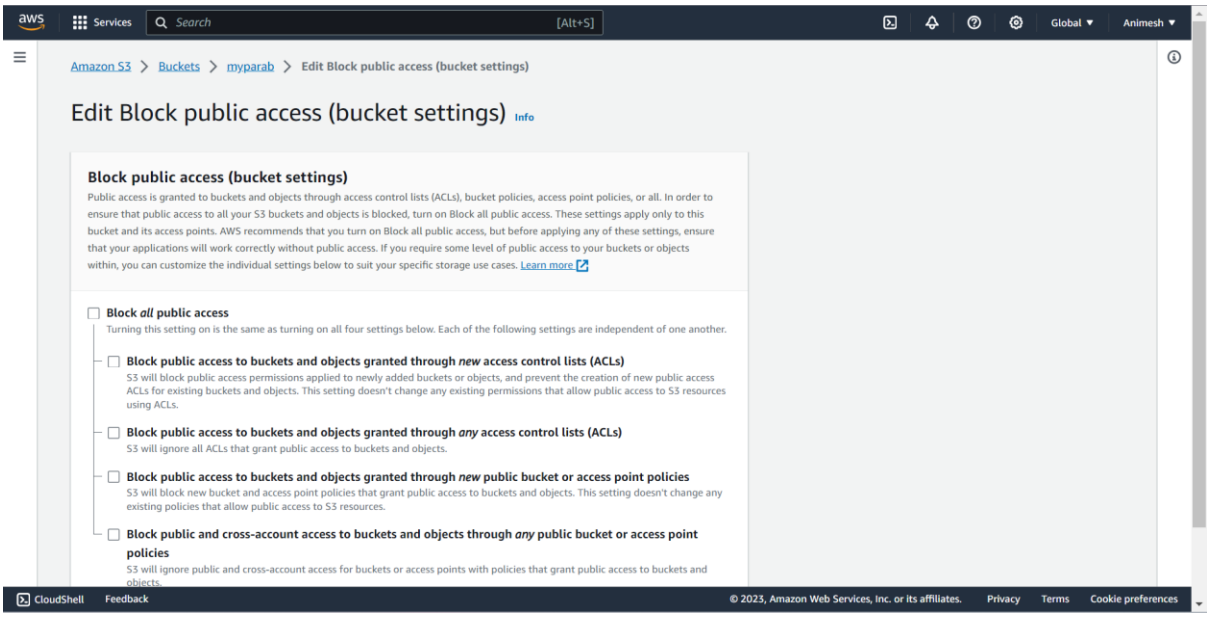
5.Go to the index.html folder and copy the Object URL. Copy this URL to another web page, it will show error.

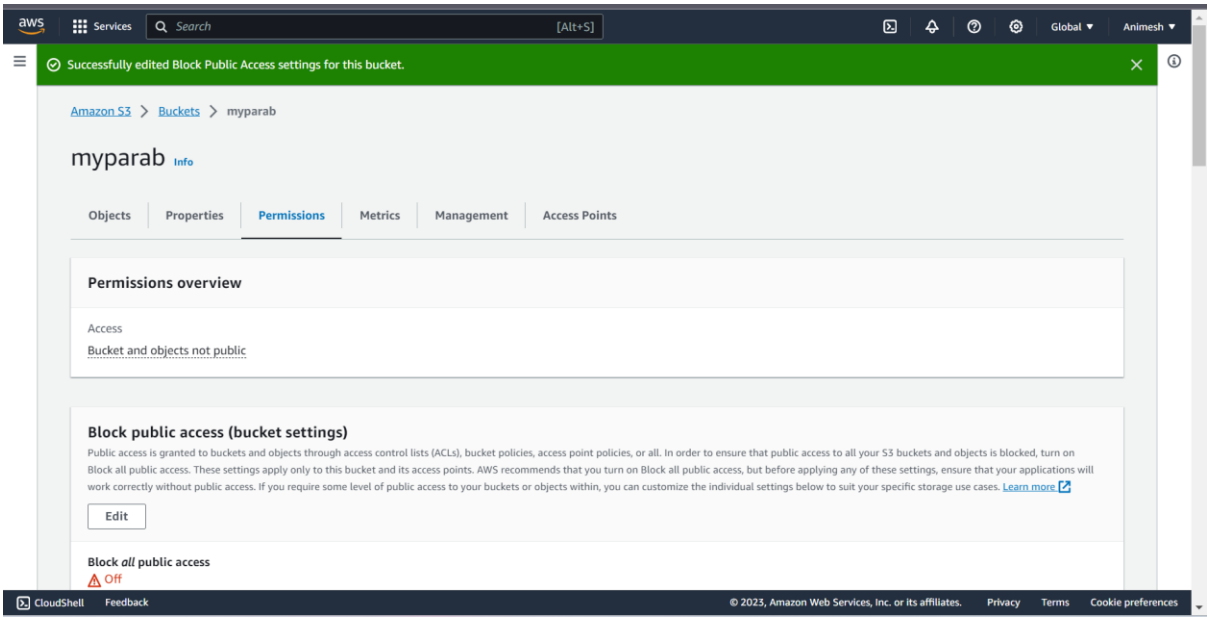


5. Now we need to enable permissions. Click on Premission tab.

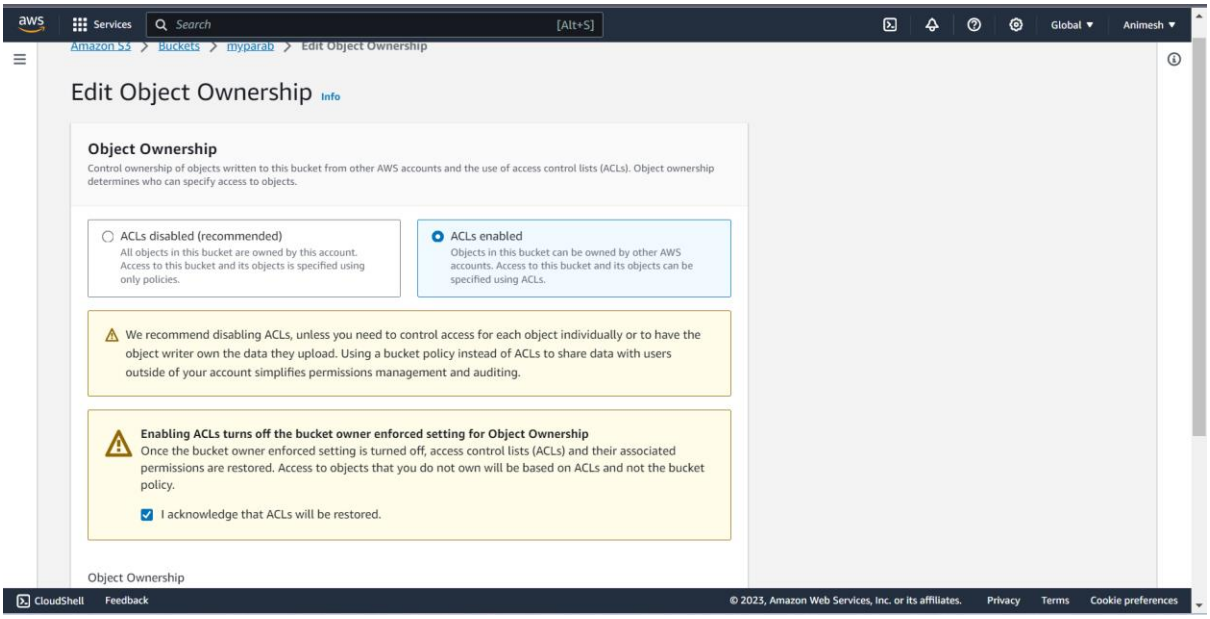


6. Change the public access permission.

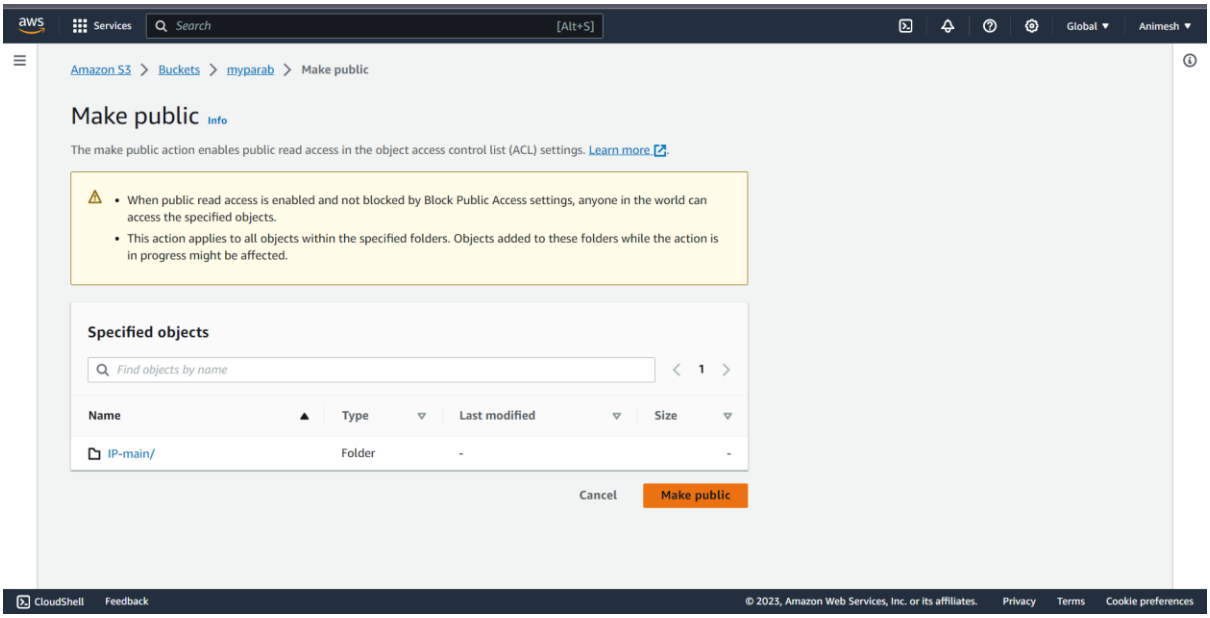


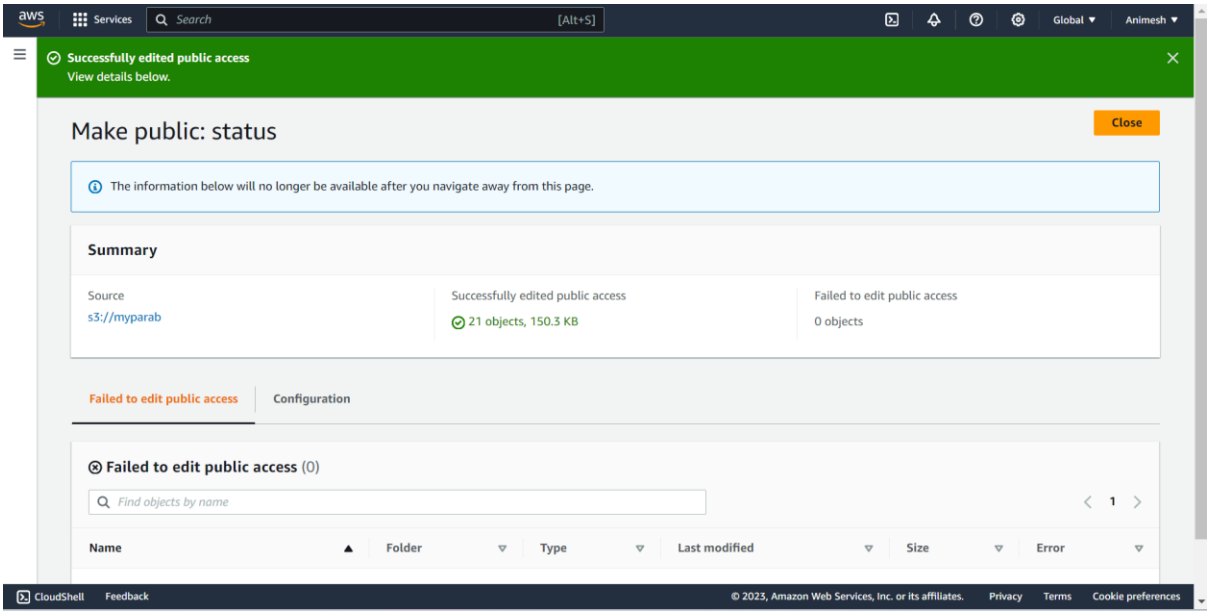


7. . Go to the Object Ownership and Enable the ACLs

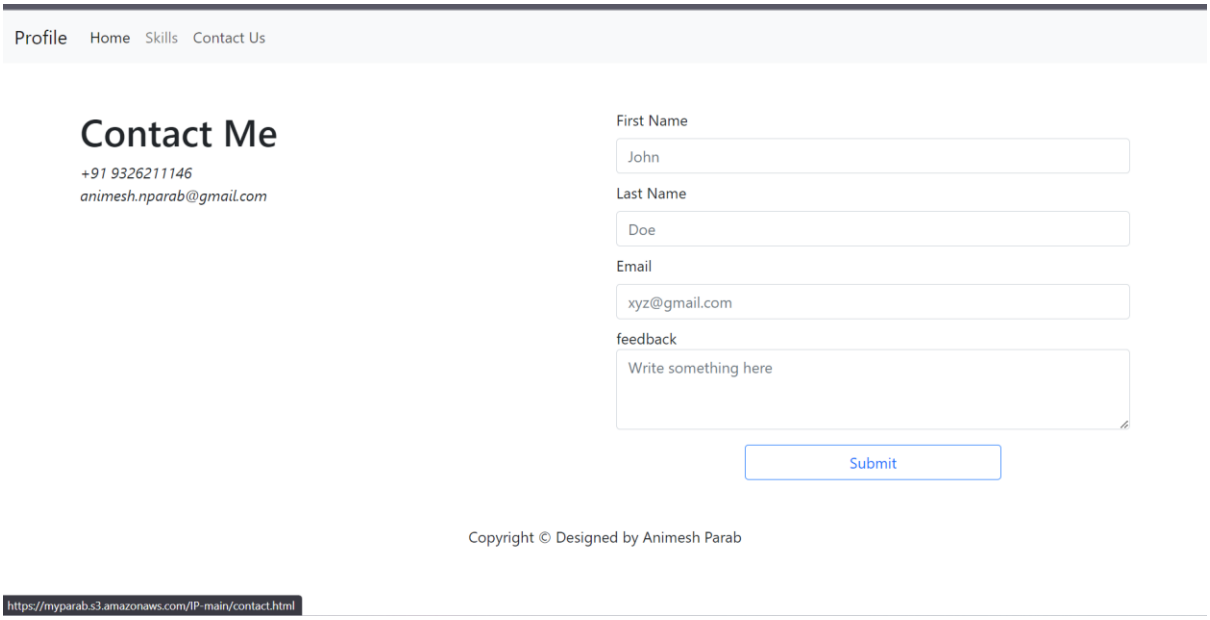


8. **Finally select all the files and folders and click on Actions and click on 'Make public using ACL'.**

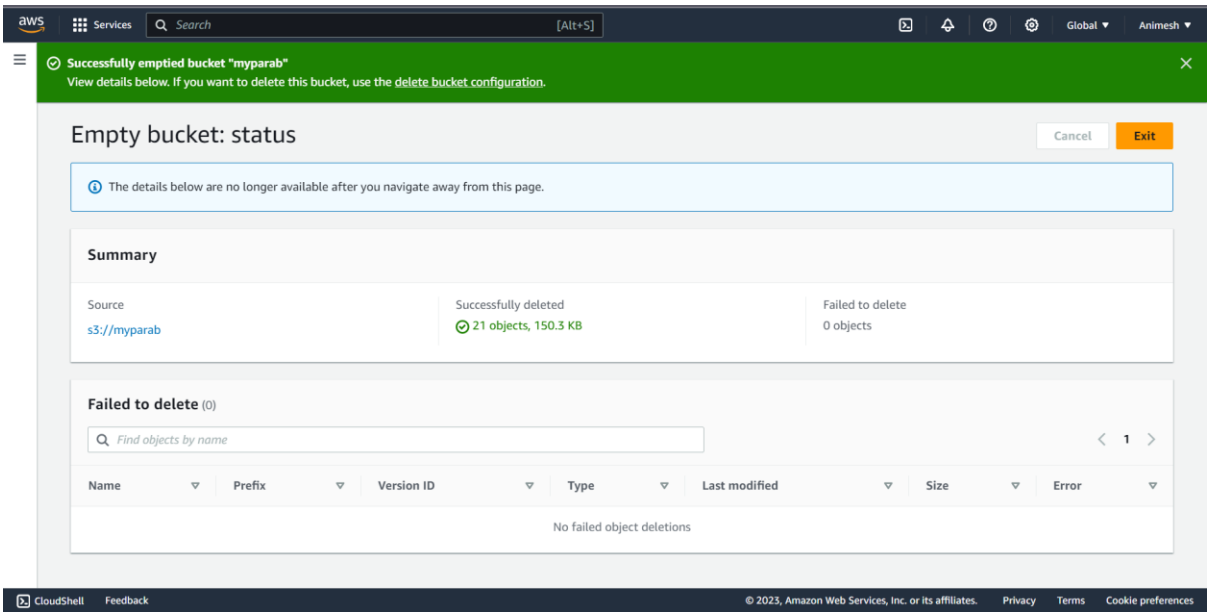


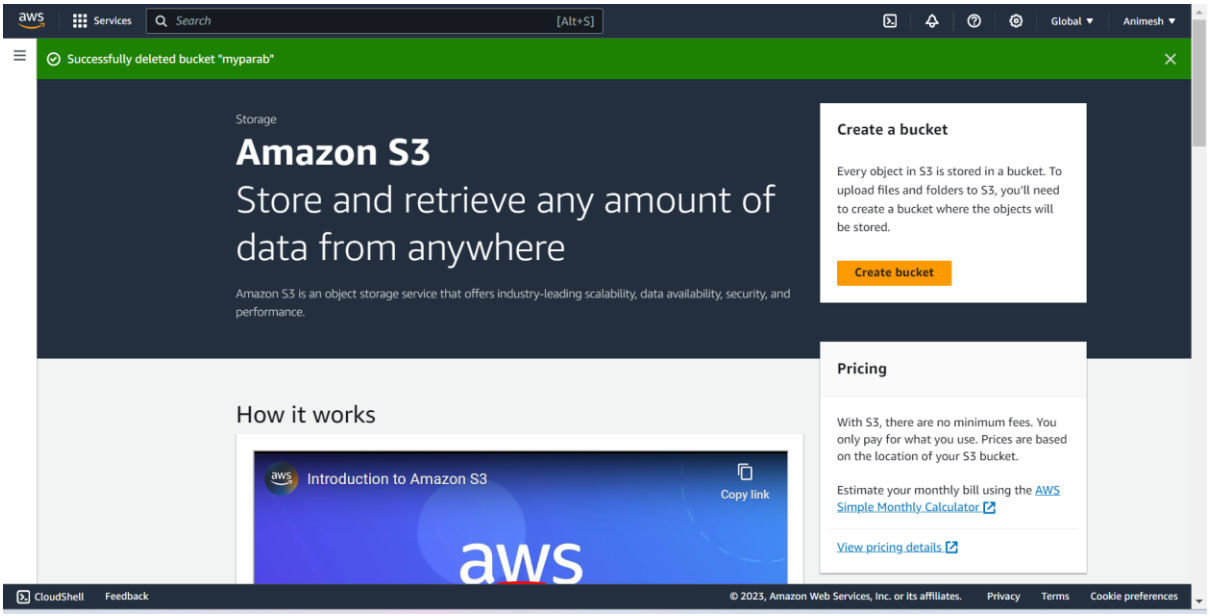

Services
Search
[Alt+S]
Global
Animesh
Successfully edited public access
View details below.
Make public: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Source
s3://myparab
Successfully edited public access
21 objects, 150.3 KB
Failed to edit public access
0 objects
Failed to edit public access
Configuration
Failed to edit public access (0)
Find objects by name
Name
Folder
Type
Last modified
Size
Error
CloudShell
Feedback
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Porfile
Home
Skills
Contact Us
Hello,
Hii Animesh Parab
Software Developer
Profile-image
About me
Animesh is a pre final year student at Thadomal shahani engineering college. He has worked with HSBC as an analyst, where he played the role ofa data engineer. He worked on a software like data visualization. He hold a degree in Information Technology.
Skills



https://myparab.s3.amazonaws.com/IP-main/contact.html





CONCLUSION:-

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

Animesh Parab T2-T21 88

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 toexpose 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: web-app 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 web-app 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.

Animesh Parab T2-T21 88

ASSIGNMENT - 06

AIM- To understand terraform lifecycle and to Build, change, and destroy AWS infrastructure Using Terraform.

LO MAPPED – LO1,LO3

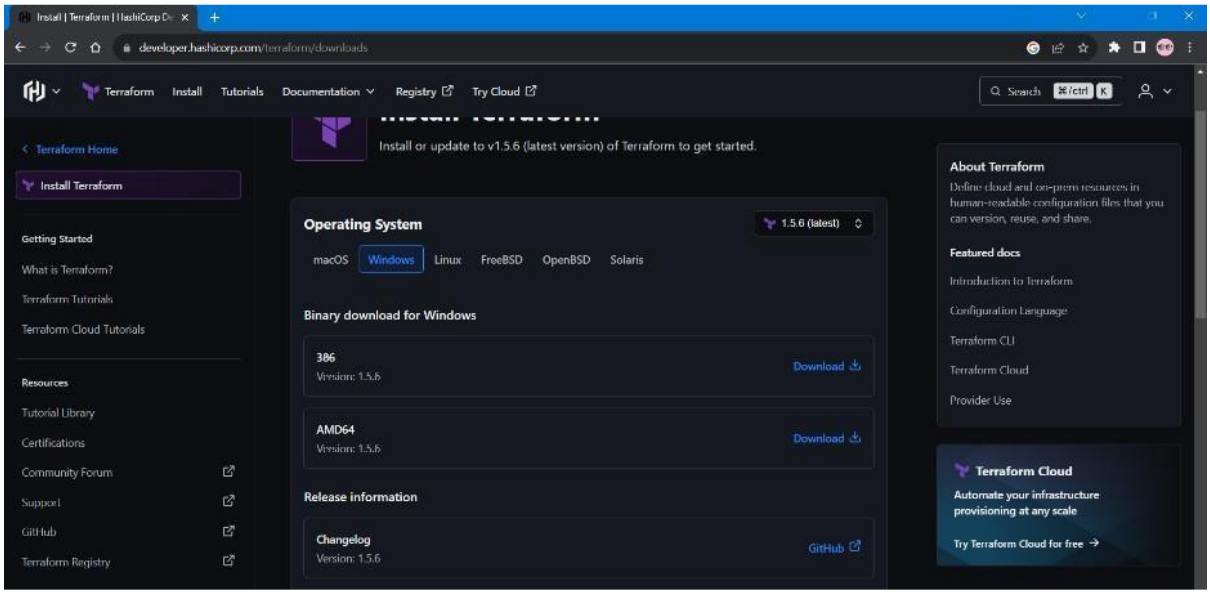
THEORY-
Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in-house solutions.
The key features of Terraform are:
• **Infrastructure as Code**: Infrastructure is described using a high-level configuration syntax. This allows a blueprint of your datacenter to be versioned and treated as you would any other code. Additionally, infrastructure can be shared and re-used.

• **Execution Plans**: Terraform has a "planning" step where it generates an execution plan. The execution plan shows what Terraform will do when you call apply. This lets you avoid any surprises when Terraform manipulates infrastructure.

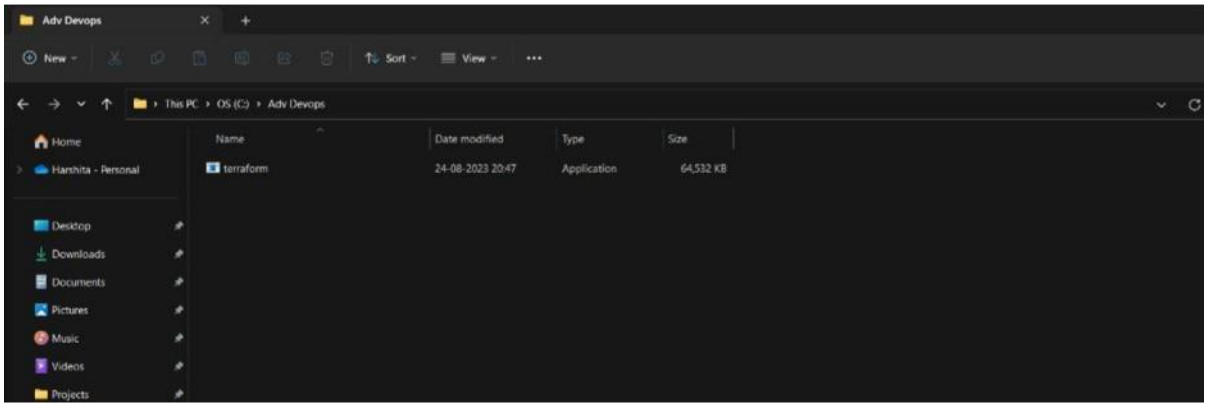
• **Resource Graph**: Terraform builds a graph of all your resources, and parallelizes the creation and modification of any non-dependent resources. Because of this, Terraform

builds infrastructure as efficiently as possible, and operators get insight into dependencies in their infrastructure.

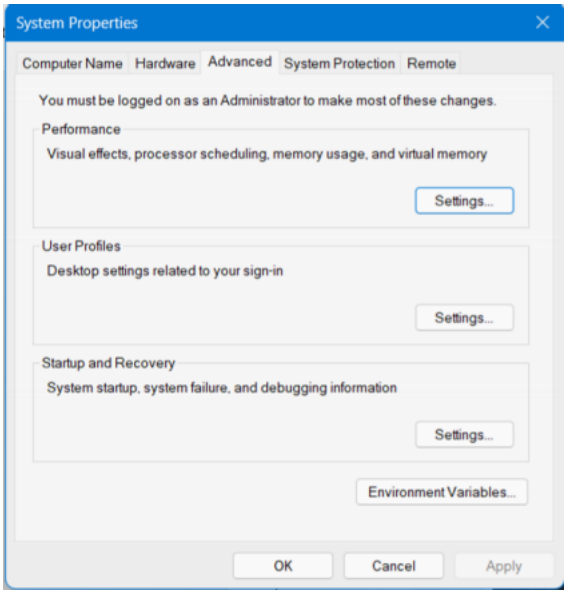
• **Change Automation**: Complex changesets can be applied to your infrastructure with minimal human interaction. With the previously mentioned execution plan and resource graph, you know exactly what Terraform will change and in what order, avoiding many possible human errors.
STEPS-
Google Search – Terraform DOWNLOAD



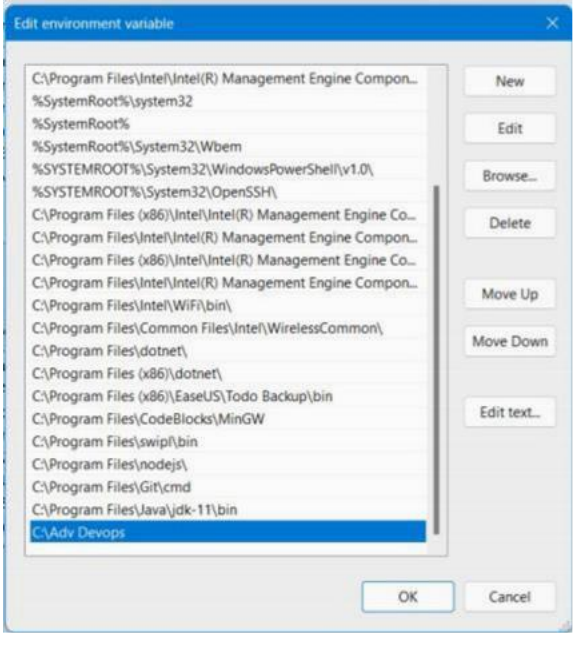
Create a Folder in Your C drive Named AdvDevops
Then Extract The downloaded file in The C drive Folder Named AdvDevops Shown Below



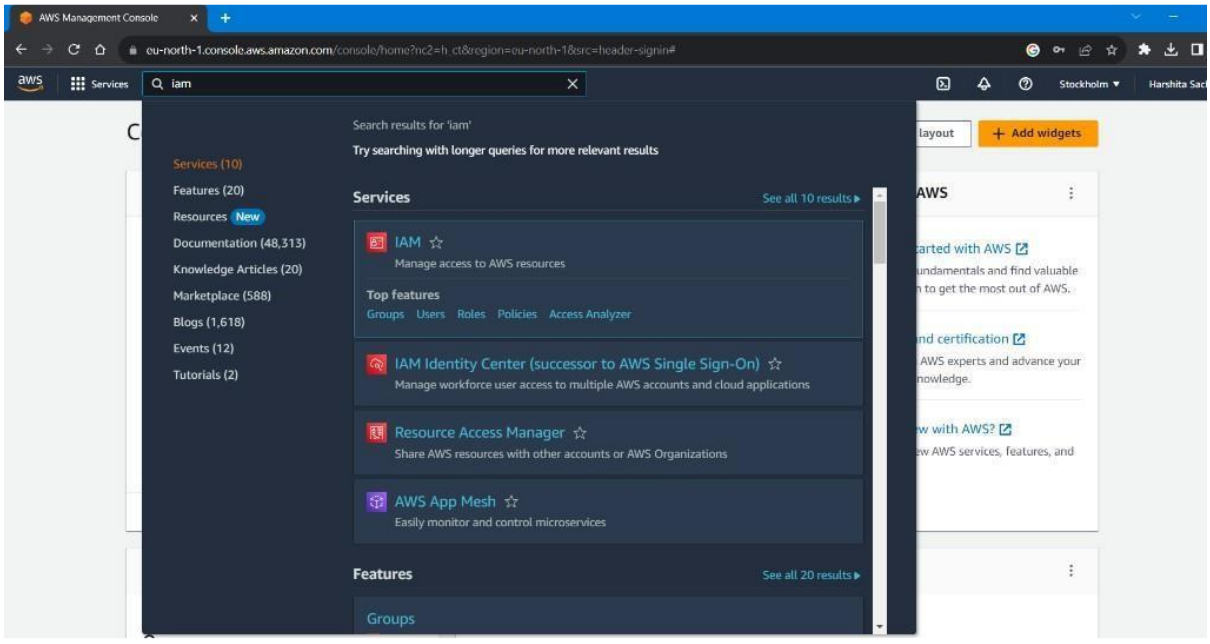
Now search EDIT THE SYSTEM ENVIRONMENT VARIABLES in your windows se



Now click on PATH OF USER VARIBLES, then click on Edit option
Now go to edit and then add new path C:\AdvDevOps



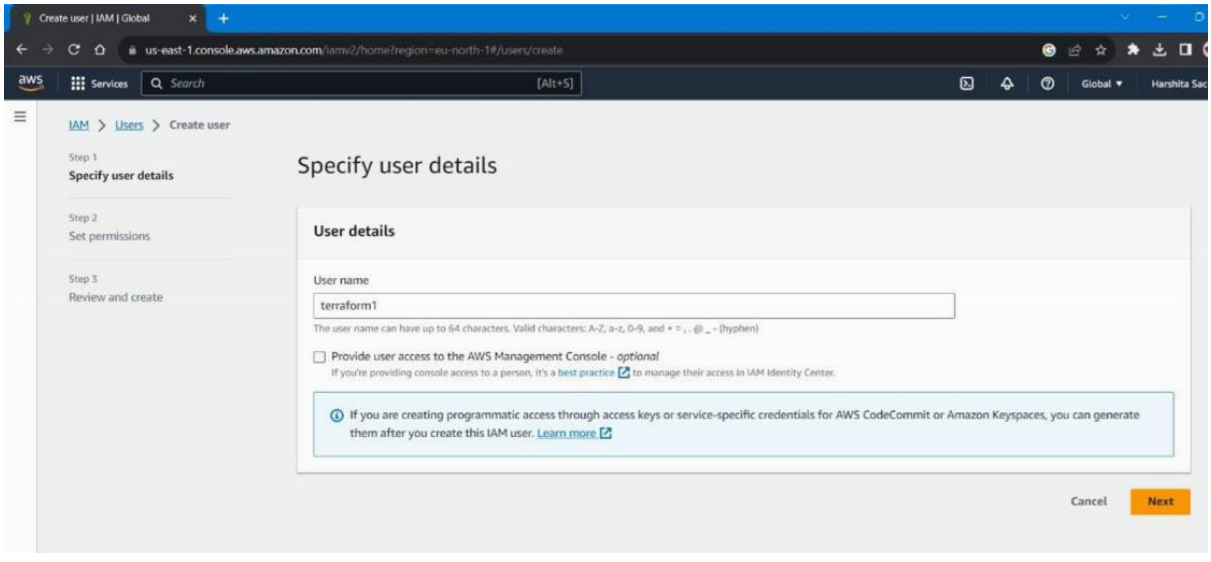
Repeat same procedure for system variables.

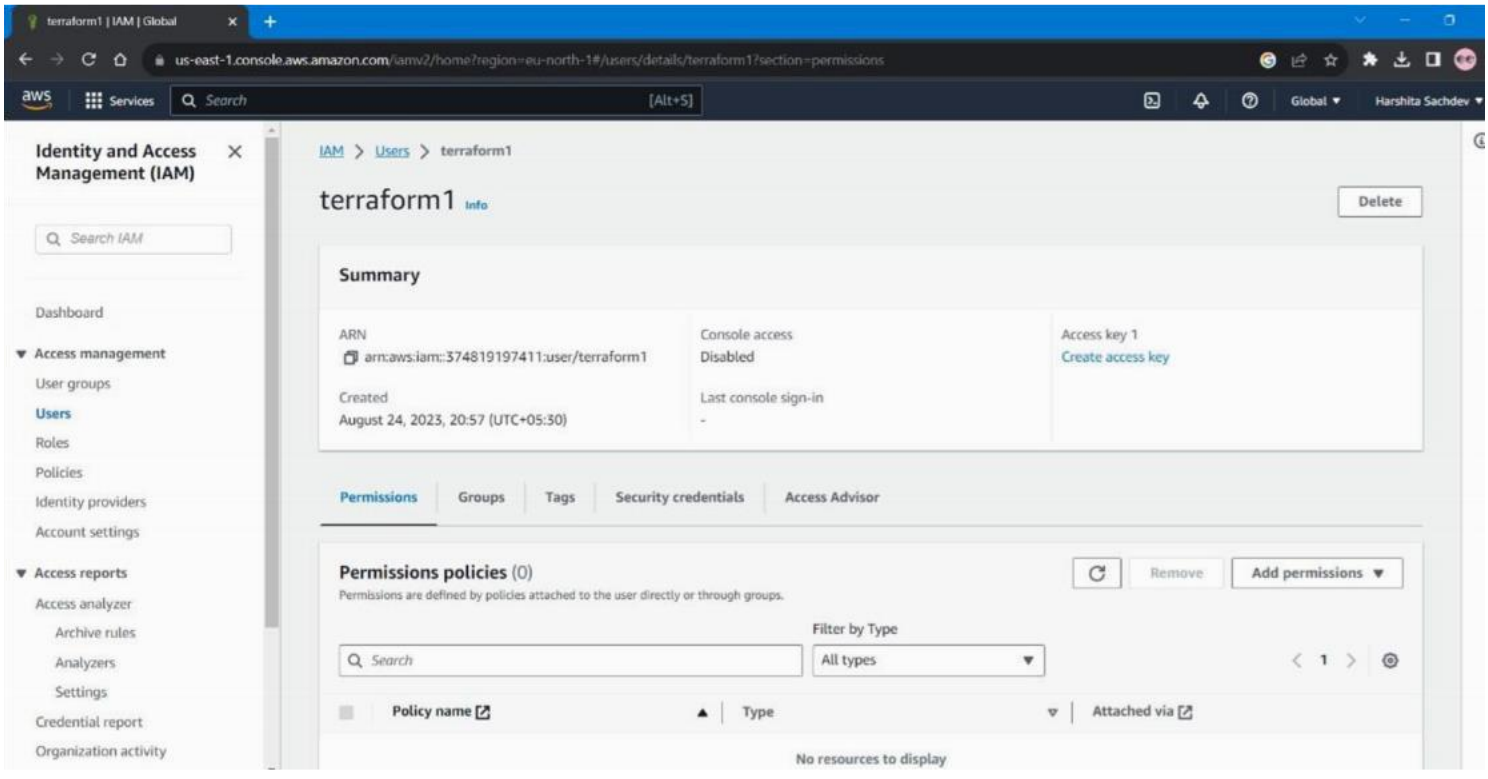


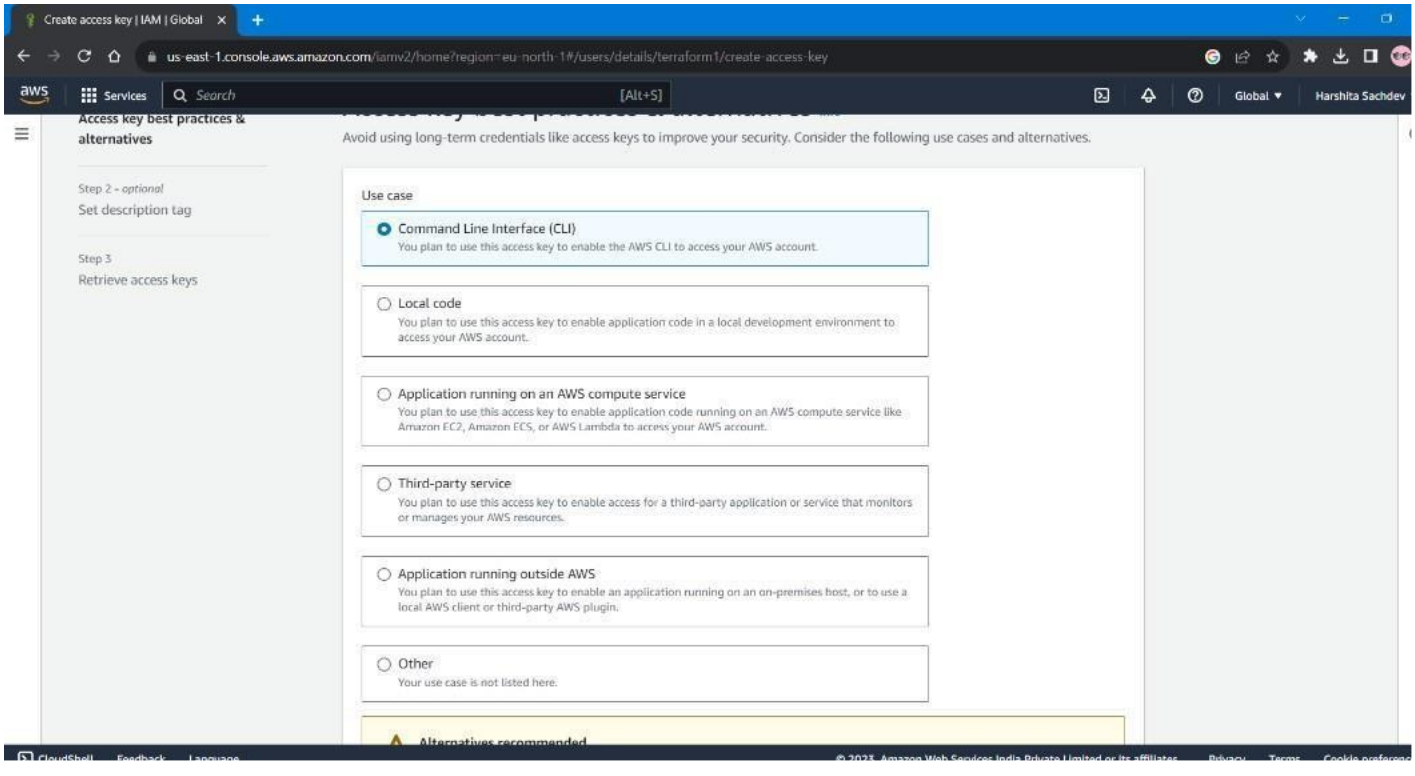
Open And Login to your AWS console- And search IAM and click on it

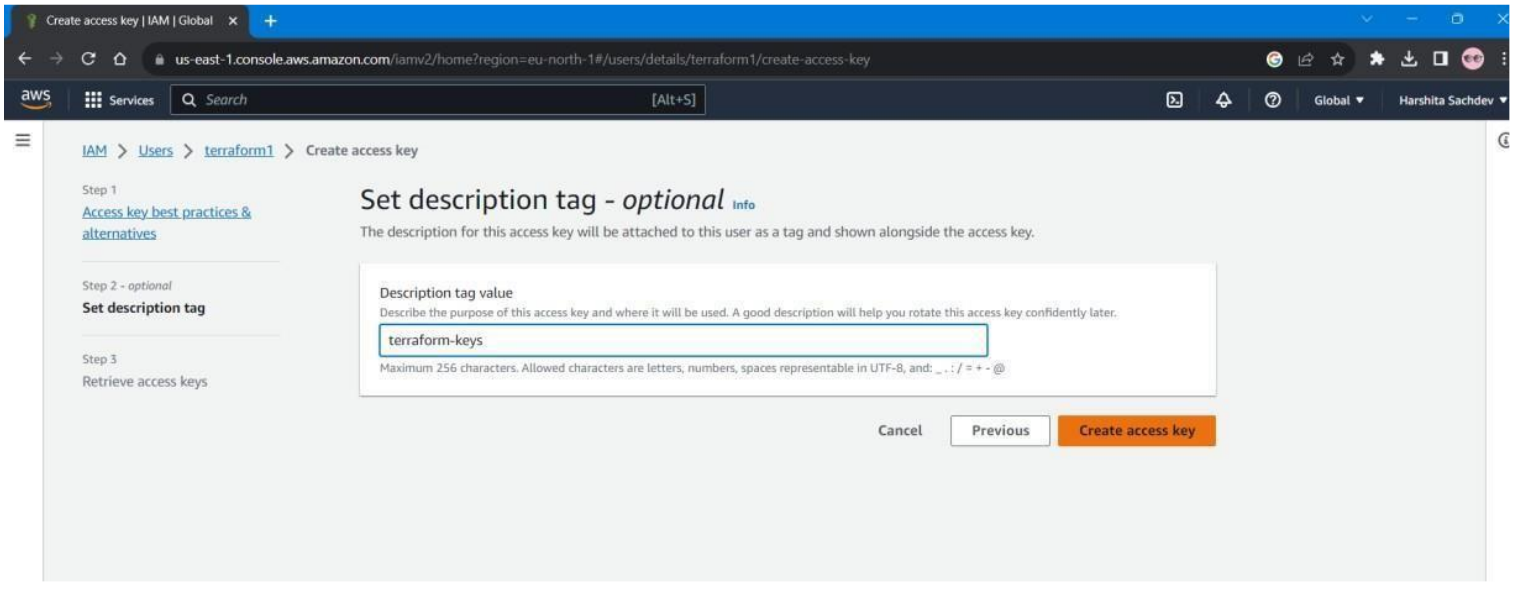
Now click on Add Users in The User Section as shown in the image and add the user

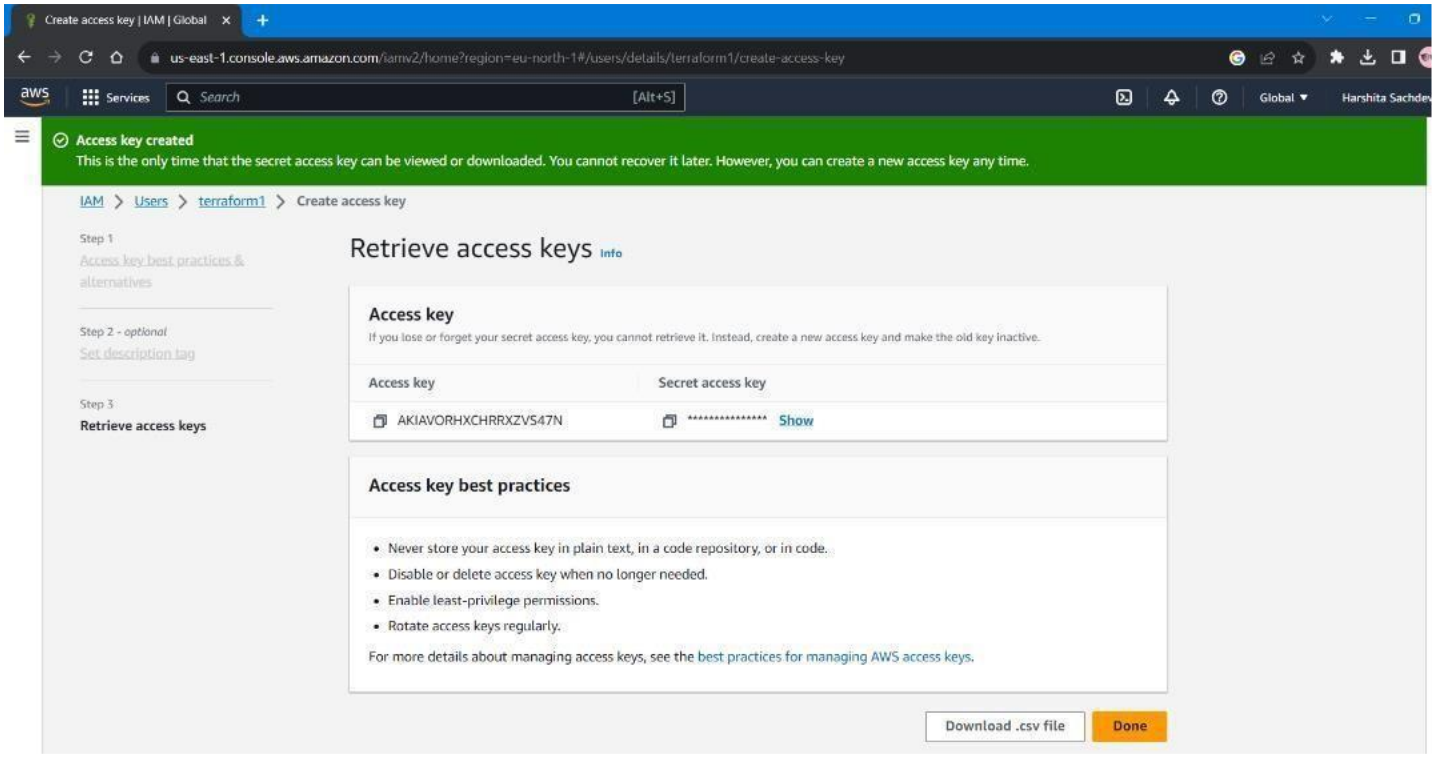
name

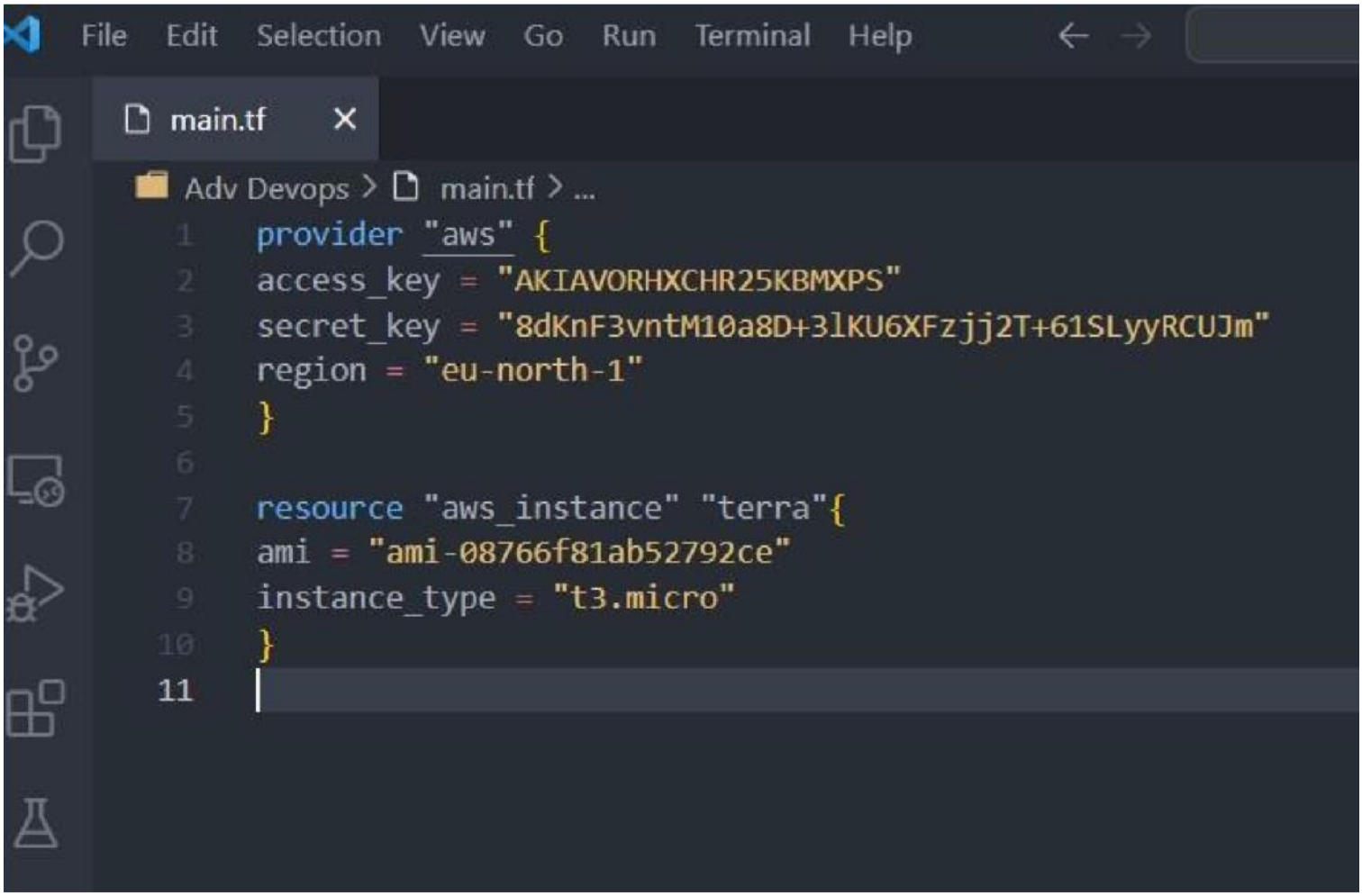


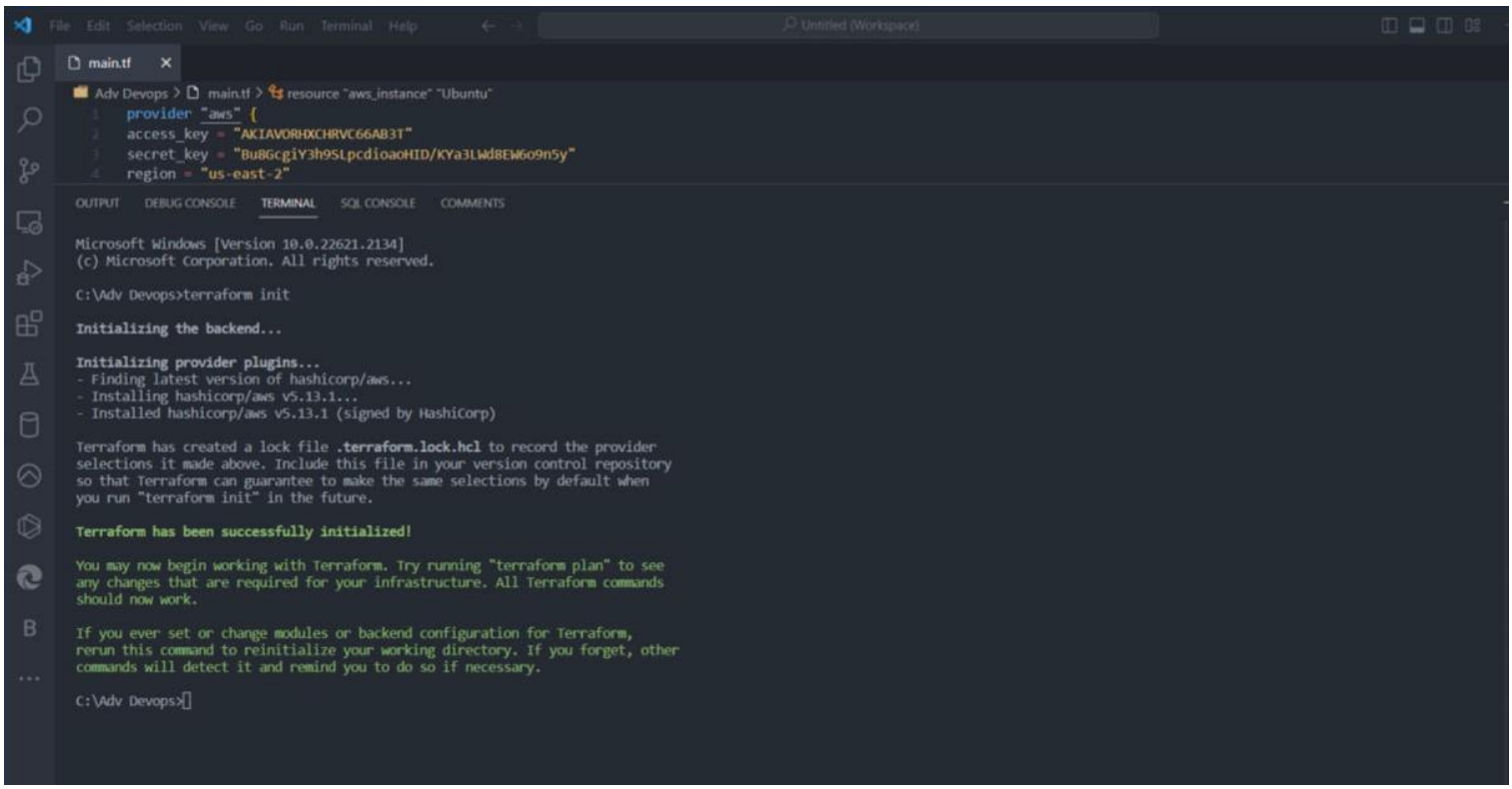

terraform1 | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1?section=permissions
aws
Services
Search
[Alt+S]
Global
Harshita Sachdev
Identity and Access Management (IAM)
Search IAM
Dashboard
Access management
User groups
Users
Roles
Policies
Identity providers
Account settings
Access reports
Access analyzer
Archive rules
Analyzers
Settings
Credential report
Organization activity
IAM > Users > terraform1
terraform1 Info
Delete
Summary
ARN
arn:aws:iam::374819197411:user/terraform1
Console access
Disabled
Access key 1
Create access key
Created
August 24, 2023, 20:57 (UTC+05:30)
Last console sign-in
-
Permissions
Groups
Tags
Security credentials
Access Advisor
Permissions policies (0)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Filter by Type
Search
All types
1
Policy name
Type
Attached via
No resources to display

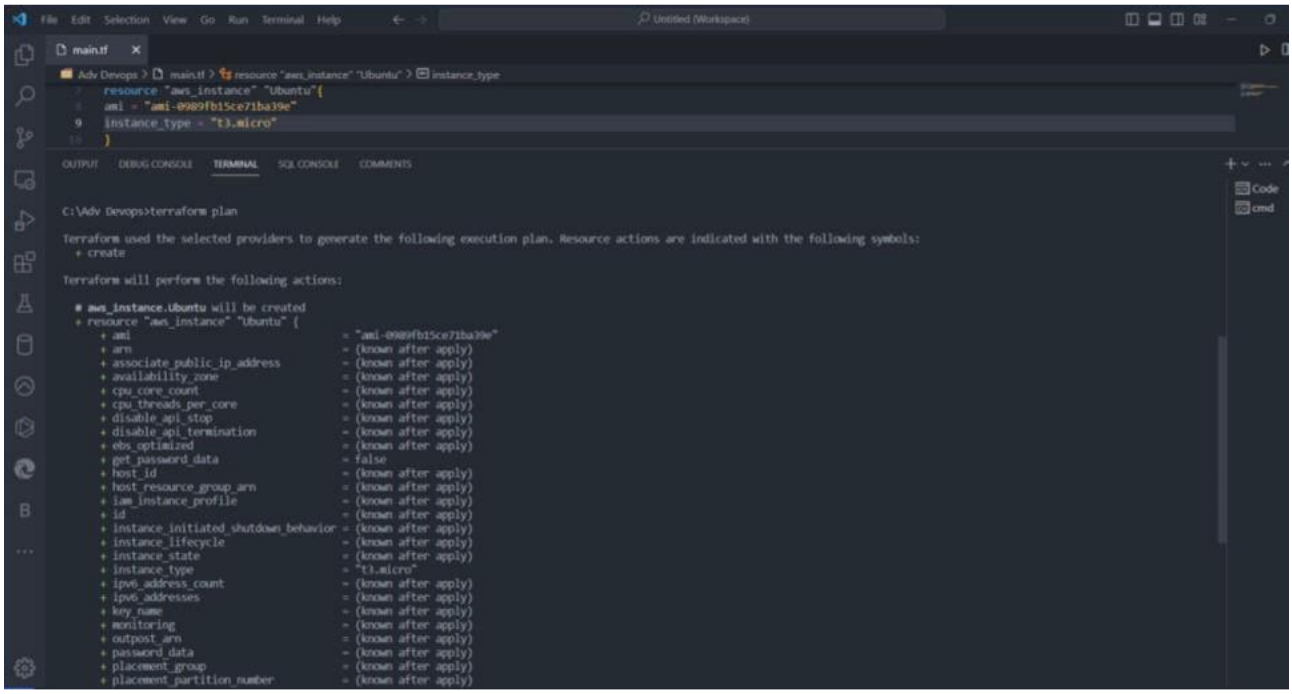

Create access key | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1/create-access-key
aws
Services
Search
[Alt+S]
Global
Harshita Sachdev
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.
Use case
Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.
Application running outside AWS
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.
Other
Your use case is not listed here.
Alternatives recommended
CloudShell
Feedback
Language

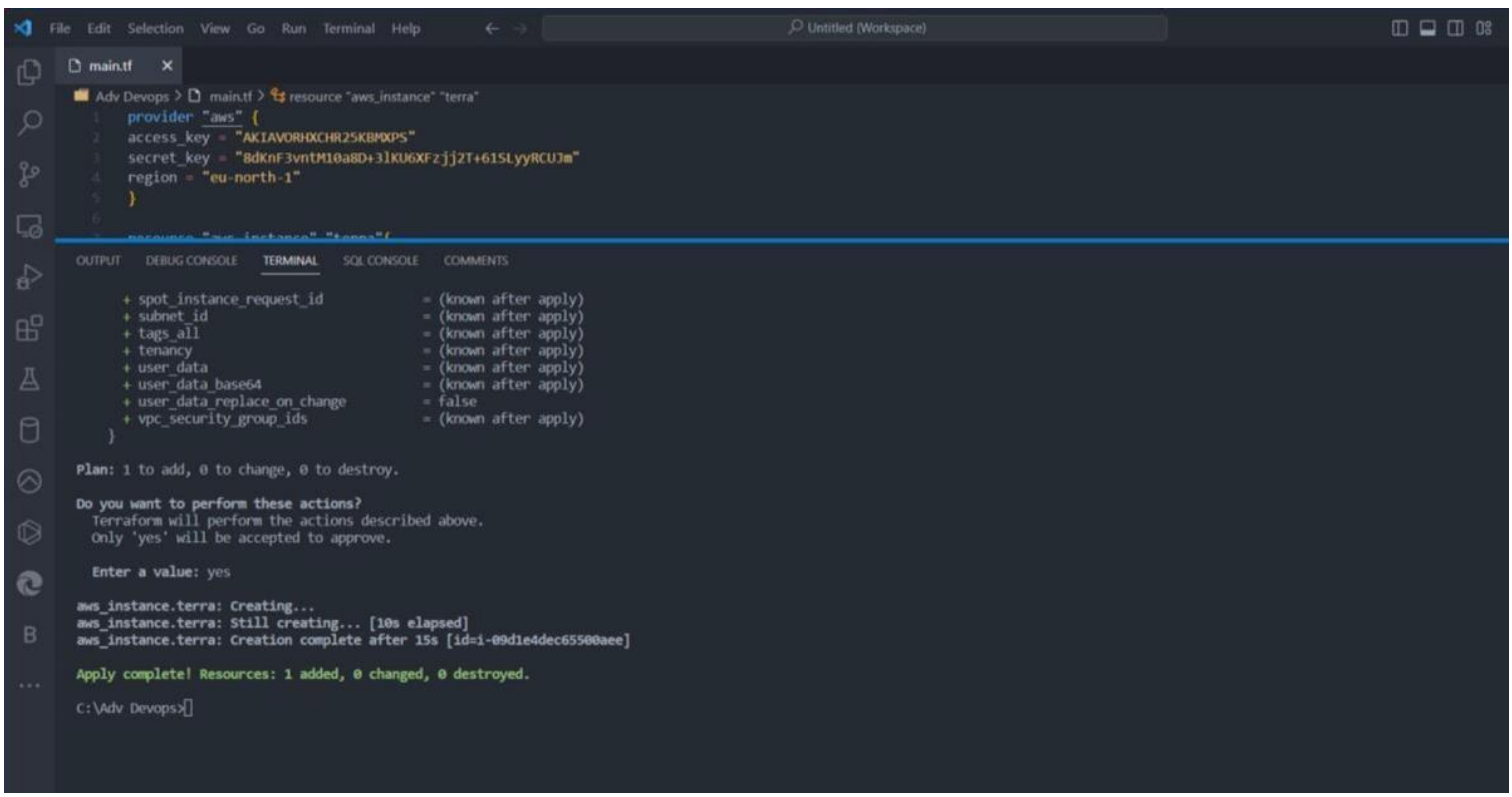

Create access key | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1/create-access-key
Services
Search
[Alt+S]
Global
Harshita Sachdev
IAM > Users > terraform1 > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Set description tag - optional Info
The description for this access key will be attached to this user as a tag and shown alongside the access key.
Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.
terraform-keys
Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @
Cancel
Previous
Create access key

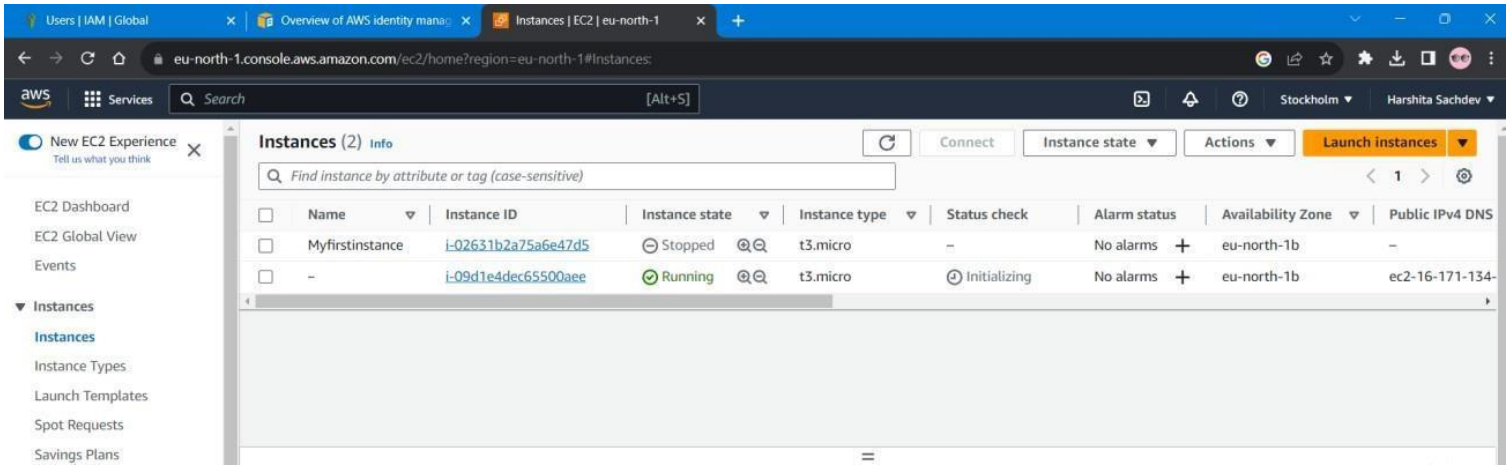

Create access key | IAM | Global
us-east-1.console.aws.amazon.com/iamv2/home?region=eu-north-1#/users/details/terraform1/create-access-key
Services
Search
[Alt+S]
Global
Harshita Sachdev
Access key created
This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.
IAM > Users > terraform1 > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Retrieve access keys Info
Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.
Access key
Secret access key
AKIAVORHXCHRRXZVS47N
********** Show
Access key best practices
Never store your access key in plain text, in a code repository, or in code.
Disable or delete access key when no longer needed.
Enable least-privilege permissions.
Rotate access keys regularly.
For more details about managing access keys, see the best practices for managing AWS access keys.
Download .csv file
Done

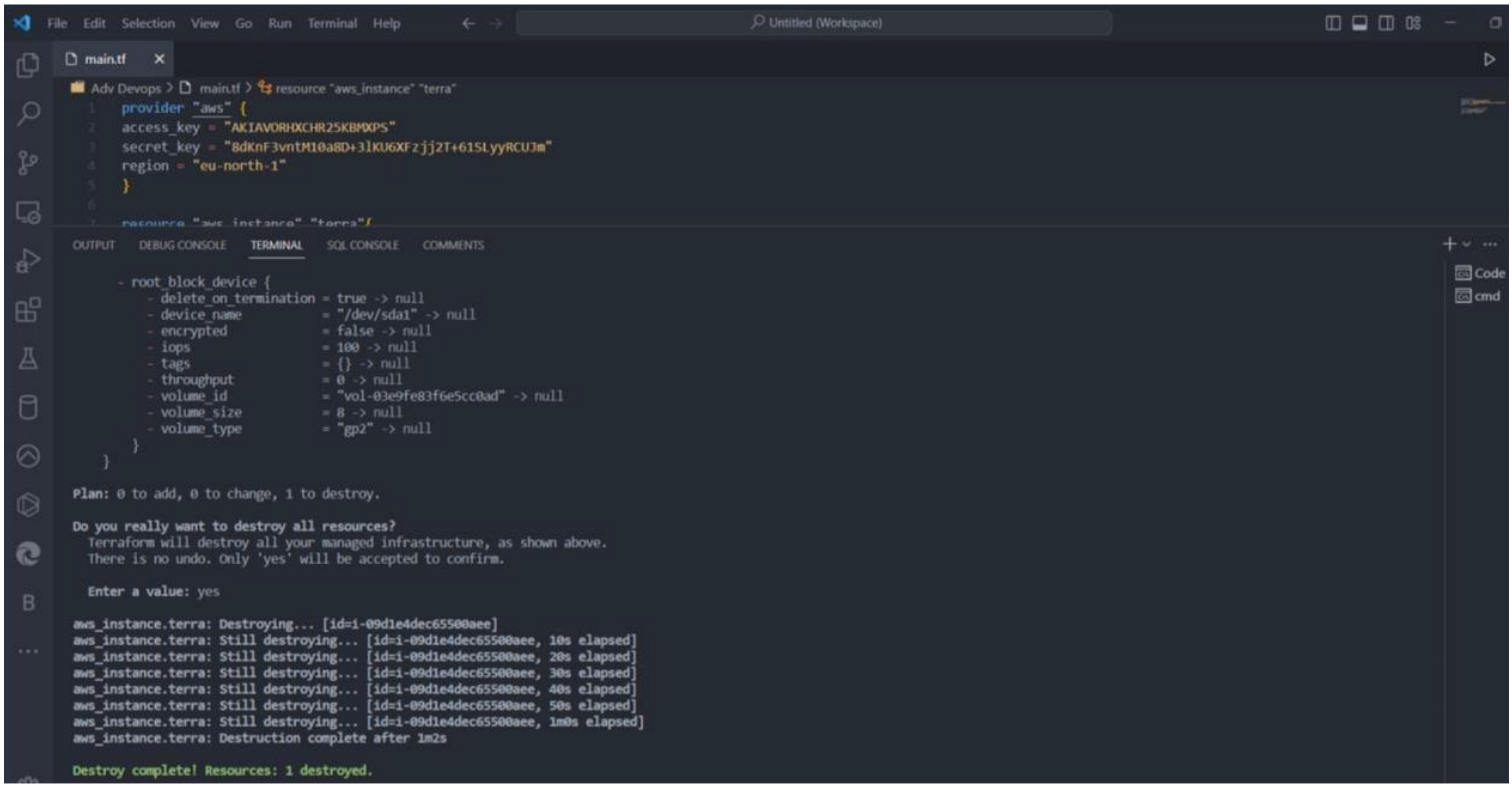

File Edit Selection View Go Run Terminal Help
main.tf
Adv Devops > main.tf > ...
provider "aws" {
access_key = "AKIAVORHXCHR25KBMXPS"
secret_key = "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"
region = "eu-north-1"
}
resource "aws_instance" "terra"{
ami = "ami-08766f81ab52792ce"
instance_type = "t3.micro"
}

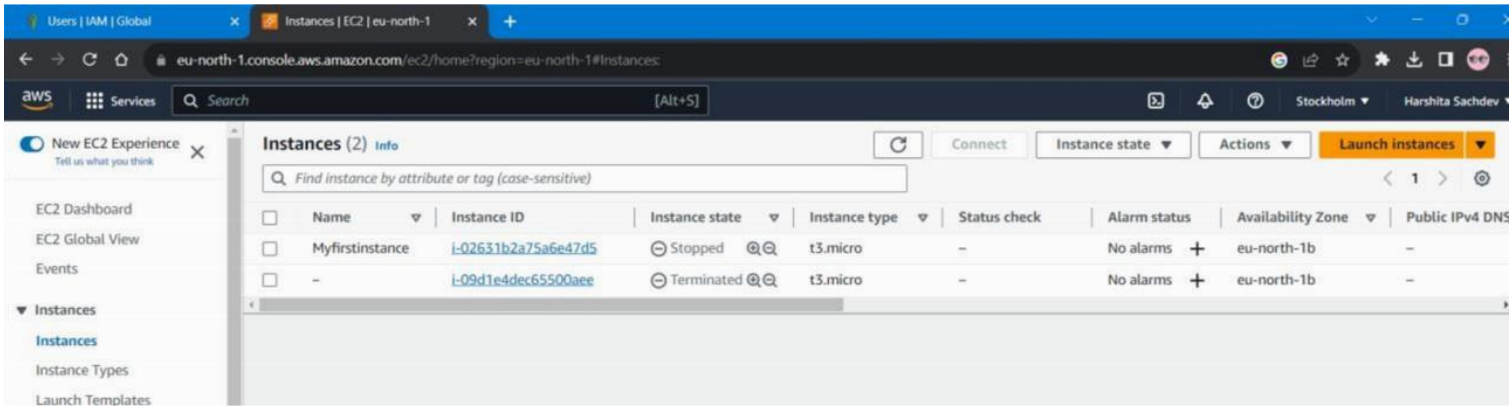

main.tf
Adv Devops > main.tf > resource "aws_instance" "Ubuntu"
provider "aws" {
access_key = "AKIAVORHXCHRVC66AB3T"
secret_key = "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"
region = "us-east-2"
OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE COMMENTS
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.
C:\Adv Devops>terraform init
Initializing the backend...
Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.13.1...
- Installed hashicorp/aws v5.13.1 (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.
C:\Adv Devops>


File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
main.tf
Adv Devops > main.tf > resource "aws_instance" "Ubuntu" > instance_type
resource "aws_instance" "Ubuntu"{
ami = "ami-0989fb15ce71ba39e"
instance_type = "t3.micro"
}
OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE COMMENTS
Code
cmd
C:\Adv Devops>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-0989fb15ce71ba39e"
+ 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_stop = (known after apply)
+ disable_api_termination = (known after apply)
+ ebs_optimized = (known after apply)
+ get_password_data = false
+ host_id = (known after apply)
+ host_resource_group_arn = (known after apply)
+ iam_instance_profile = (known after apply)
+ id = (known after apply)
+ instance_initiated_shutdown_behavior = (known after apply)
+ instance_lifecycle = (known after apply)
+ instance_state = (known after apply)
+ instance_type = "t3.micro"
+ ipv6_address_count = (known after apply)
+ ipv6_addresses = (known after apply)
+ key_name = (known after apply)
+ monitoring = (known after apply)
+ outpost_arn = (known after apply)
+ password_data = (known after apply)
+ placement_group = (known after apply)
+ placement_partition_number = (known after apply)


File Edit Selection View Go Run Terminal Help
Untitled (Workspace)
main.tf
Adv Devops > main.tf > resource "aws_instance" "terra"
provider "aws" {
access_key = "AKIAVORHXCHR25KBMXPS"
secret_key = "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"
region = "eu-north-1"
}
OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE COMMENTS
+ spot_instance_request_id = (known after apply)
+ subnet_id = (known after apply)
+ tags_all = (known after apply)
+ tenancy = (known after apply)
+ user_data = (known after apply)
+ user_data_base64 = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = (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.terra: Creating...
aws_instance.terra: Still creating... [10s elapsed]
aws_instance.terra: Creation complete after 15s [id=i-09d1e4dec65500aee]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
C:\Adv Devops>







Delete all security keys and csv files.
Delete users and user groups

CONCLUSION –

In this assignment we learnt about the terraform lifecycle, core concepts/terminologies and installation of terraform in windows and creating/destroying an EC2 instance.

Animesh Parab T2-T21 88

ASSIGNMENT-7

AIM: To perform static analysis on Python programs using SonarQube SAST process.

LO MAPPED: LO4

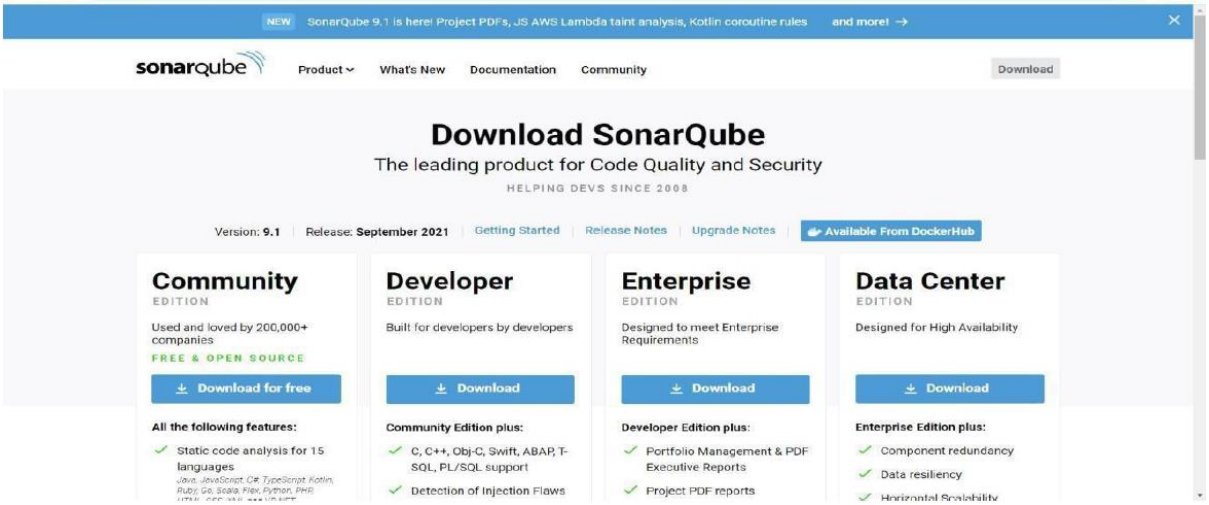
THEORY:
SonarQube is a universal tool for static code analysis that has become more or less the industry standard. Keeping code clean, simple, and easy to read is also a lot easier with SonarQube.
What is SonarQube?
SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.It supports 25+ major programming languages through built-in rulesets and can also be extended with various plugins.
Benefits of SonarQube
Sustainability - Reduces complexity, possible vulnerabilities, and code duplications, optimising the life of applications.Increase productivity - Reduces the scale, cost of maintenance, and risk of the application; as such, it removes the need to spend more time changing the code
☐ Quality code - Code quality control is an inseparable part of the process of software development.
☐ Detect Errors - Detects errors in the code and alerts developers to fix them automatically before submitting them for output.
☐ Increase consistency - Determines where the code criteria are breached and enhances the quality
☐ Business scaling - No restriction on the number of projects to be evaluated
☐ Enhance developer skills - Regular feedback on quality problems helps developers to improve their coding skills

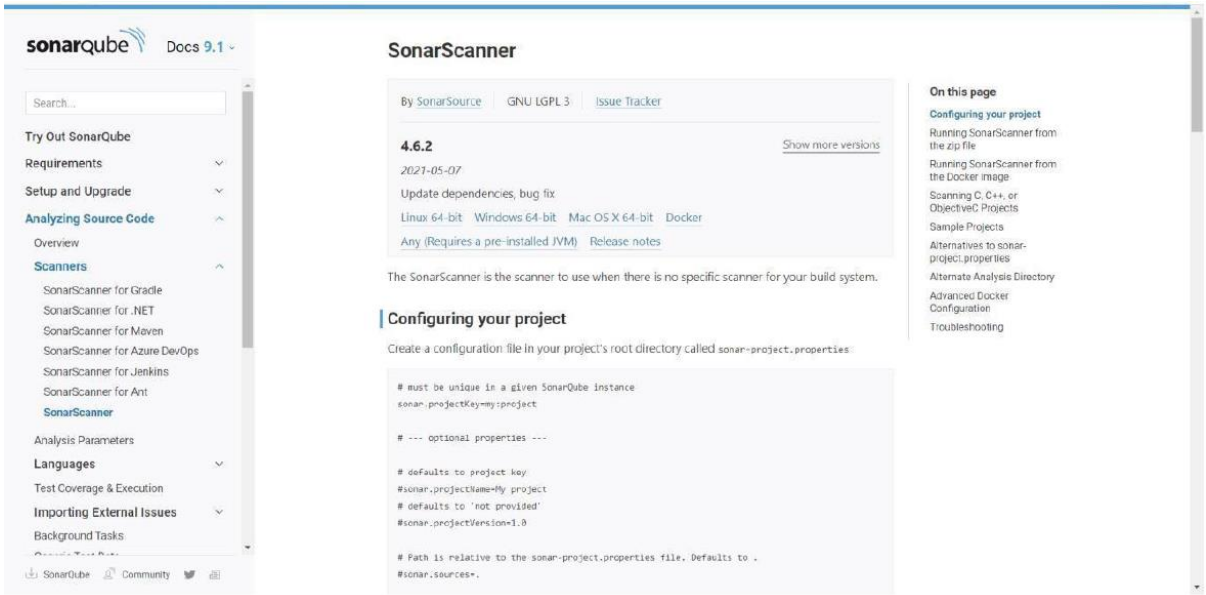
Why SonarQube?
Developers working with hard deadlines to deliver the required functionality to the customer. It is so important for developers that many times they compromise with the code quality, potential bugs, code duplications, and bad distribution of complexity.

Additionally, they tend to leave unused variables, methods, etc. In this scenario, the code would work in the desired way.
To avoid these issues in code, developers should always follow the good coding practice, but sometimes it is not possible to follow the rules and maintain the good quality as there may be many reasons.

In order to achieve continuous code integration and deployment, developers need a tool that not only works once to check and tell them the problems in the code but also to track and control the code to check continuous code quality. To satisfy all these requirements, here comes SonarQube in the picture.

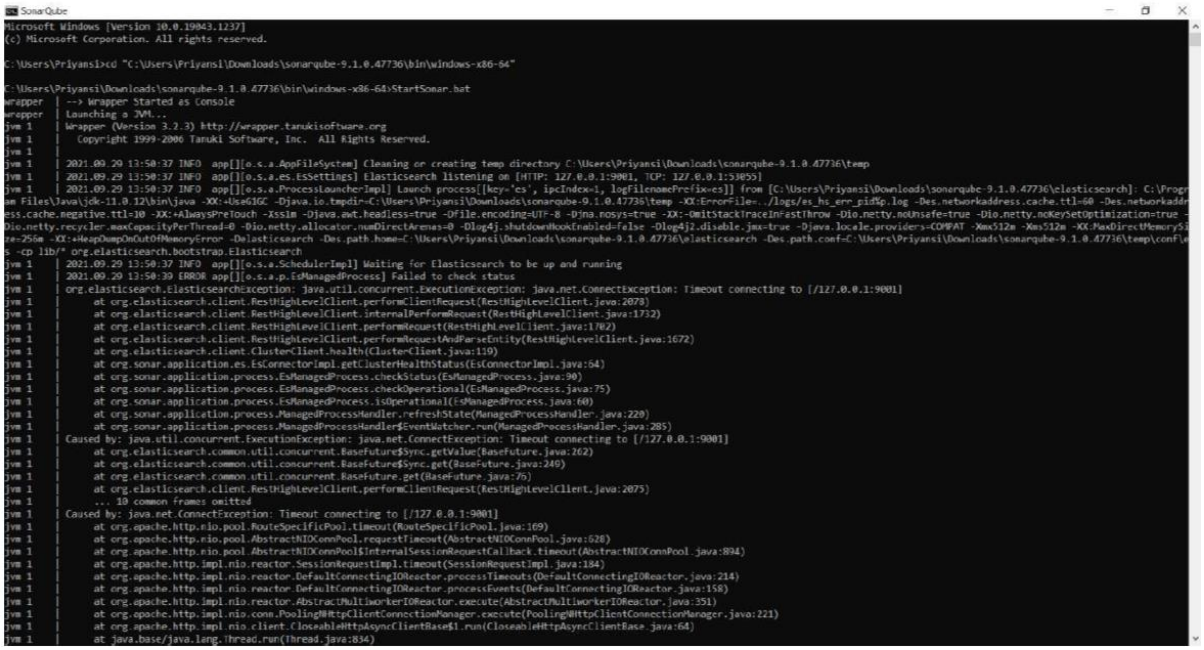
STEPS:
Download SonarQube and Sonar Scanner

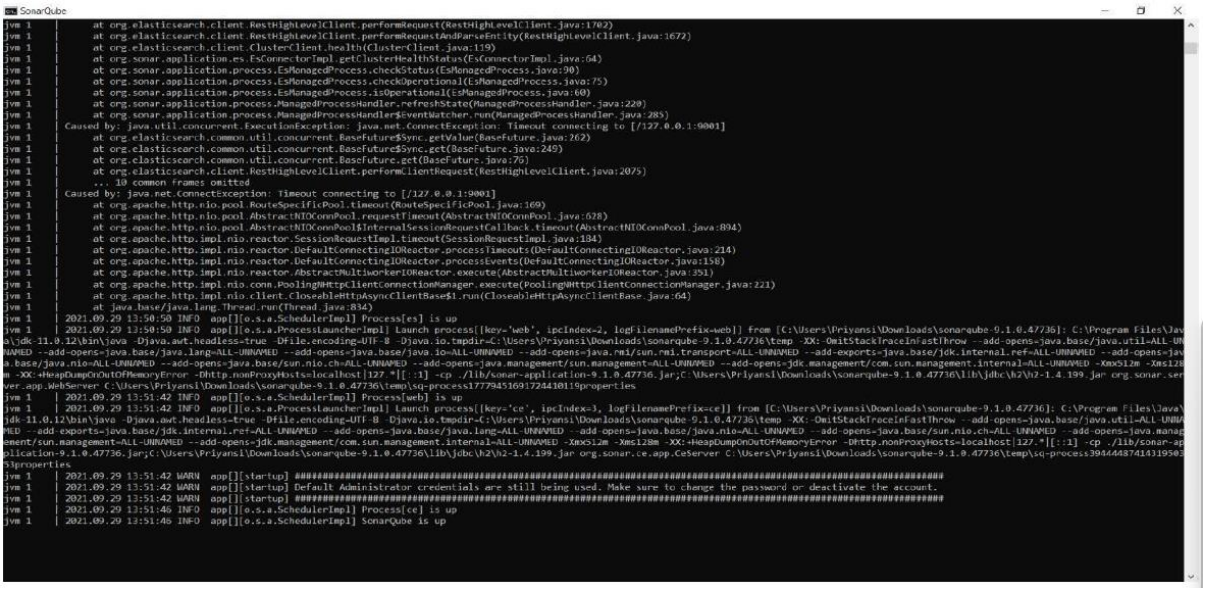




After downloading, set Environment Variables. Add "sonarqube-9.1.0.47736\bin" to Path
Open command prompt. Run commands:

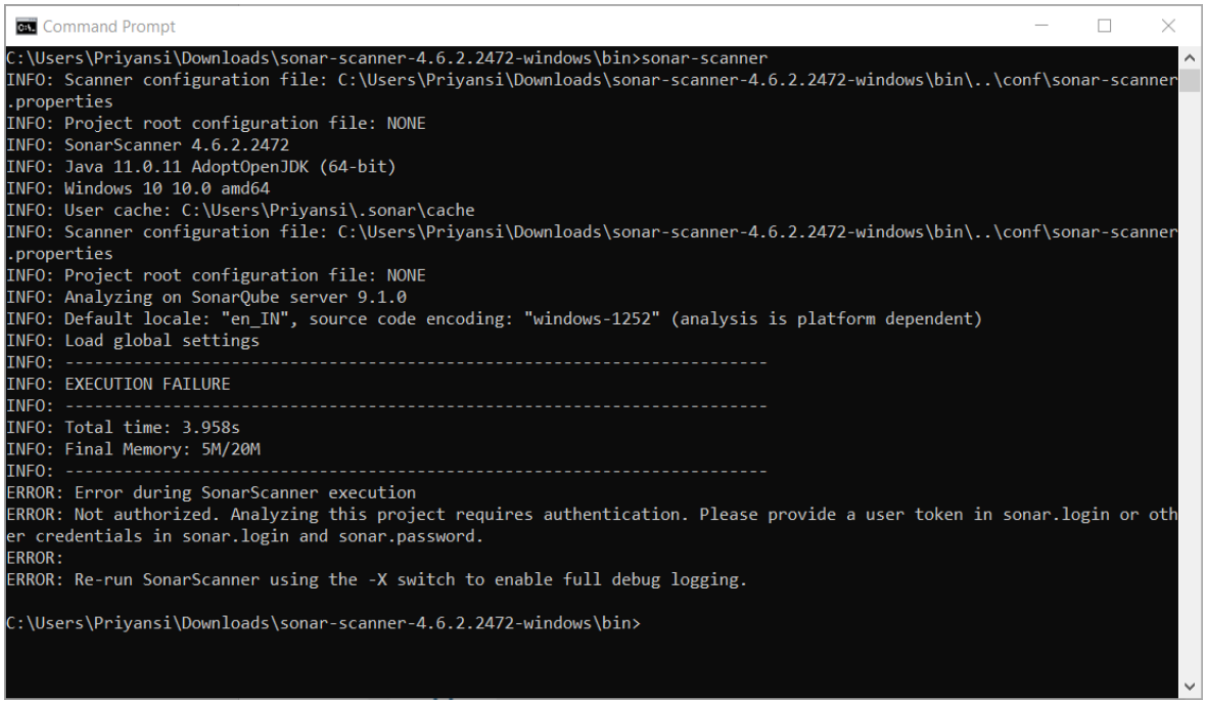
- cd "sonarqube-9.1.0.47736\bin\windows-x86-64"
- StartSonar.bat



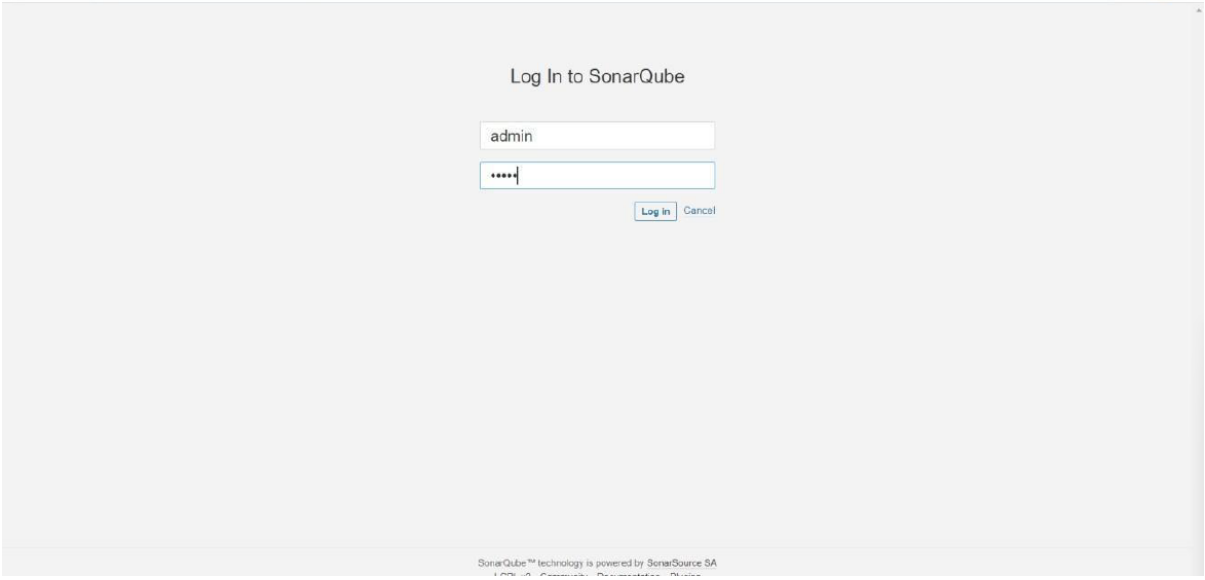


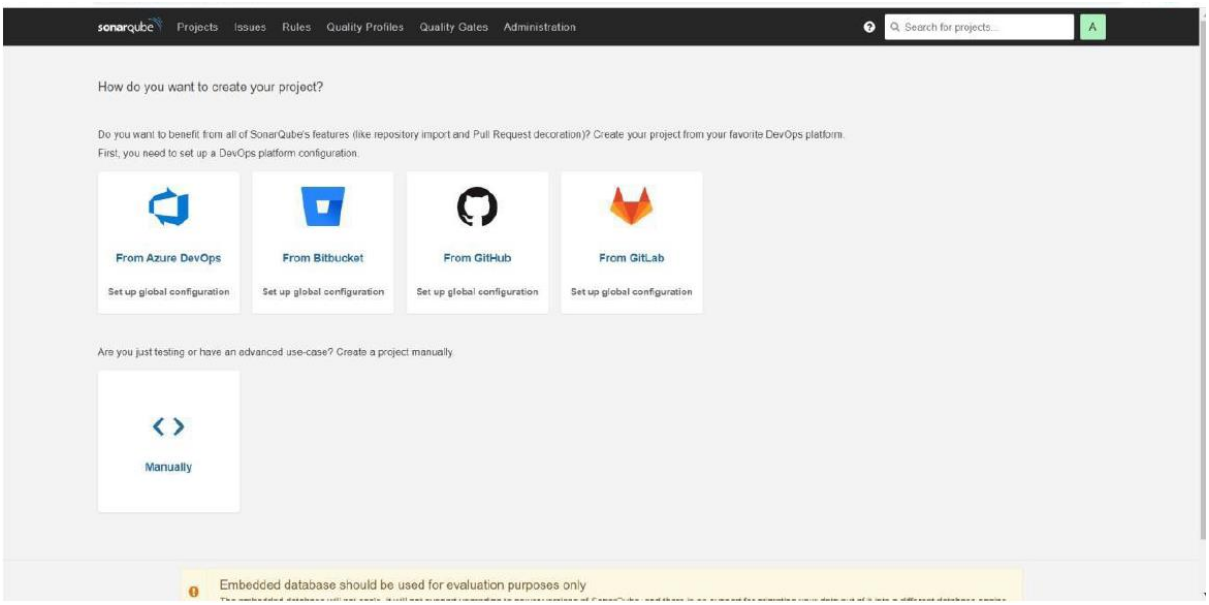
Open another command prompt. Run command:

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

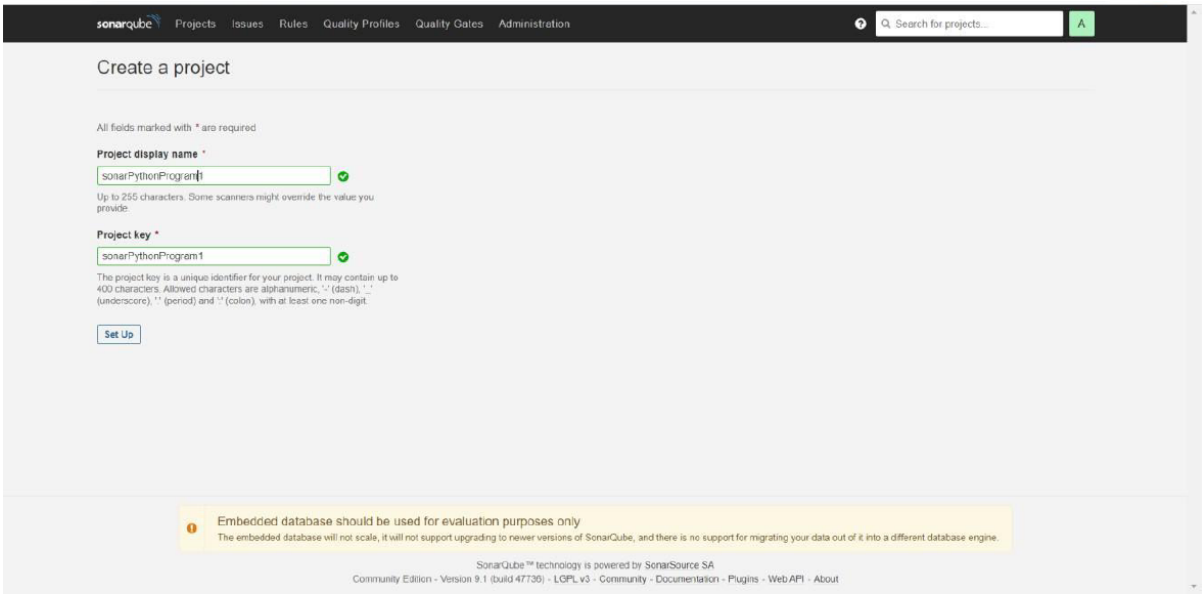


Server up and running on **localhost:9000**
Login using credentials as User: admin and Password: admin and Set a new password

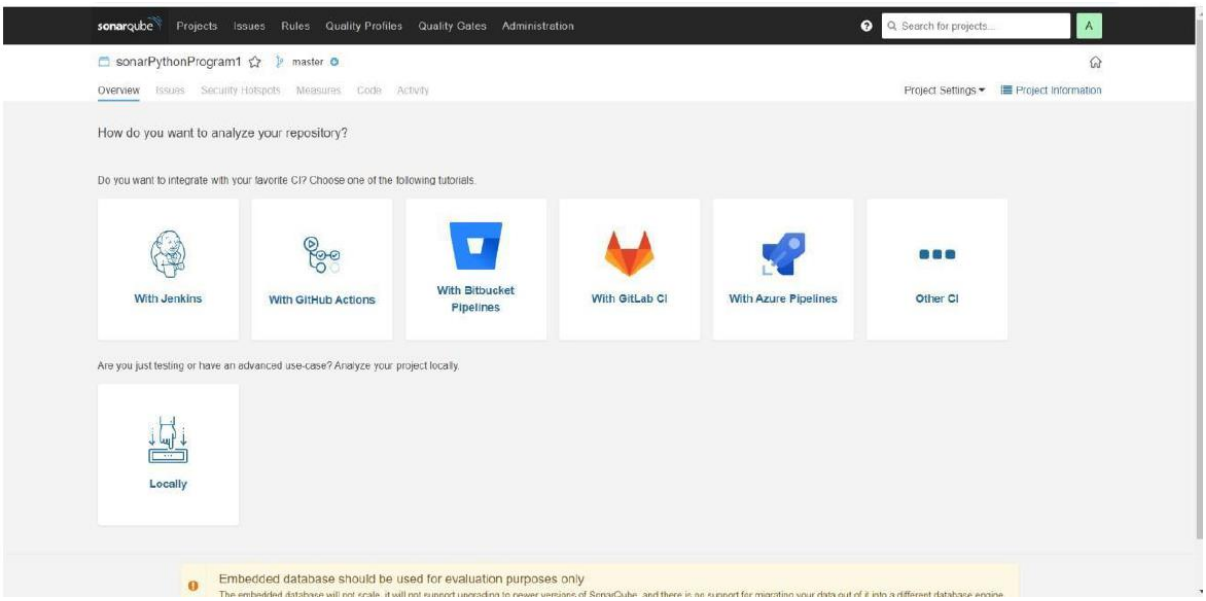




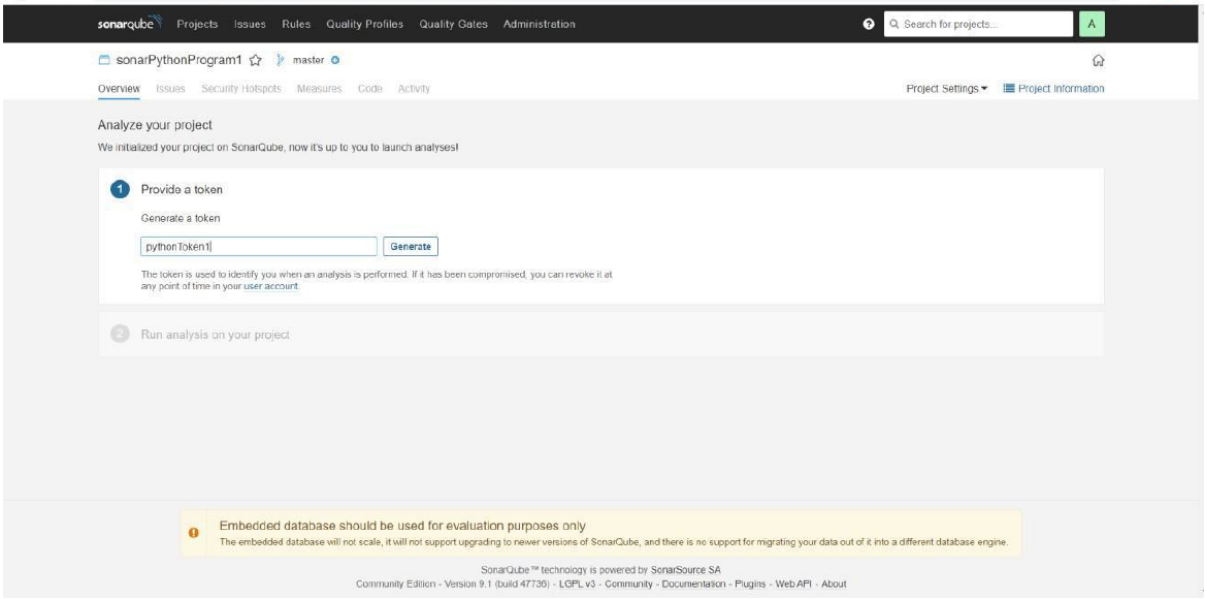
Click on Create a project **Manually.**



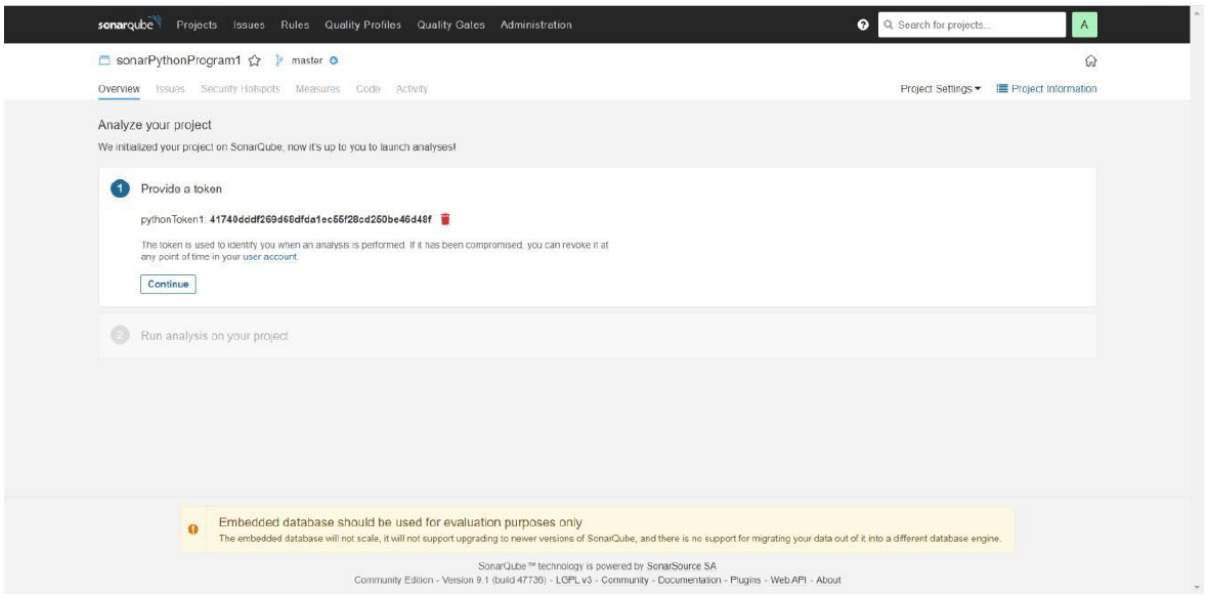
Give any Project display name.



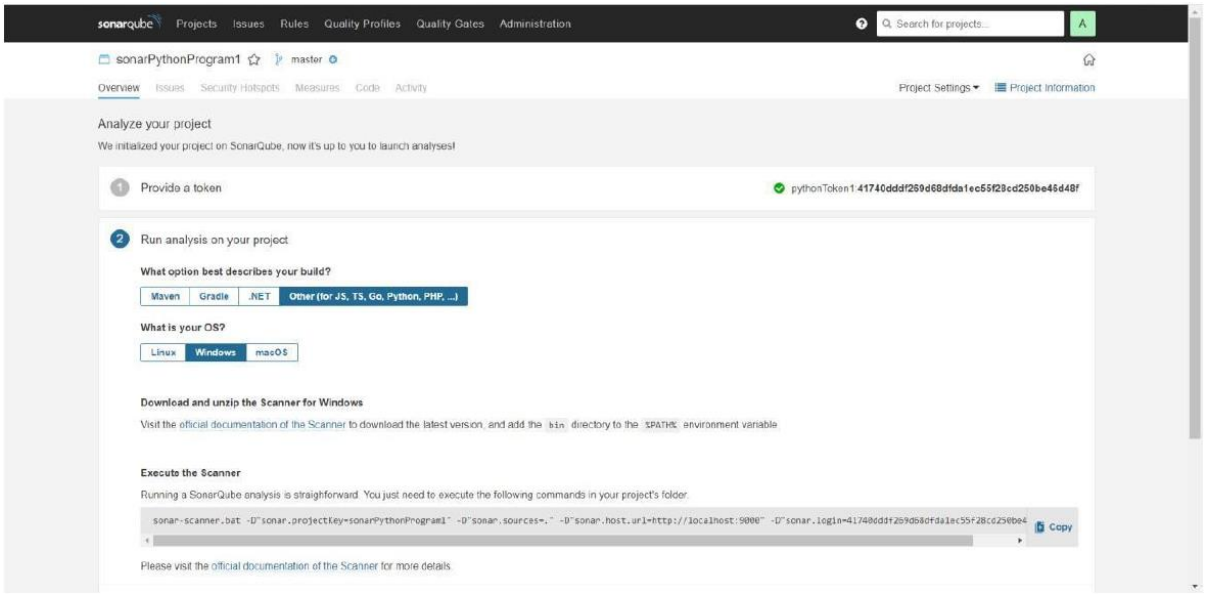
Click on **Locally**



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



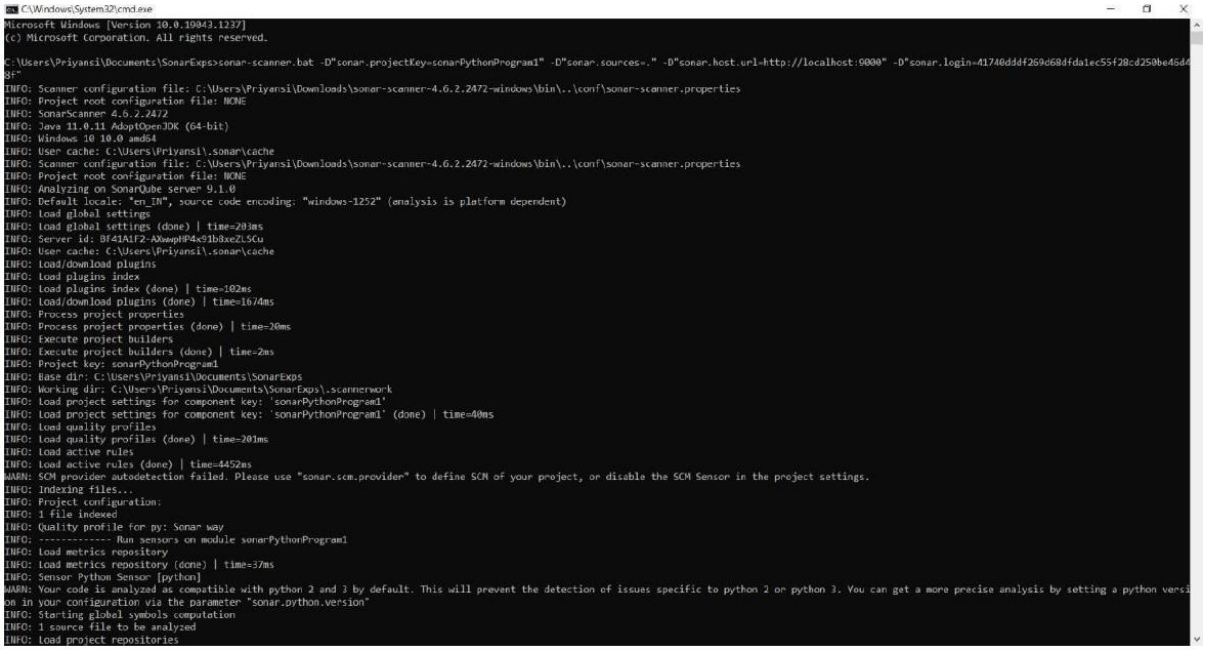
Click on **Continue.**

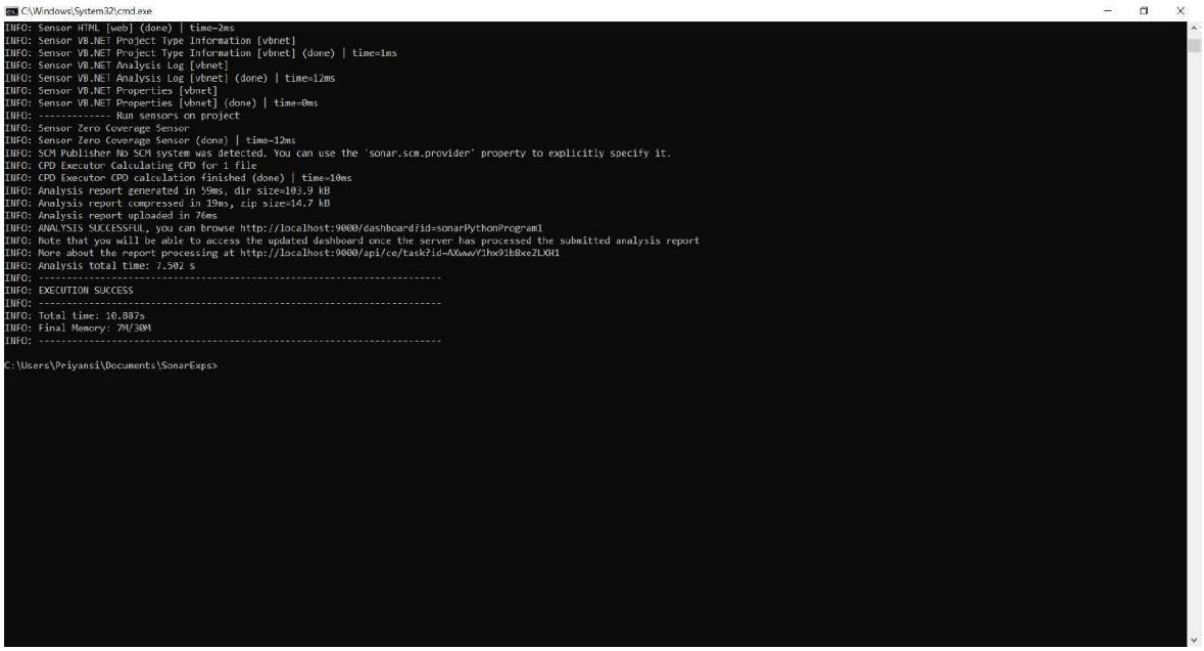


Save a Python program in a folder. class Solution(object):
def romanToInt(self, s)
roman =
{'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000,'IV':4,'IX':9,'XL':40,'XC':90,'CD':400,'CM':900}

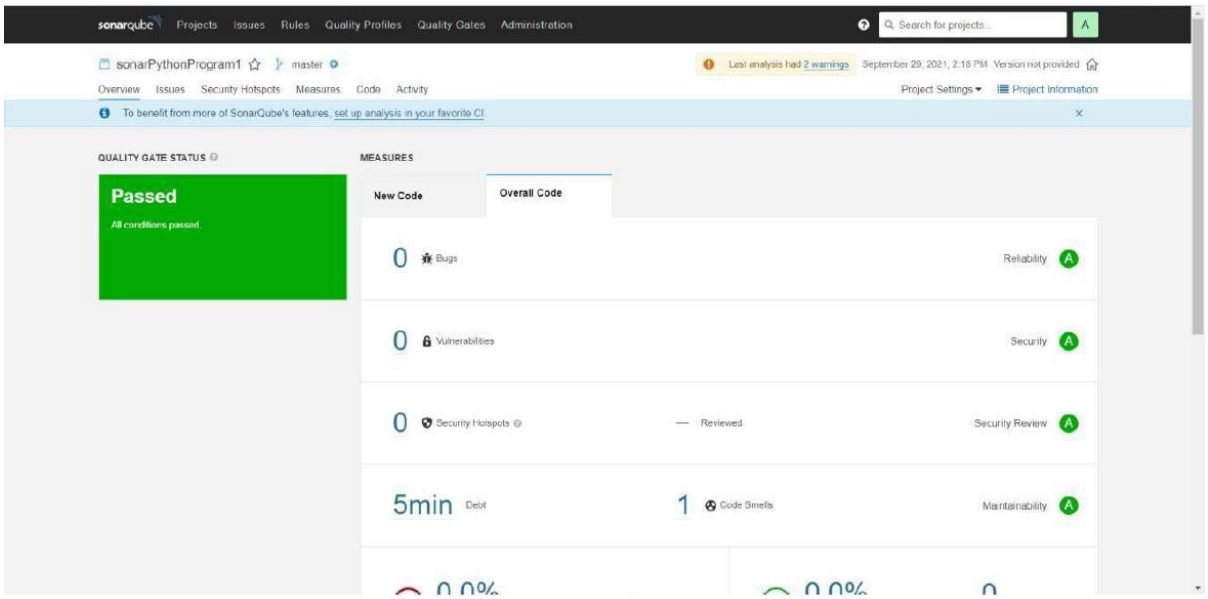
i = 0 num = " "
while i < len(s):
if i+1<len(s) and s[i:i+2] in roman:
num+=roman[s[i:i+2]] i+=2
else:
#print(i) num+=roman[s[i]] i+=1
return num ob1 = Solution()
print(ob1.romanToInt("III")) print(ob1.romanToInt("CDXLIII"))

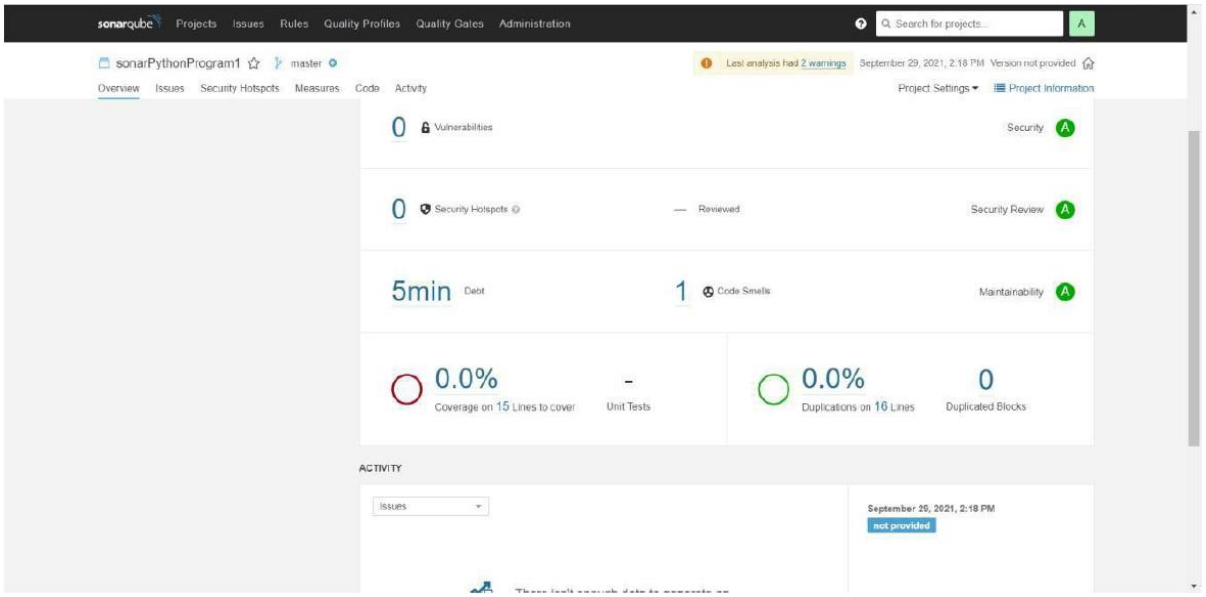
Open command prompt in this folder and Run program using copied command. “sonar-scanner.bat -D"sonar.projectKey=sonarPythonProgram1" -D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -D"sonar.login=41740dddf269d68dfda1ec55f28cd250be46d48f"

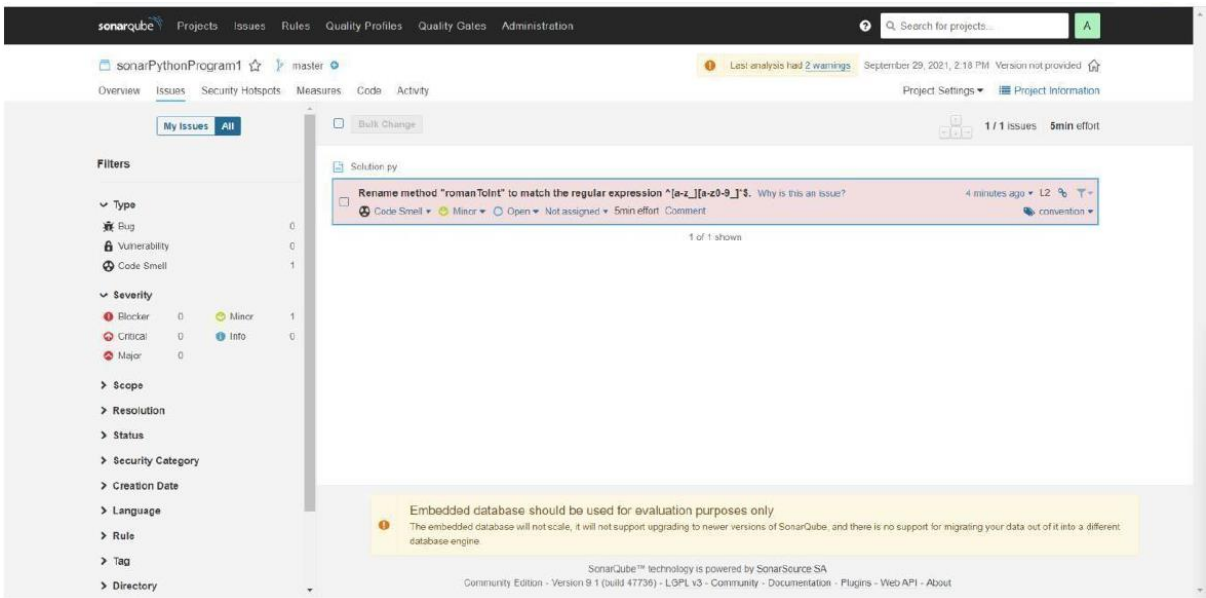




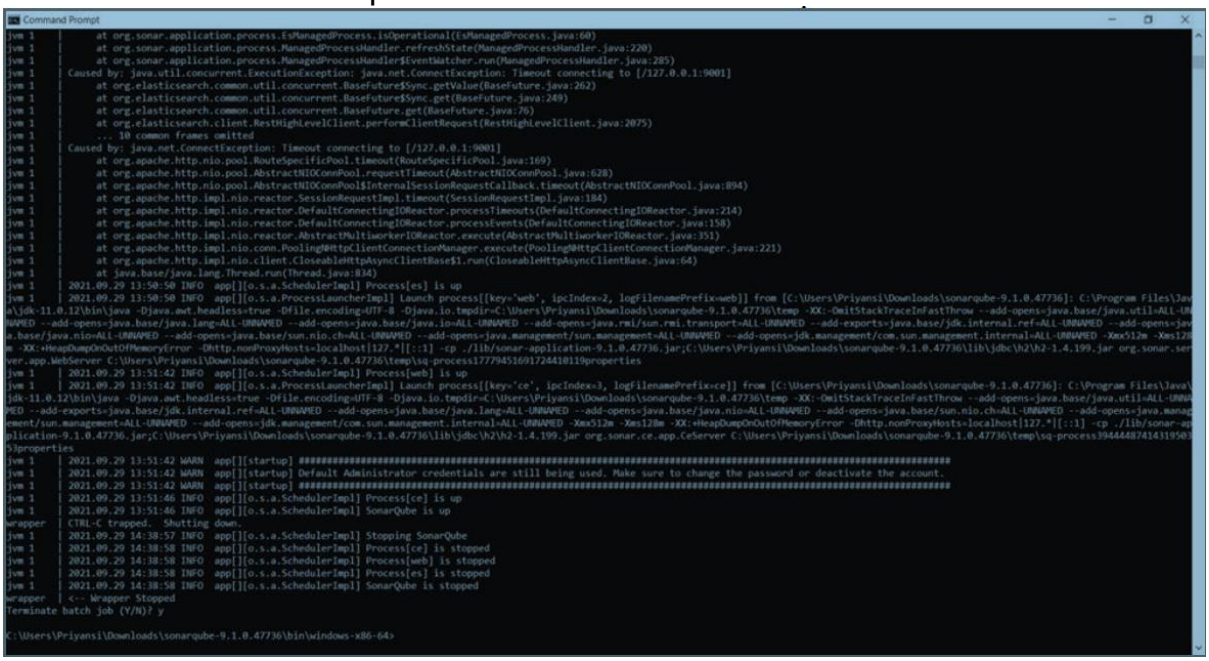
Given below is the inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
sonarPythonProgram1
master
Last analysis had 2 warnings
September 29, 2021, 2:18 PM Version not provided
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
To benefit from more of SonarQube's features, set up analysis in your favorite CI.
QUALITY GATE STATUS
Passed
All conditions passed.
MEASURES
New Code
Overall Code
0 Bugs
Reliability A
0 Vulnerabilities
Security A
0 Security Hotspots
Reviewed
Security Review A
5min Debt
1 Code Smells
Maintainability A
0.0%
0.0%
0


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
sonarPythonProgram1
master
Last analysis had 2 warnings
September 29, 2021, 2:18 PM Version not provided
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
0 Vulnerabilities
Security A
0 Security Hotspots
Reviewed
Security Review A
5min Debt
1 Code Smells
Maintainability A
0.0%
Coverage on 15 Lines to cover
-
Unit Tests
0.0%
Duplications on 16 Lines
0
Duplicated Blocks
ACTIVITY
Issues
September 29, 2021, 2:18 PM
not provided


sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
sonarPythonProgram1
master
Last analysis had 2 warnings
September 29, 2021, 2:18 PM Version not provided
Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Information
My Issues
All
Bulk Change
1 / 1 issues
5min effort
Filters
Type
Bug 0
Vulnerability 0
Code Smell 1
Severity
Blocker 0
Minor 1
Critical 0
Info 0
Major 0
Scope
Resolution
Status
Security Category
Creation Date
Language
Rule
Tag
Directory
Solution.py
Rename method "romanToInt" to match the regular expression ^[a-z_][a-z0-9_]*$. Why is this an issue?
4 minutes ago L2
Code Smell Minor Open Not assigned 5min effort Comment
convention
1 of 1 shown
Embedded database should be used for evaluation purposes only
The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.
SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 9.1 (build 47736) - LGPL v3 - Community - Documentation - Plugins - Web API - About

Press "**Ctrl + C**" to stop the server.



CONCLUSION: In this assignment, we learnt analysis of using sonarqube. The goal of SonarQube is to empower developers first and to grow an open community around the quality and security of code.

Animesh Parab T2-T21 88

ASSIGNMENT-8

AIM:-To understand continuous monitoring using Nagios

LO MAPPED:- LO1, LO5

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.

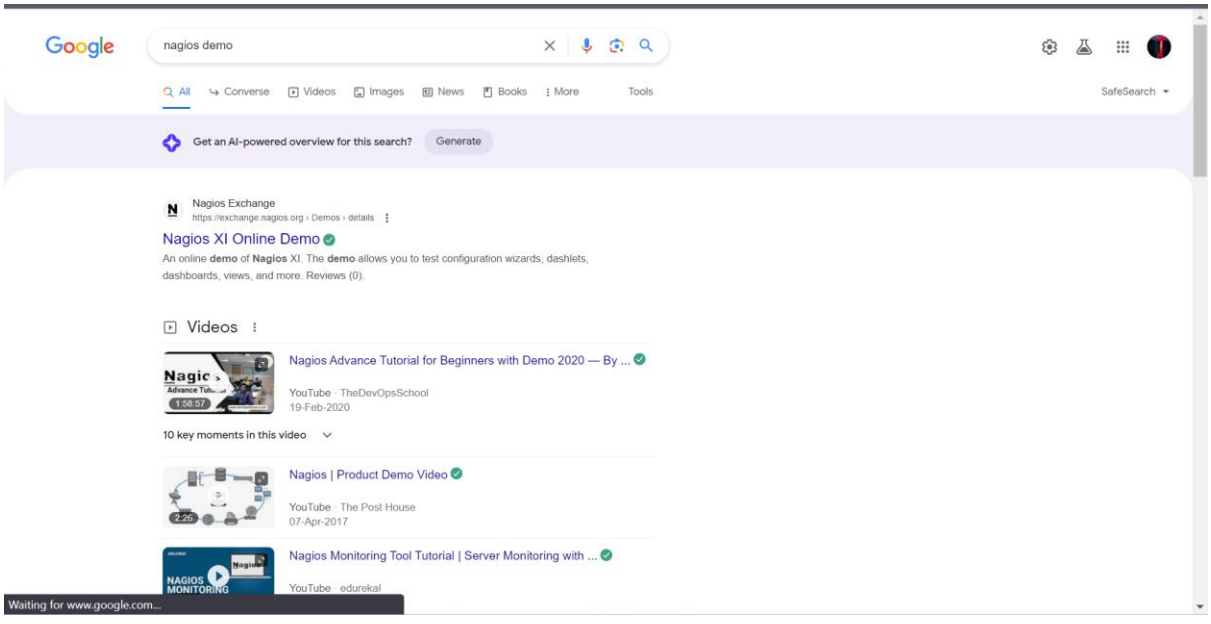
Community Support and Active Development: Being an open-source project, Nagios has a vibrant community that contributes to its development and support. This community-driven approach ensures that the software remains updated and robust.

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.

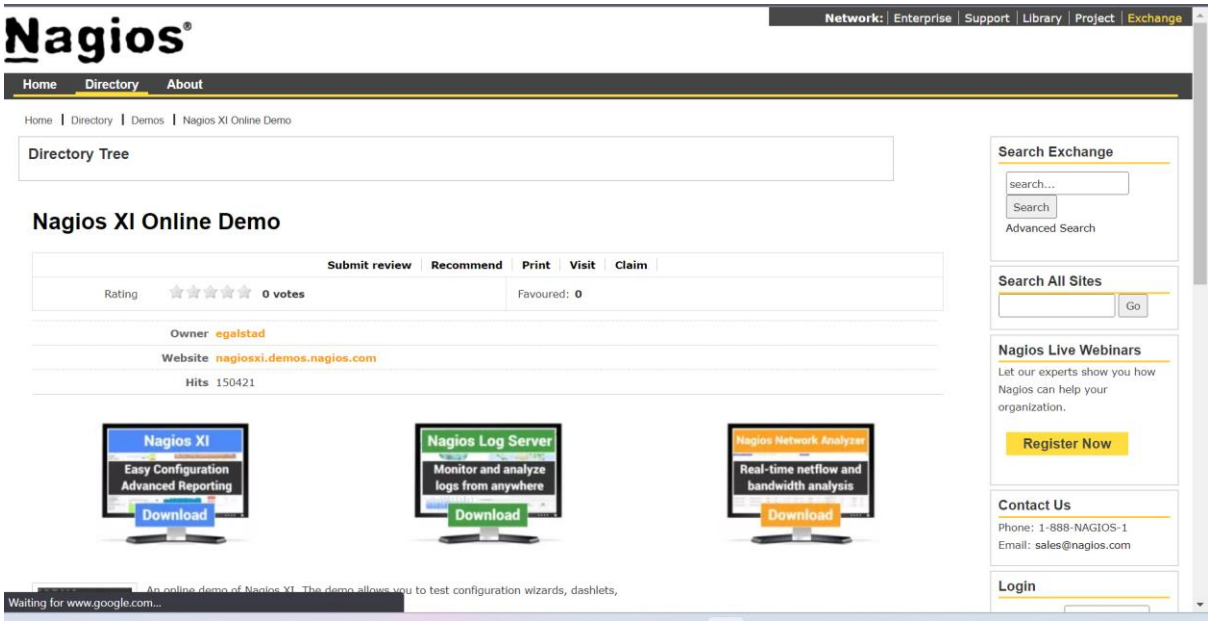
Integration with Third-Party Tools: Nagios can integrate with various third-party tools and services, making it a versatile monitoring solution that can fit into different IT ecosystems and workflows.

STEPS:-

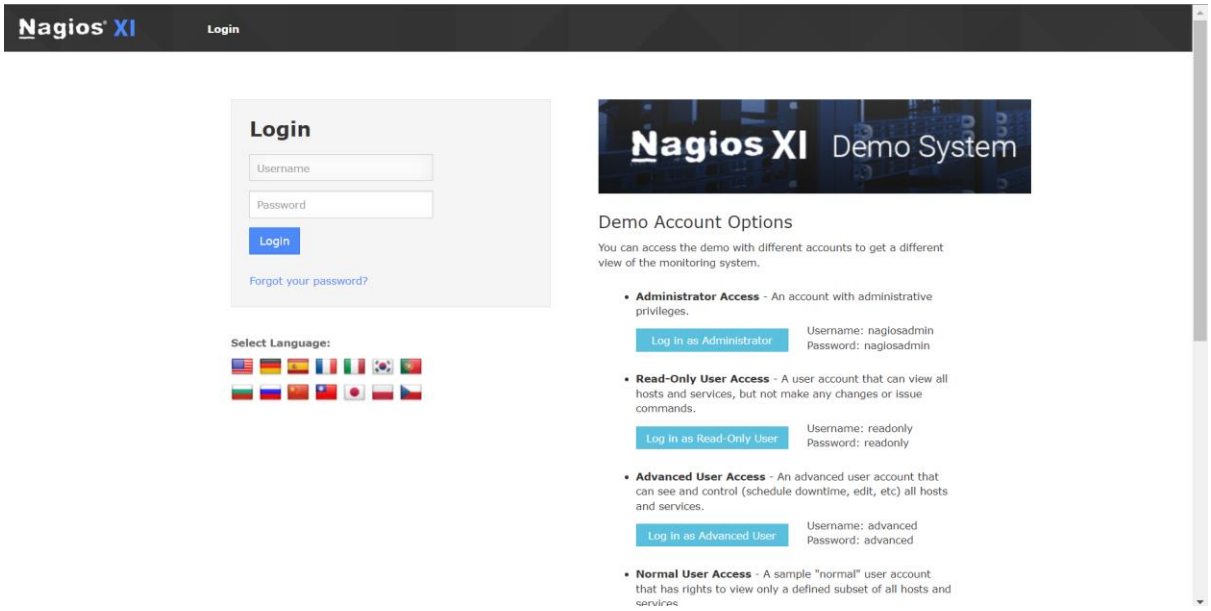
1) **Go to google.com, Search Nagios Demo**
Click on the first link shown below

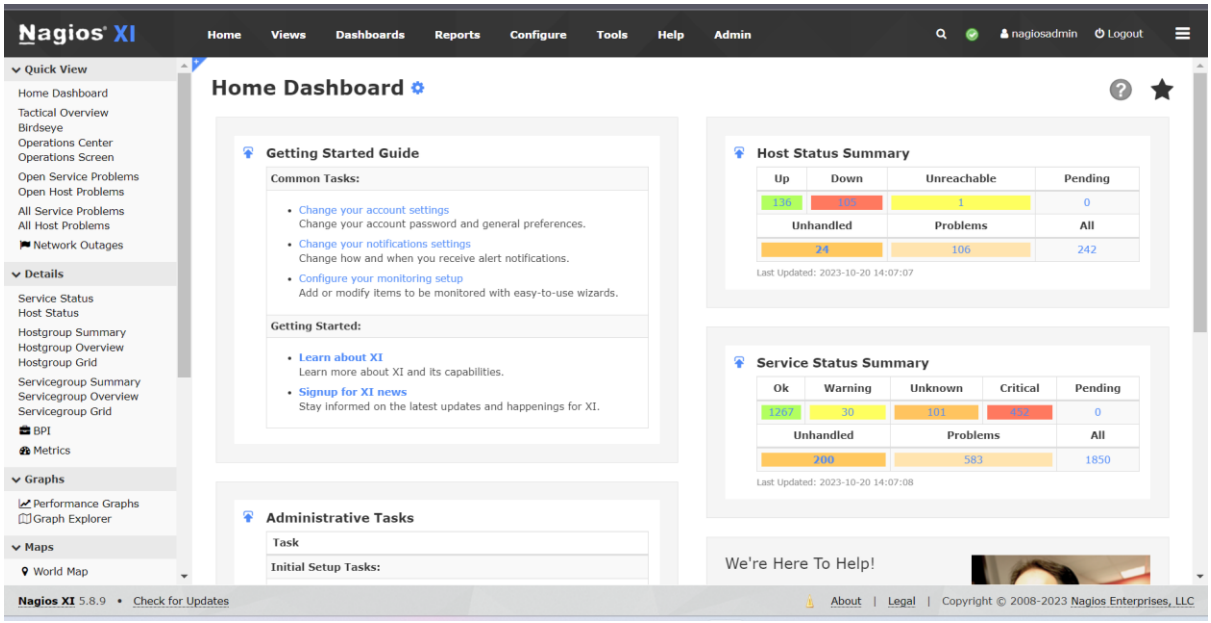


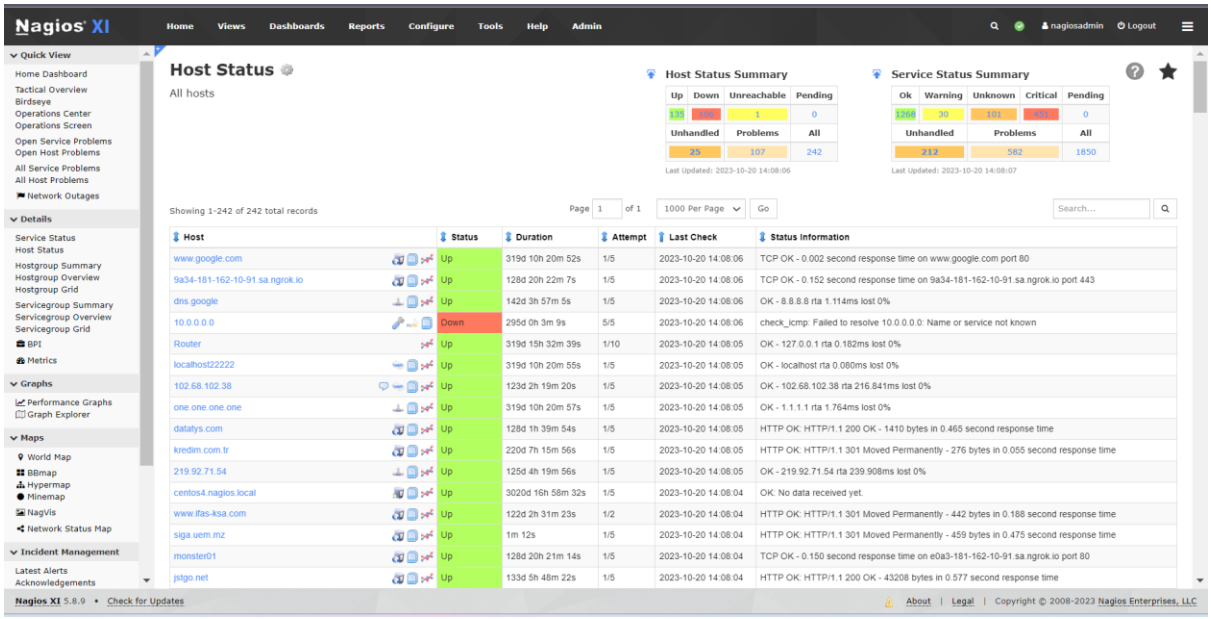
2) Now click on the website-



3) Now click on login as administrator

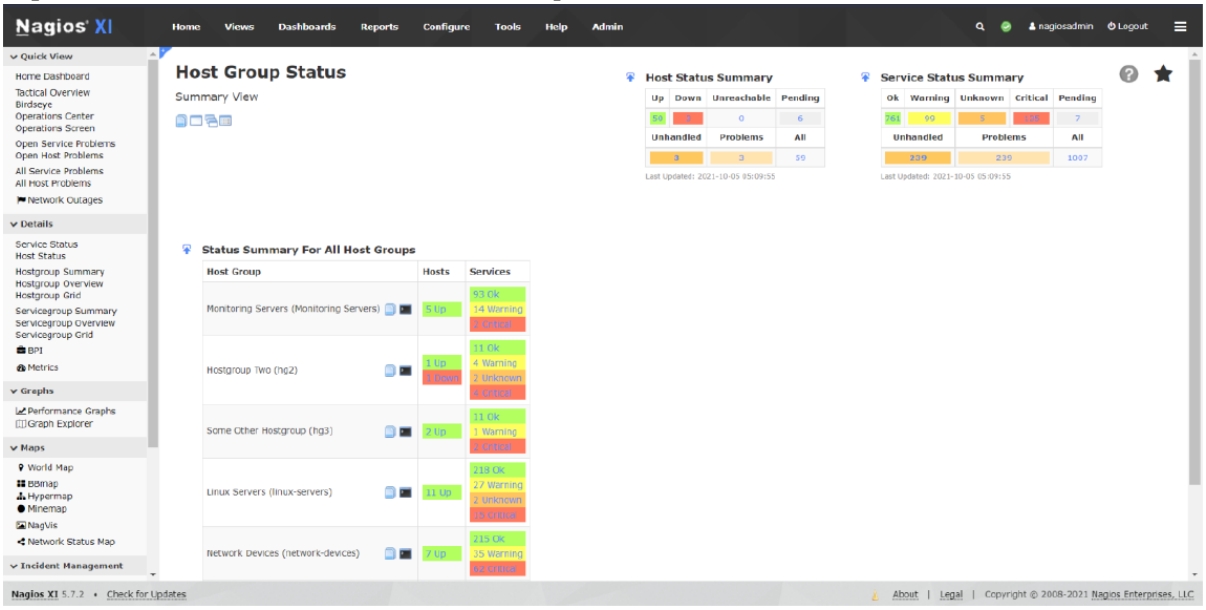




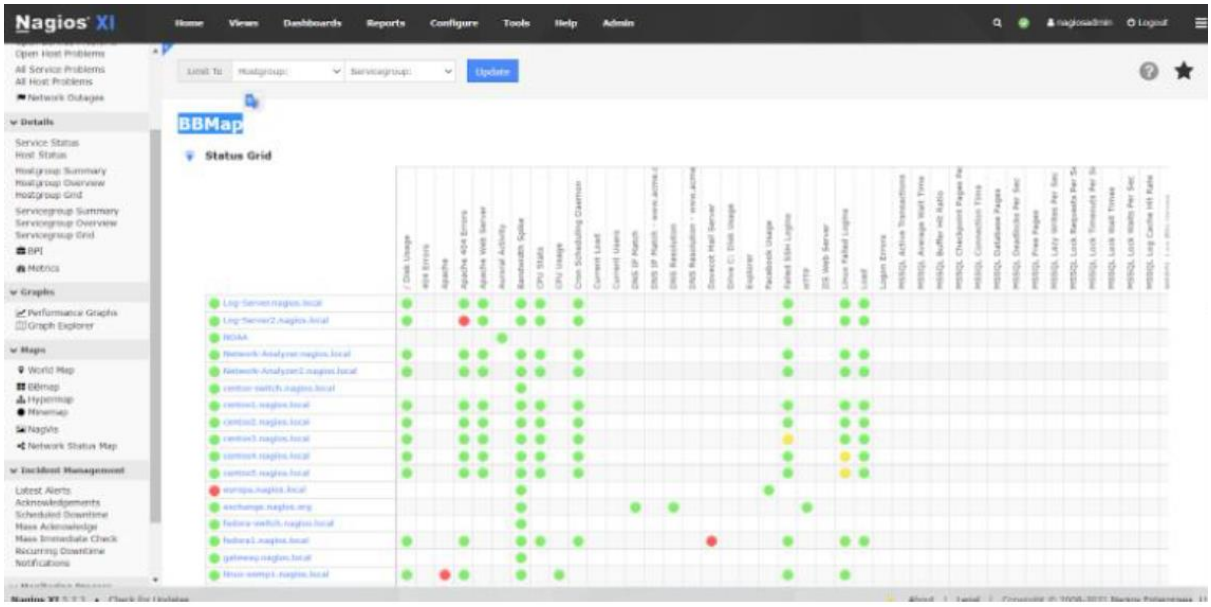


In the above image one can see Host Status Summary and Service Status Summary also how many host are up, down and also errors in detail

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-



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.

Animesh Parab T2-T21 88

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.

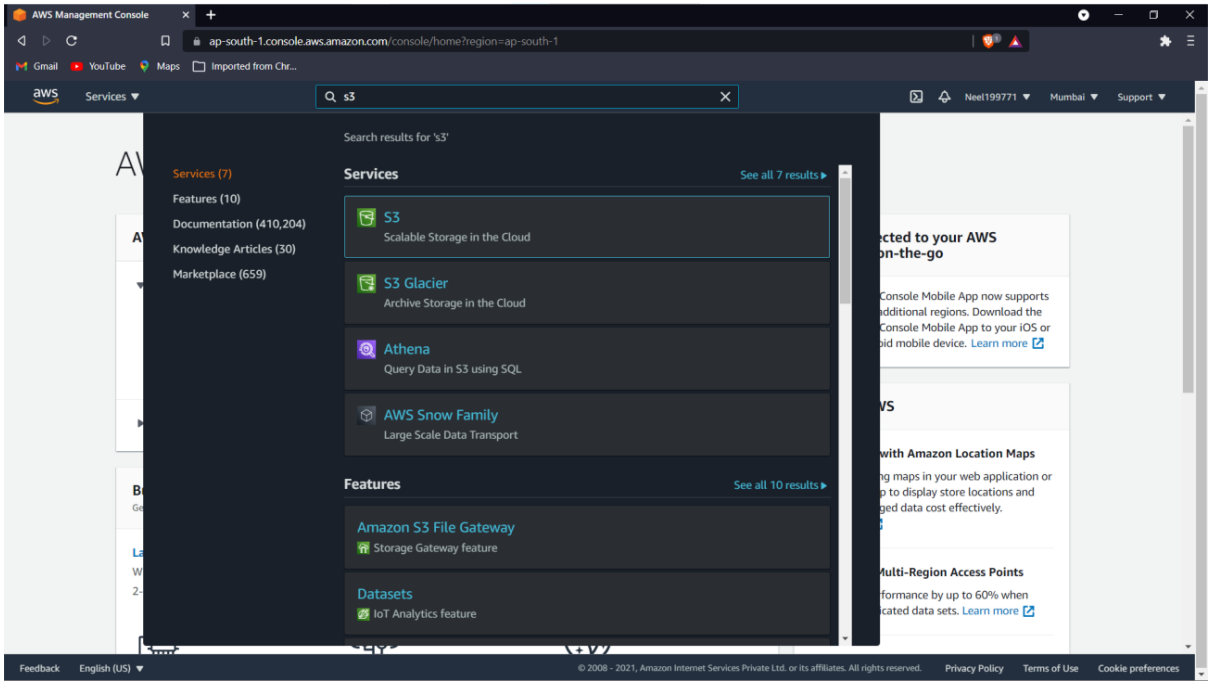
Triggers and Destinations:

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.

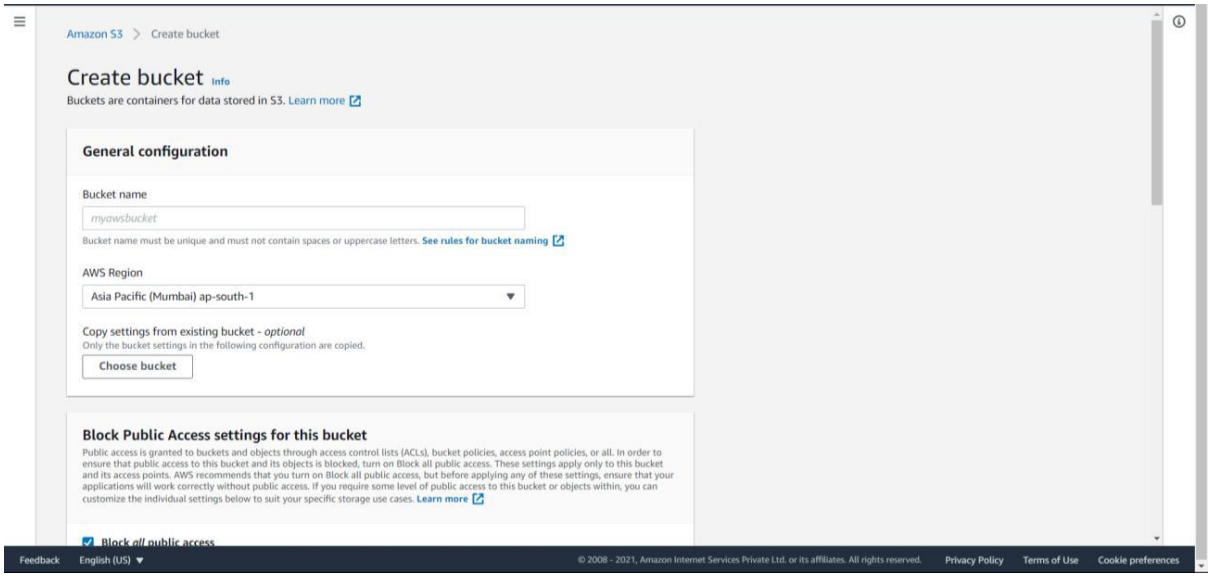
STEPS:

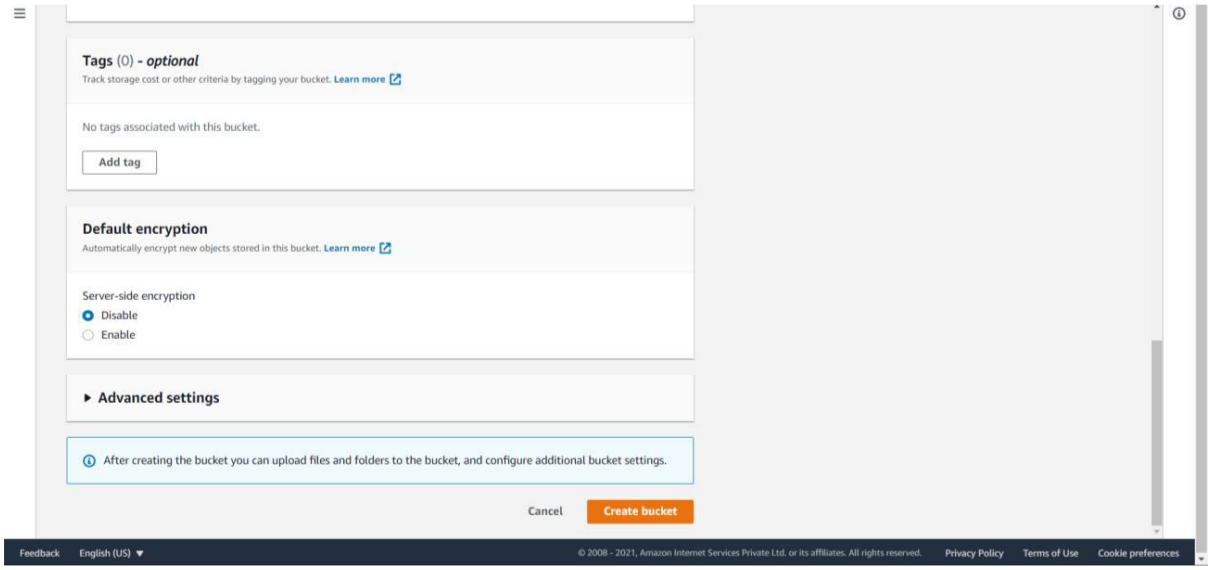
Login to Aws account-

Search S3 ,click on the option below shown-

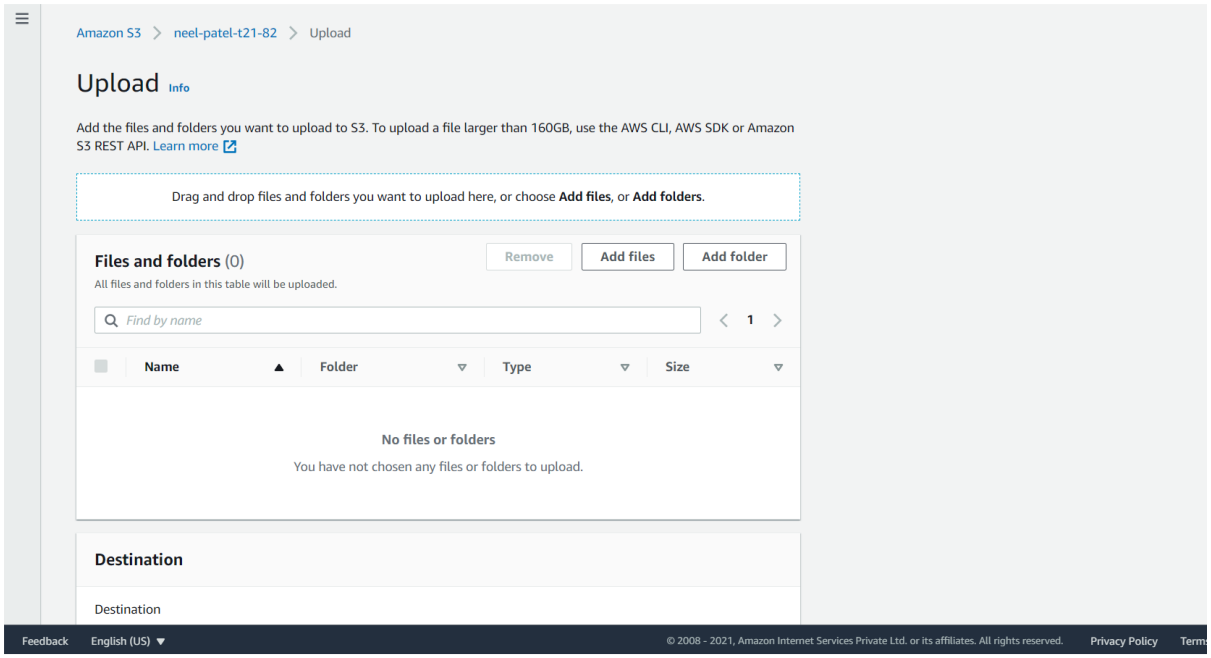


Create an S3 bucket by giving it a name

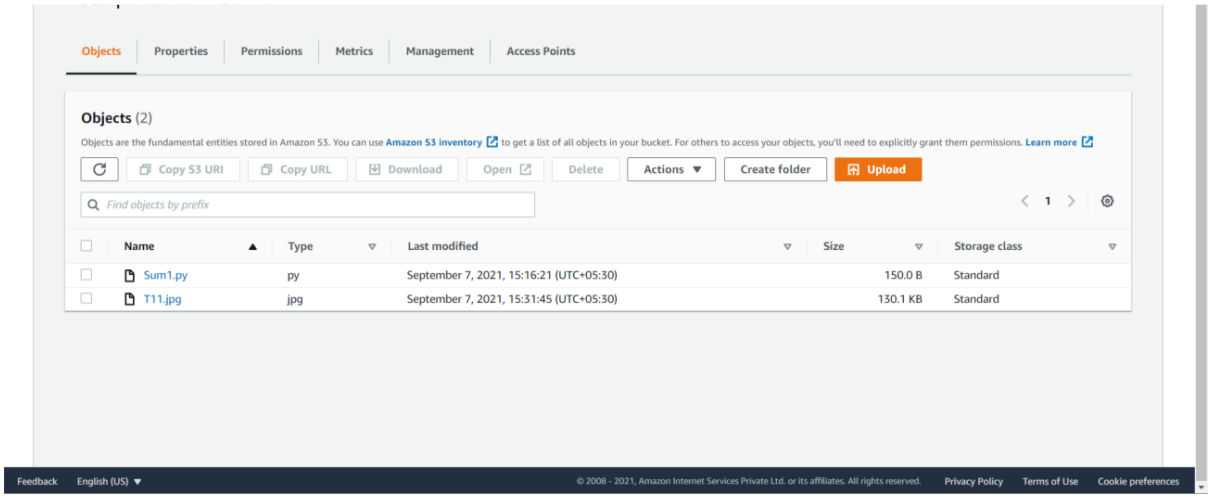




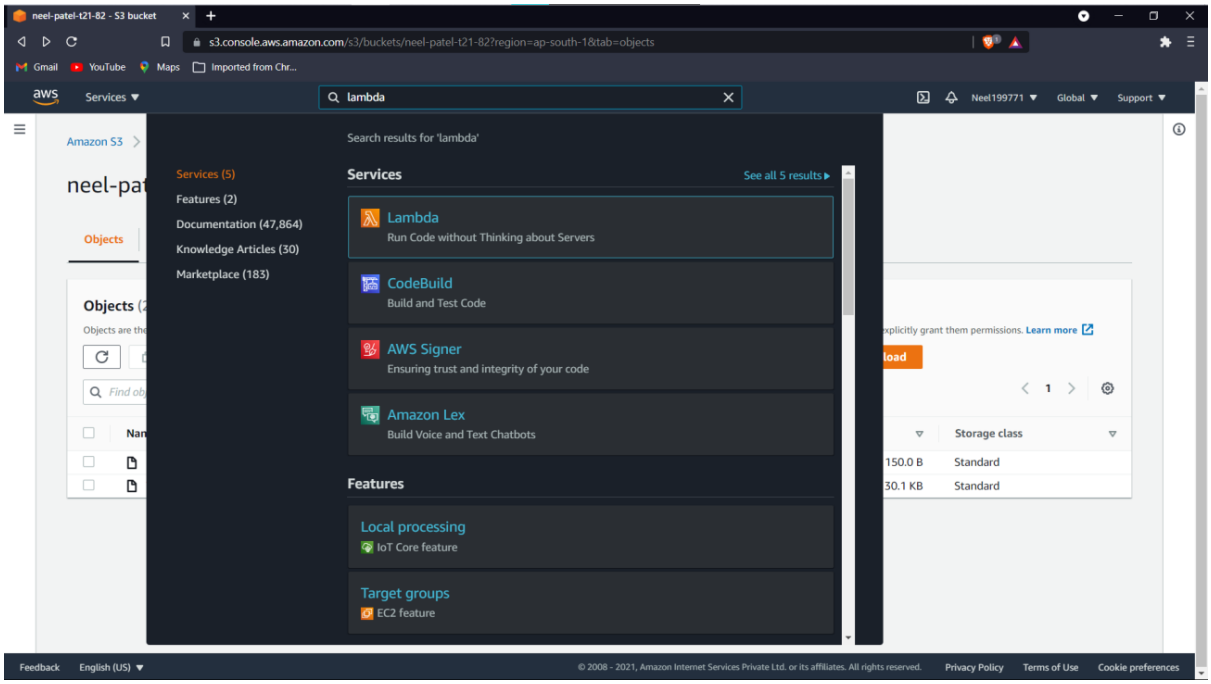
Click on upload button after the s3 bucket is created in the object section



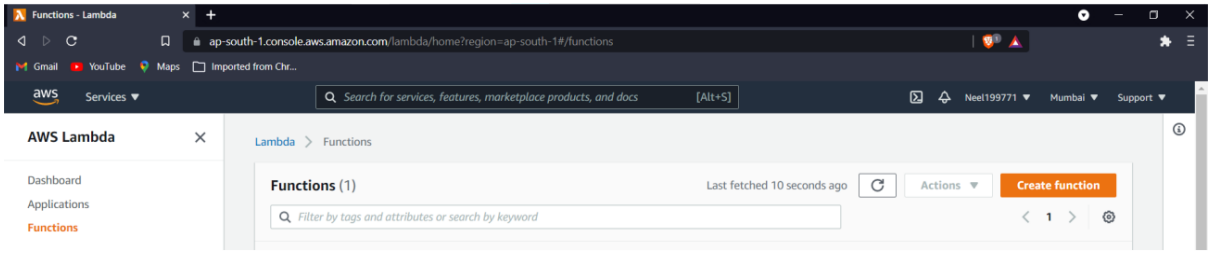
Add any .py or .java extenstion file and click on upload Now search



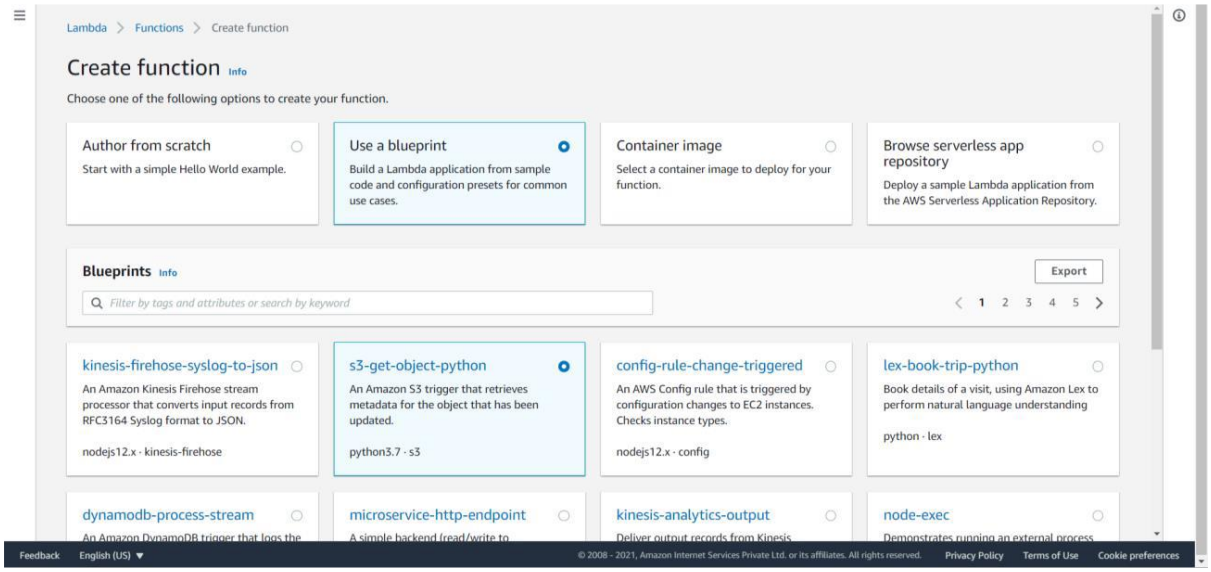
Now search Lamda

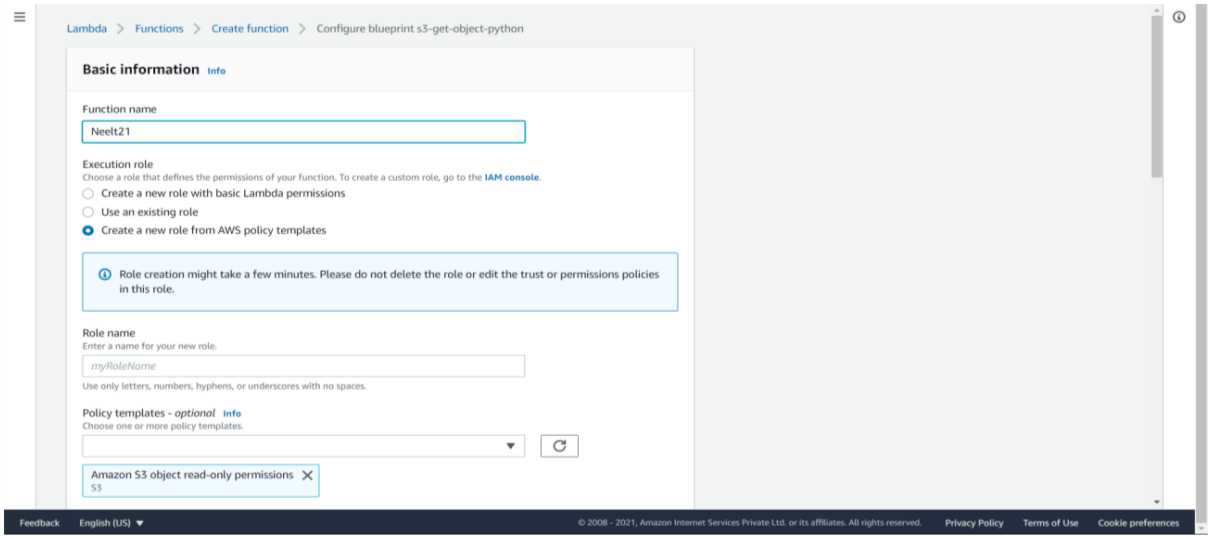


Click create function

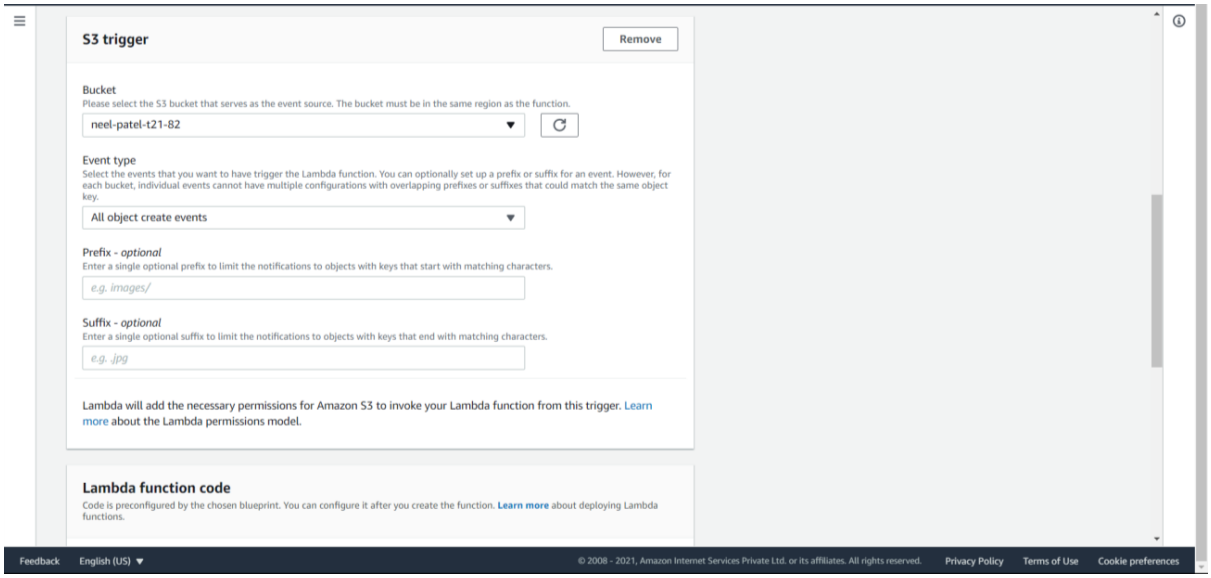


Click on below options and click on configure

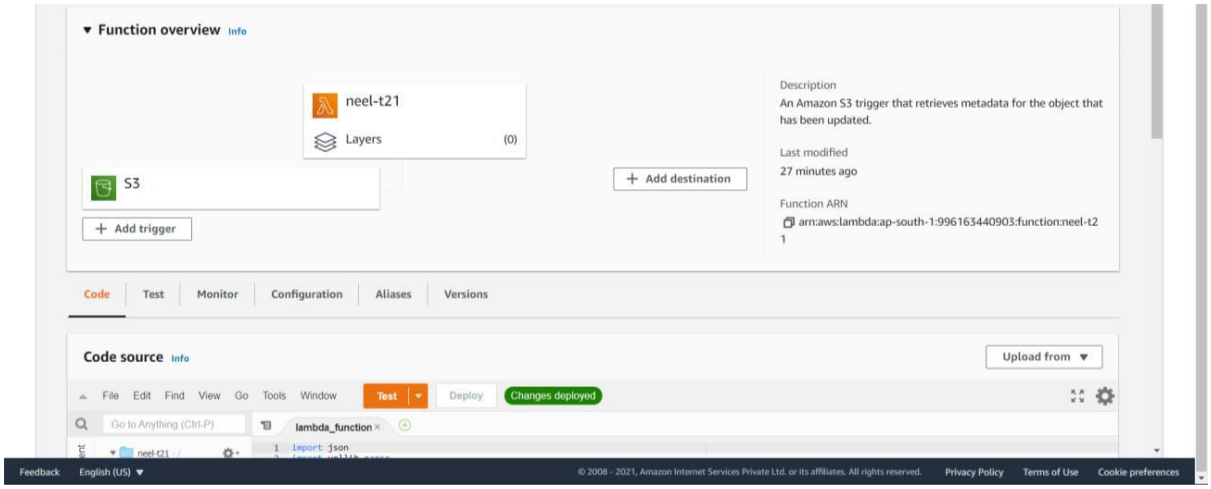




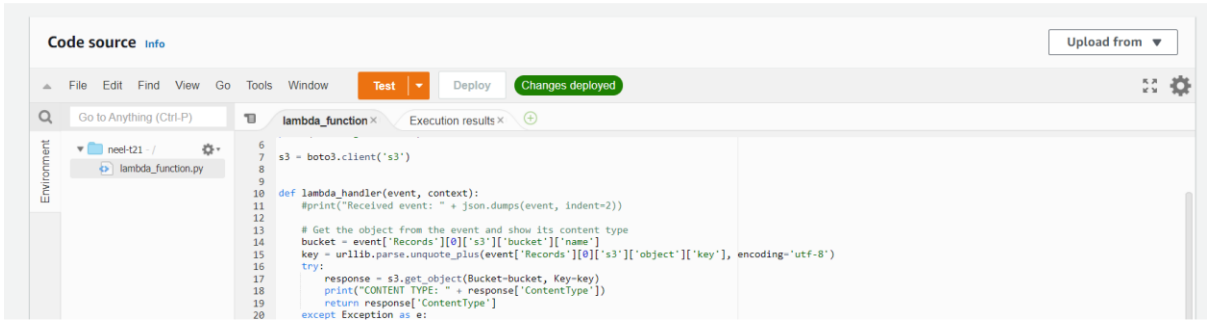
Select the bucket created and create trigger ,click on create function-
Check the given trigger is created
Click on



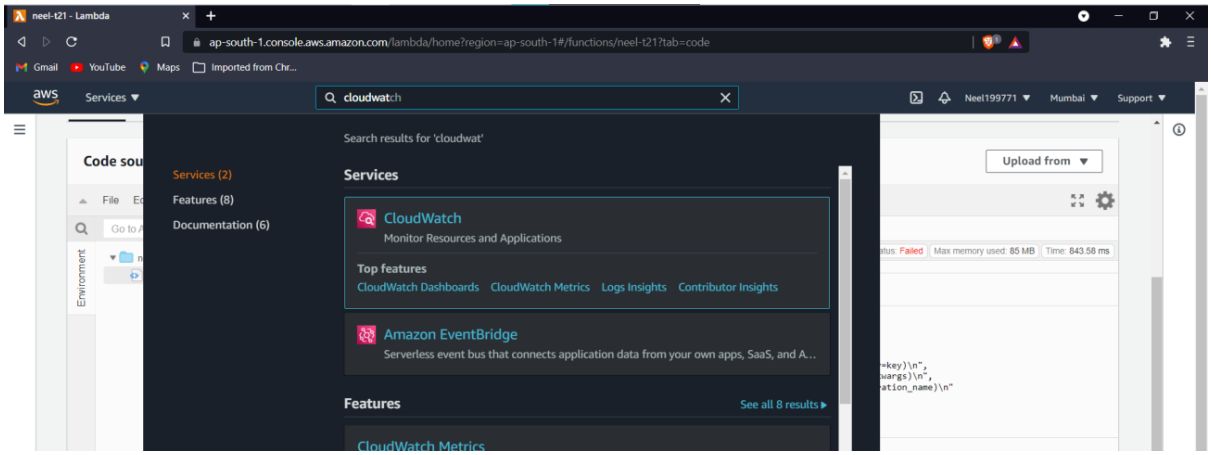
Check the given trigger is created
Click on



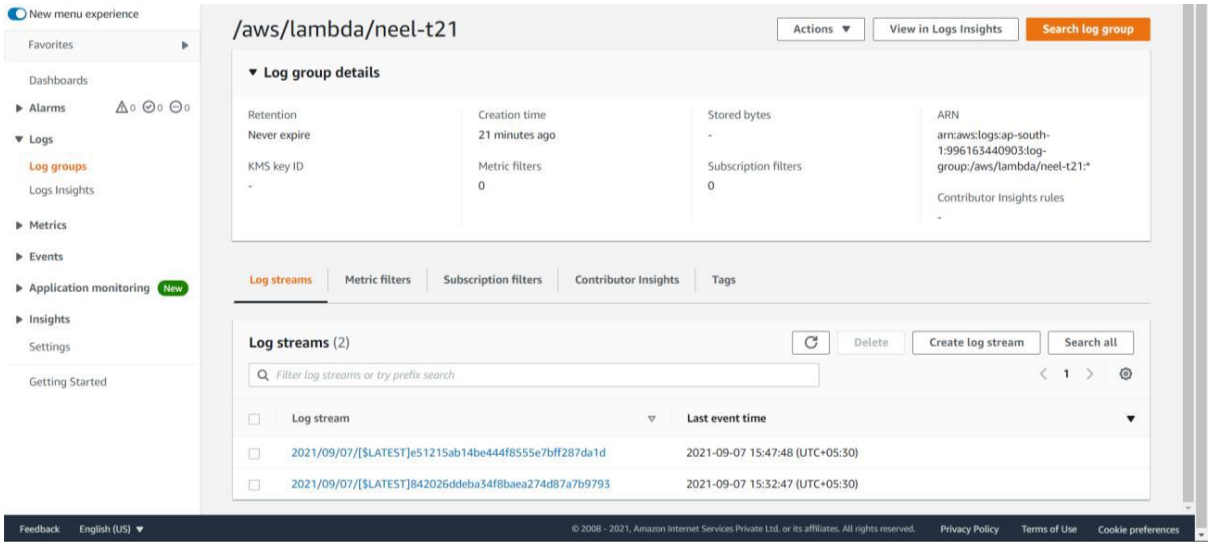
Click on the orange test button-



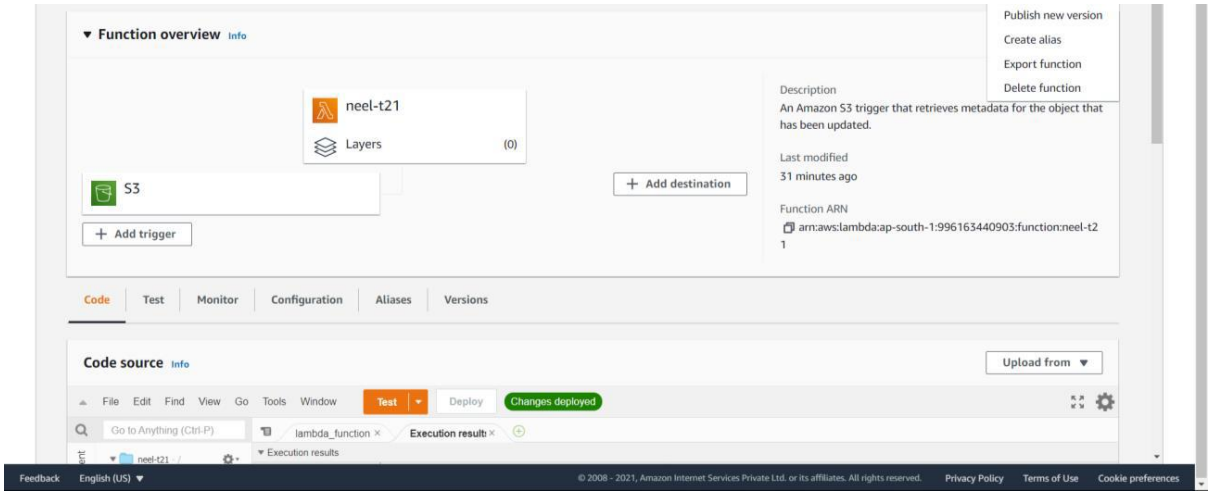
Now,

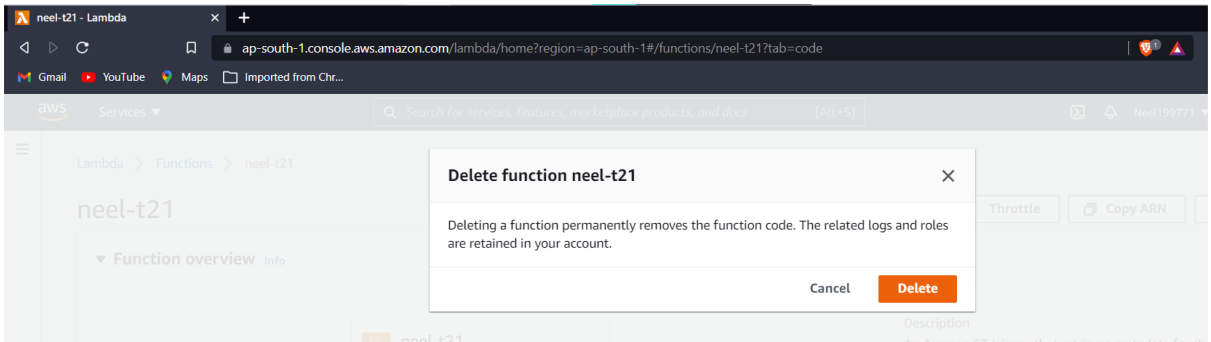


Check the logs of the test-

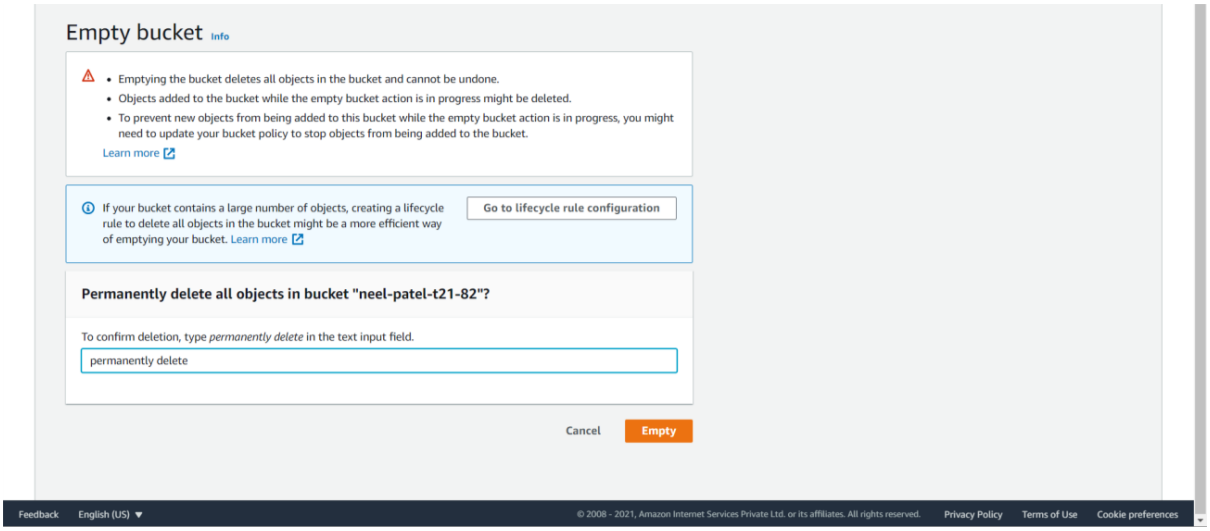


Now terminate-
Click on delete function.

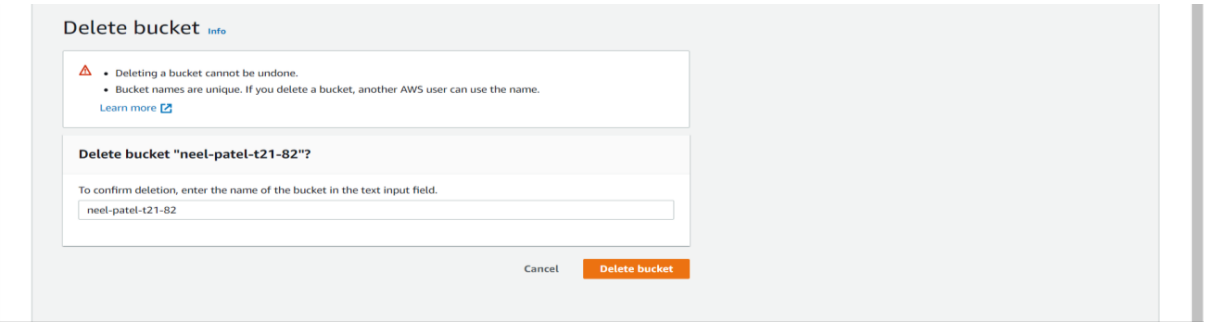




Empty bucket



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.

Animesh Parab T2-t21 88

ASSIGNMENT-10

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

LO MAPPED: LO1, LO5

THEORY:-

DYNAMO DB

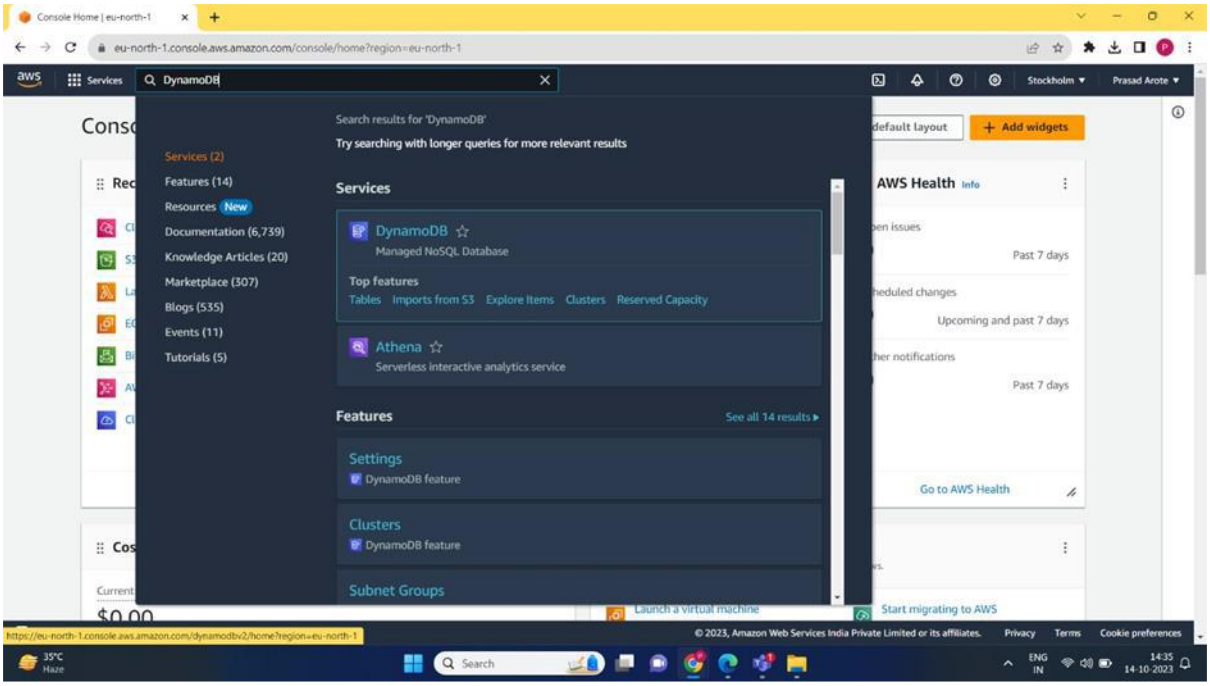
Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and 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

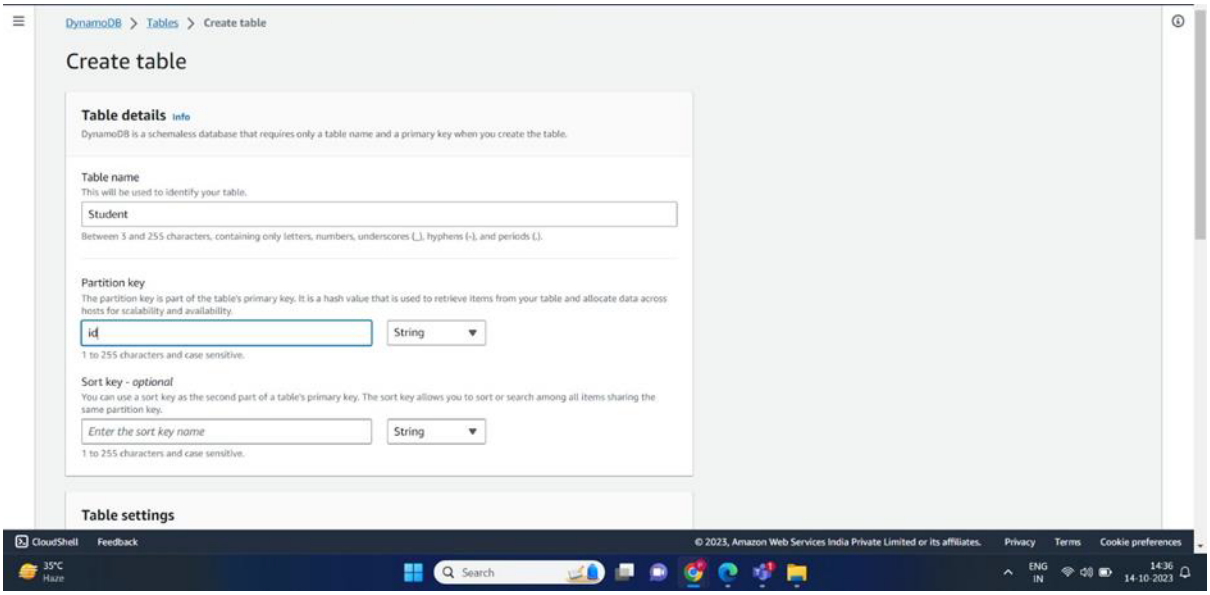
DynamoDB provides on-demand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

Steps :-

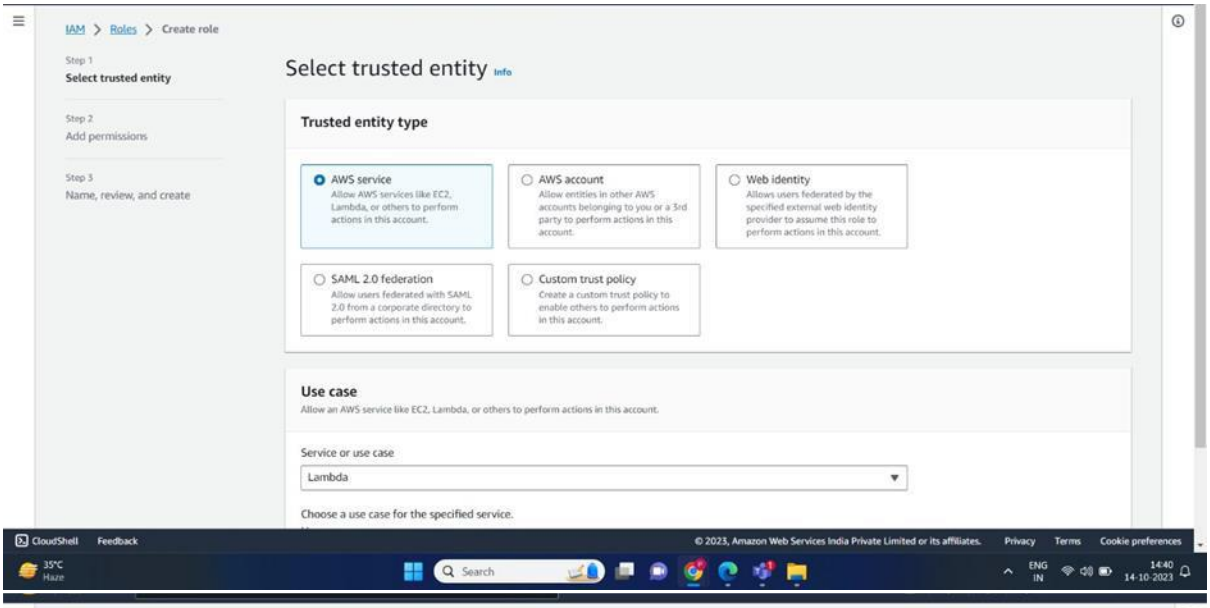
1) **Login to AWS account and search for DynamoDB in search bar**

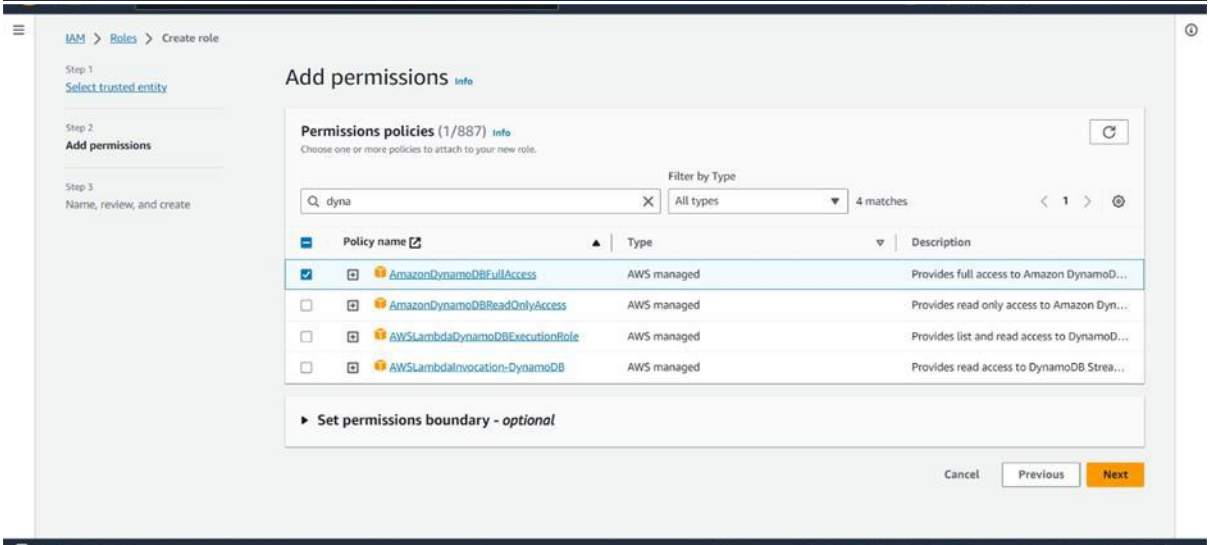


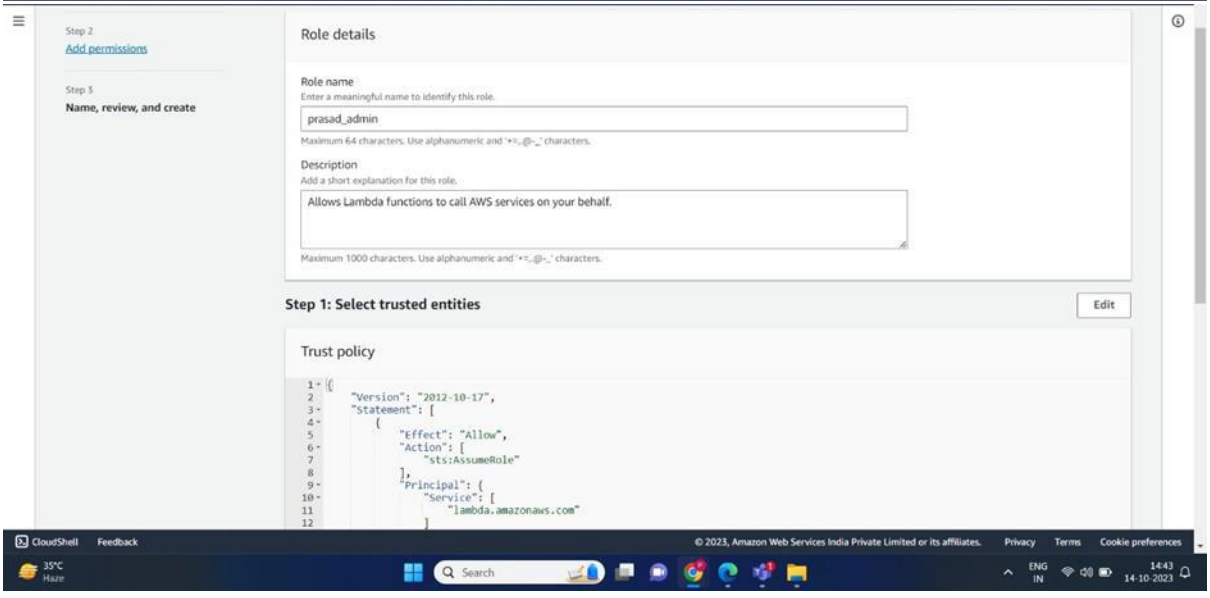
2) Click on DynamoDB option shown above and then click on create table

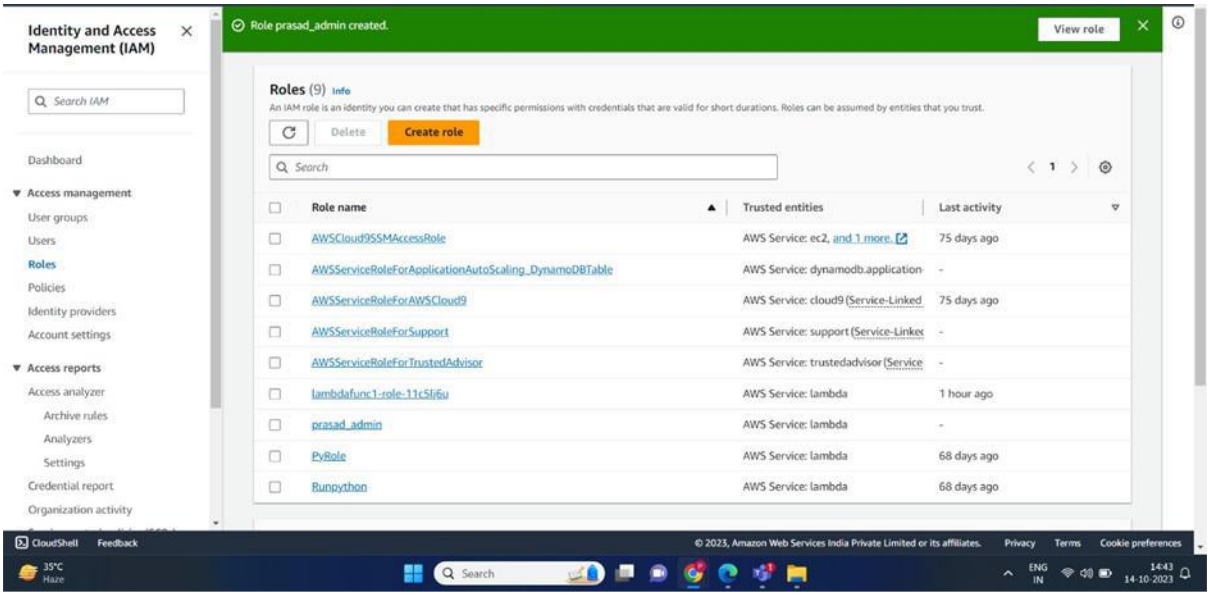


3)Then search IAM in the search box above and create a new role , give AmazonDynamoFullAccess permission to created user

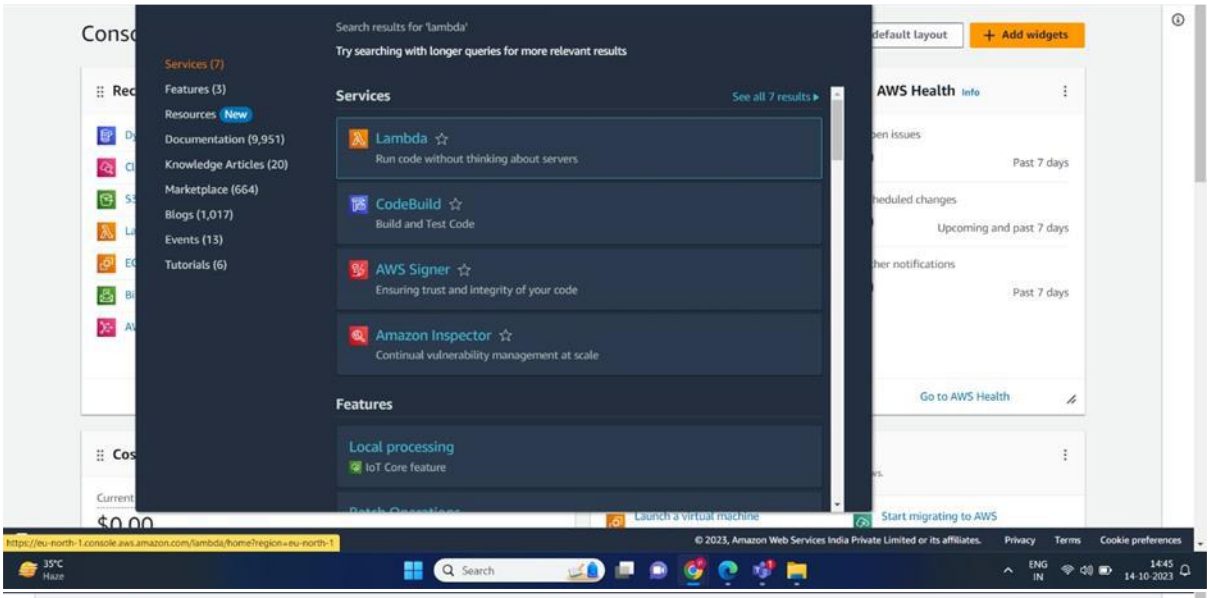

IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Select trusted entity Info
Trusted entity type
AWS service
Allow AWS services like EC2, Lambda, or others to perform actions in this account.
AWS account
Allow entities in other AWS accounts 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
Search

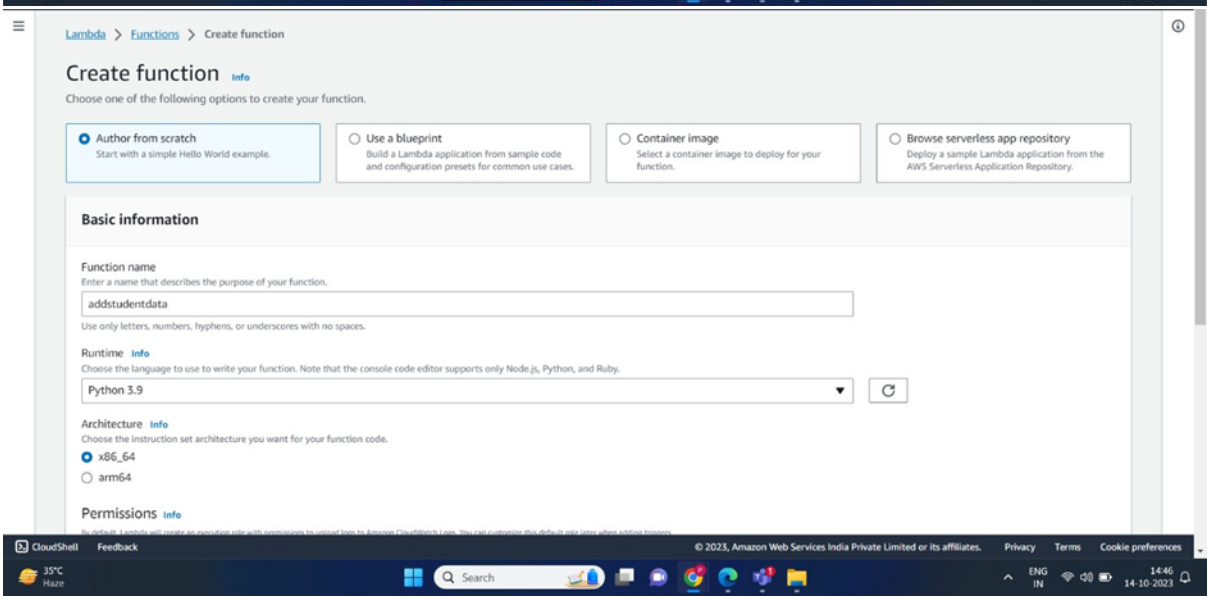

IAM > Roles > Create role
Step 1
Select trusted entity
Step 2
Add permissions
Step 3
Name, review, and create
Add permissions Info
Permissions policies (1/887) Info
Choose one or more policies to attach to your new role.
Filter by Type
dyna
All types
4 matches
Policy name
Type
Description
AmazonDynamoDBFullAccess
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
Search

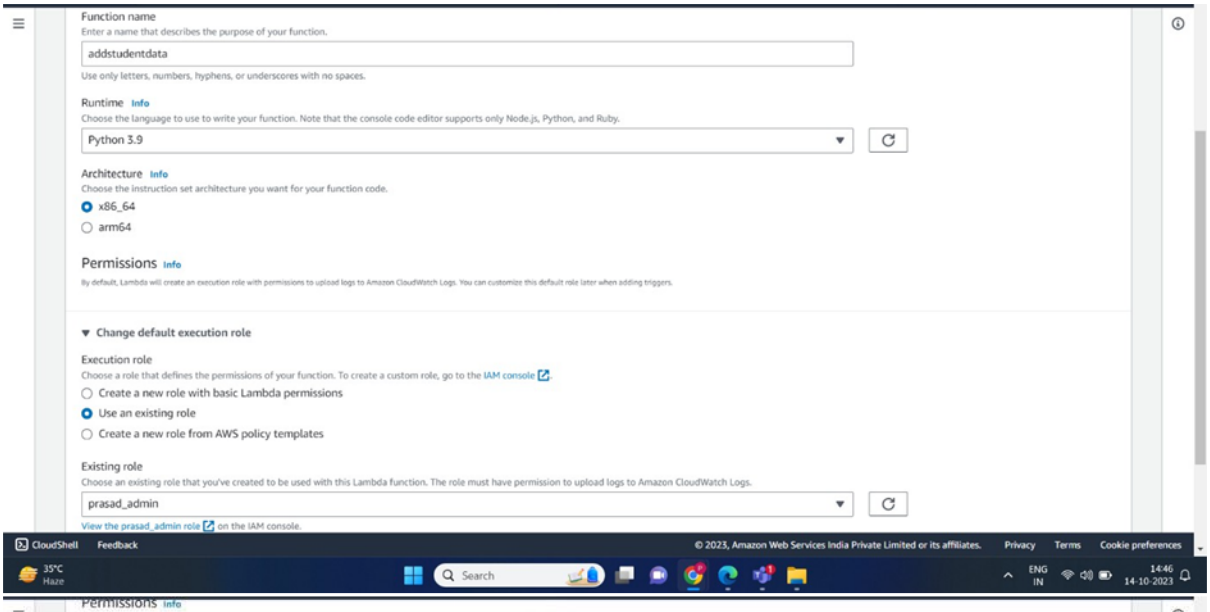

Step 2
Add permissions
Step 3
Name, review, and create
Role details
Role name
Enter a meaningful name to identify this role.
prasad_admin
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.
Description
Add a short explanation for this role.
Allows Lambda functions to call AWS services on your behalf.
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.
Step 1: Select trusted entities
Edit
Trust policy
{
"Version": "2012-10-17",
"Statement": [
{
"Effect": "Allow",
"Action": [
"sts:AssumeRole"
],
"Principal": {
"Service": [
"lambda.amazonaws.com"
]
CloudShell
Feedback
© 2023, Amazon Web Services India Private Limited or its affiliates.
Privacy
Terms
Cookie preferences
Search

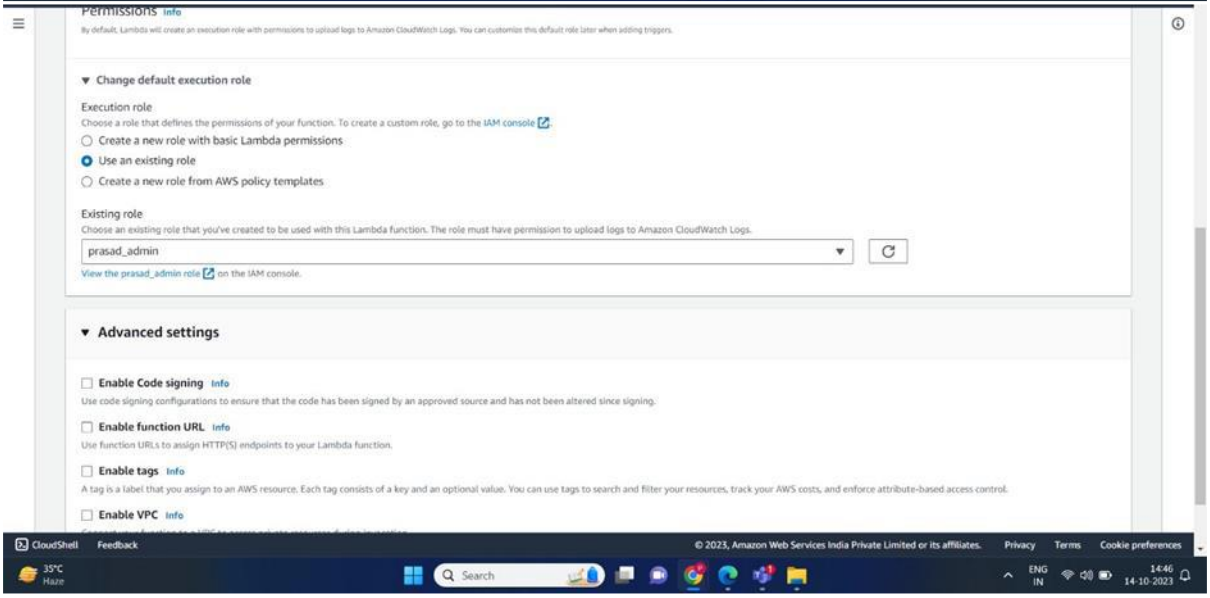


3) Search Lambda in search box and click on it , then create a new lambda function

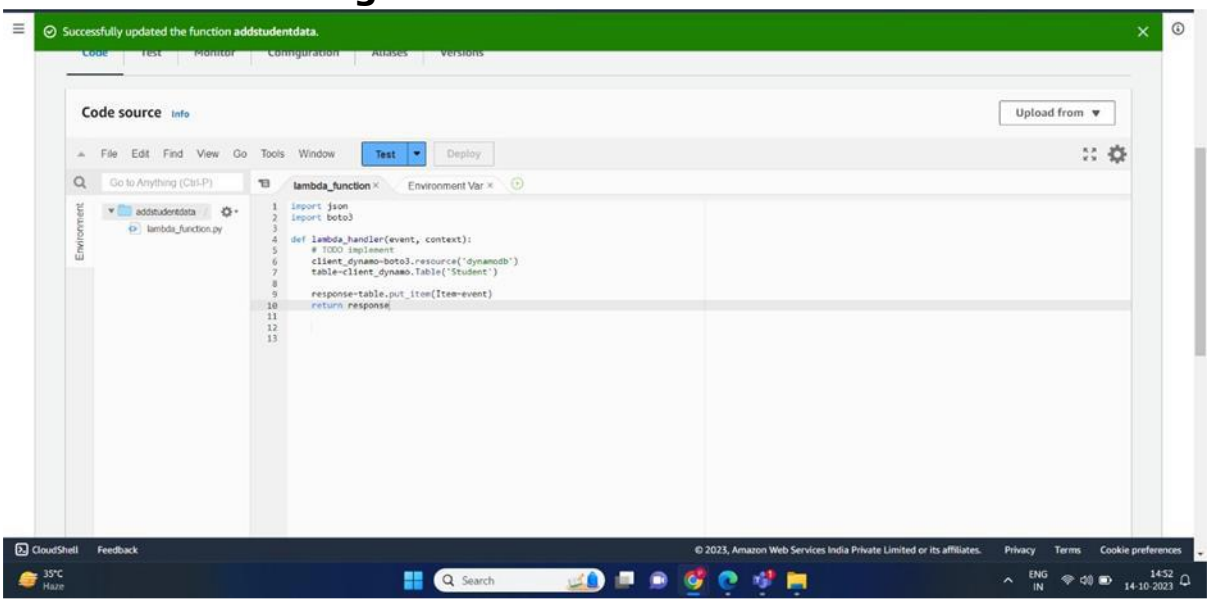




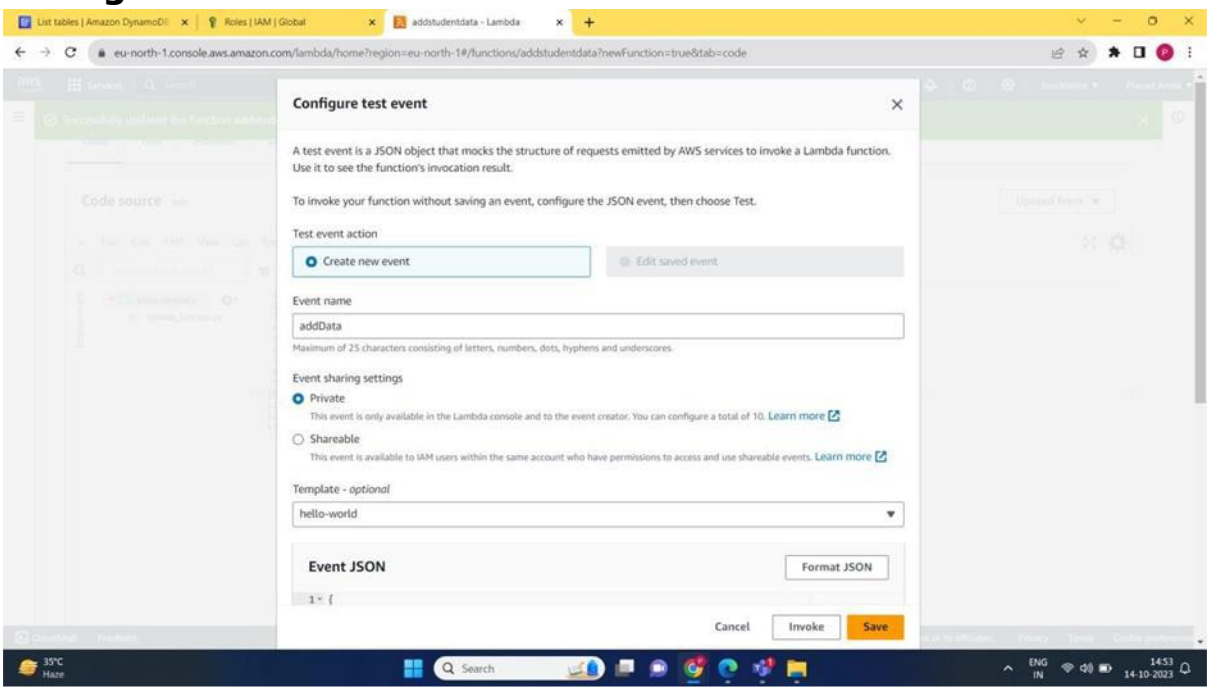


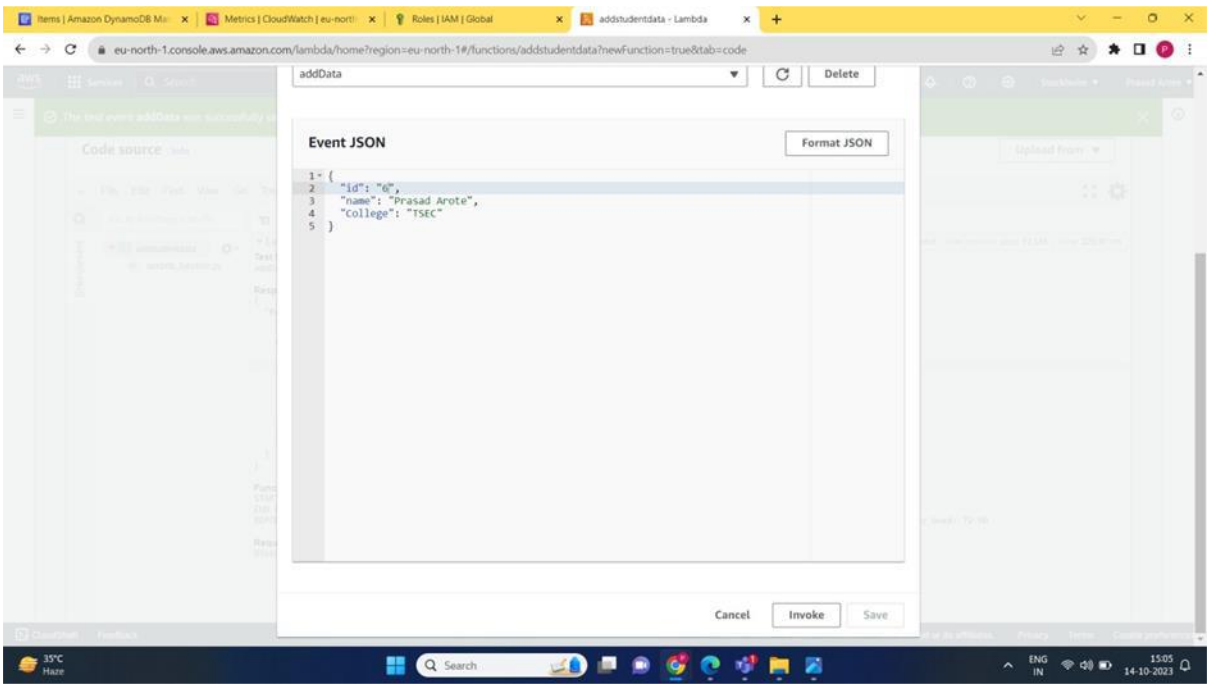


4) Write the following code in code source

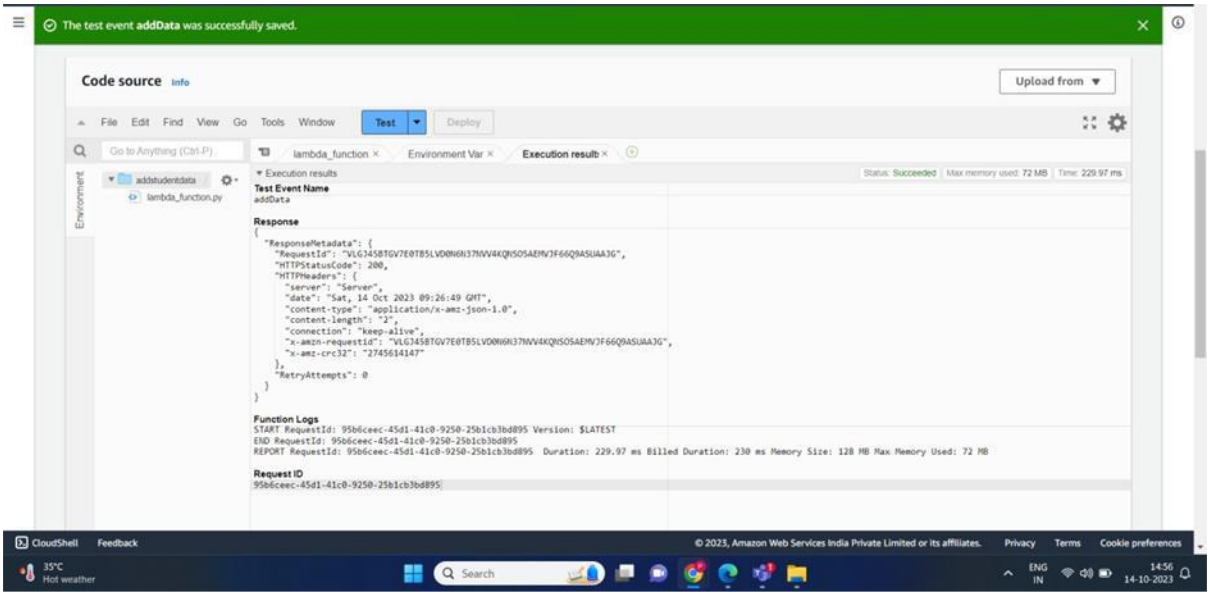


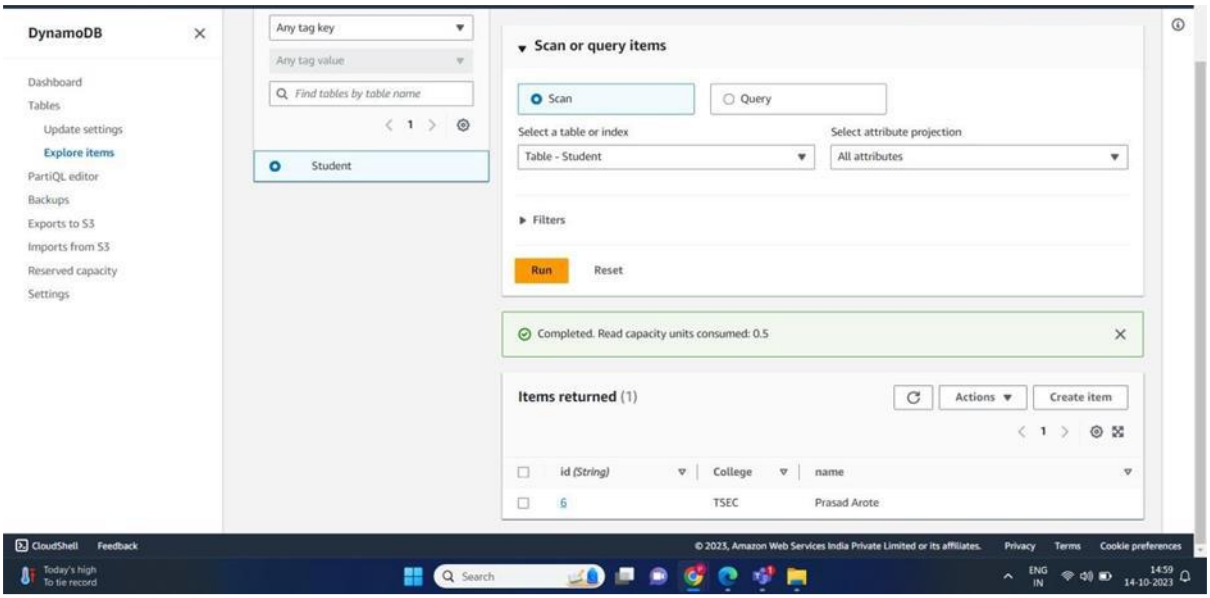
5) Configure the test event and save





6) 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

Animesh Parab T2-T21 88

Adv. DevOps Written Assignment : 01

1. what security measures can be taken while using Kubernetes?

1. Role-Based Access Control (RBAC): RBAC restricts who can perform actions within a Kubernetes cluster. It defines roles and role bindings to specify what resources and operations users or service accounts can access. This prevents unauthorized access and actions within the cluster.

2. Regular Updates: Keeping Kubernetes and its components up to date is crucial. New releases often include security patches. Regular updates help mitigate known vulnerabilities and ensure your cluster remains secure.

3. Network Policies: Network policies allow you to define rules for communication between pods. By specifying which pods can communicate with each other, you can limit the attack surface and prevent unauthorized access.

4. Container Security Tools: Employ container security tools like vulnerability scanners to assess the security of container images. These tools can identify and remediate vulnerabilities in the containerized applications before they are deployed.

5. Monitoring and Audit: Implement monitoring and auditing solutions to track cluster activity. This helps detect and respond to suspicious or unauthorized behavior. Tools like Prometheus and Grafana can be used for monitoring, while audit logs provide insights into cluster activity.

6. Secrets Management: Sensitive data like API keys, passwords, and certificates should be stored securely using Kubernetes secrets or external vaults. This prevents sensitive information from being exposed within containers or configuration files.

7. PodSecurityPolicies (PSP): PSP is a Kubernetes feature that enforces security policies at the pod level. It allows you to define restrictions on privilege escalation, host access, and other security-sensitive configurations for pods.

8. Namespaces: Use Kubernetes namespaces to logically isolate workloads. This provides a level of separation between different applications or teams, reducing the risk of unauthorized access or interference between them.

9. Admission Controllers: Admission controllers are webhook plugins that intercept and validate requests to the Kubernetes API server. You can use them to enforce custom policies and ensure that only compliant resources are admitted to the cluster.

10. Container Runtime Security: Implement container runtime security solutions like Docker Security Scanning or container runtime protection tools. These tools monitor containers at runtime for abnormal behavior, helping to detect and respond to potential threats.

Combining these measures into a comprehensive security strategy is essential for safeguarding your Kubernetes cluster and the applications running within it. It's important to stay informed about best practices and evolving security threats in the Kubernetes ecosystem.

2. What are the three security techniques that can be used to protect data?

Three security techniques commonly used to protect data are:

1. **Encryption:** Encryption is the process of converting data into a secure format that can only be read by someone with the decryption key. It ensures that even if unauthorized parties access the data, they cannot understand it without the correct key. Two common types of encryption are:

- Data-at-rest Encryption: Protects data when it's stored on disk or in a database.
- Data-in-transit Encryption: Secures data as it's transmitted between systems over networks.

2. **Access Control:** Access control mechanisms regulate who can access data and what actions they can perform on it. This involves setting permissions, roles, and policies to ensure that only authorized users or applications can access and manipulate data. Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC) are commonly used access control models.

3. **Data Masking/Redaction:** Data masking or redaction involves obscuring or replacing sensitive data with fictitious or scrambled values. This is often used in non-production environments or when sharing data with third parties. It ensures that even if someone gains access to the data, they cannot see the actual sensitive information.

These techniques are often used in combination to create a layered approach to data security, providing multiple levels of protection to safeguard sensitive information from unauthorized access and disclosure.

3. How do you expose a service using ingress in Kubernetes?

To expose a service using Ingress in Kubernetes, you need to follow these steps:

1. **Set up Kubernetes:** Ensure you have a Kubernetes cluster up and running, and you have the `kubectl` command-line tool configured to communicate with the cluster.

2. **Deploy Your Application:** Deploy your application as a Kubernetes Deployment or a Pod, and create a Kubernetes Service to expose it internally within the cluster. This Service will be the target for the Ingress.

3. **Install an Ingress Controller:** You need to have an Ingress controller installed in your cluster. Some popular options include Nginx Ingress Controller, Traefik, or HAProxy Ingress. The controller will manage the Ingress resources and configure the load balancer.

   For example, to install the Nginx Ingress Controller, you can use:

   ```bash
   kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.0.0/deploy/static/provider/cloud/deploy.yaml
   ```

4. **Create an Ingress Resource:** Define an Ingress resource that specifies the rules for routing traffic to your service. Here's an example Ingress resource manifest:

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

number: 80

```

In this example, traffic for `example.com/path` will be routed to `your-service`.

5. **Apply the Ingress Resource:** Use `kubectl apply` to create the Ingress resource in your cluster:

```bash
kubectl apply -f your-ingress.yaml
```

6. **Configure DNS:** Ensure that the DNS records for the specified hostname (e.g., `example.com`) point to the external IP address of your Ingress controller.

7. **Access Your Service:** After DNS propagation, you should be able to access your service externally via the hostname and path you defined in the Ingress resource.
```

4. Which service protocols does Kubernetes ingress expose?

Kubernetes Ingress is primarily designed to expose HTTP and HTTPS services, making it suitable for routing and load balancing web traffic. However, with the evolution of Kubernetes and Ingress controllers, it has expanded to support additional protocols and features:

1. **HTTP:** Ingress is commonly used to expose HTTP services. You can define routing rules based on URL paths, hostnames, and other HTTP attributes.

2. **HTTPS:** Secure HTTP services can be exposed through Ingress by configuring TLS certificates. This allows you to terminate SSL/TLS encryption at the Ingress controller and route decrypted traffic to your services.

3. **TCP:** Some Ingress controllers, like Nginx Ingress, support TCP services. This enables you to expose non-HTTP services such as databases or custom protocols. TCP-based routing typically relies on port numbers.

4. **UDP:** While less common, some Ingress controllers support UDP services. UDP is a connectionless protocol used for various purposes, including DNS and VoIP. Exposing UDP services may require specific controller support.

5. **gRPC:** If your services use the gRPC protocol, you can configure Ingress resources to handle gRPC traffic. gRPC is a high-performance RPC (Remote Procedure Call) framework often used for communication between microservices.

6. **WebSocket:** Ingress controllers can be configured to support WebSocket connections. WebSocket is a protocol that enables fullduplex communication over a single TCP connection and is used for real-time applications.

7. **Custom Protocols:** In some cases, you may need to expose services using custom or proprietary protocols. Depending on your Ingress controller and its capabilities, you might be able to configure it to handle these custom protocols.

Additionally, Ingress controllers often evolve, so it's essential to refer to the documentation and features of the specific controller you plan to use to ensure compatibility with your service protocols.

Animesh Parab T2-T21 88

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 acombination 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.