Assignment-1

Aim- To make a EC2 Machine in AWS

THEORY-

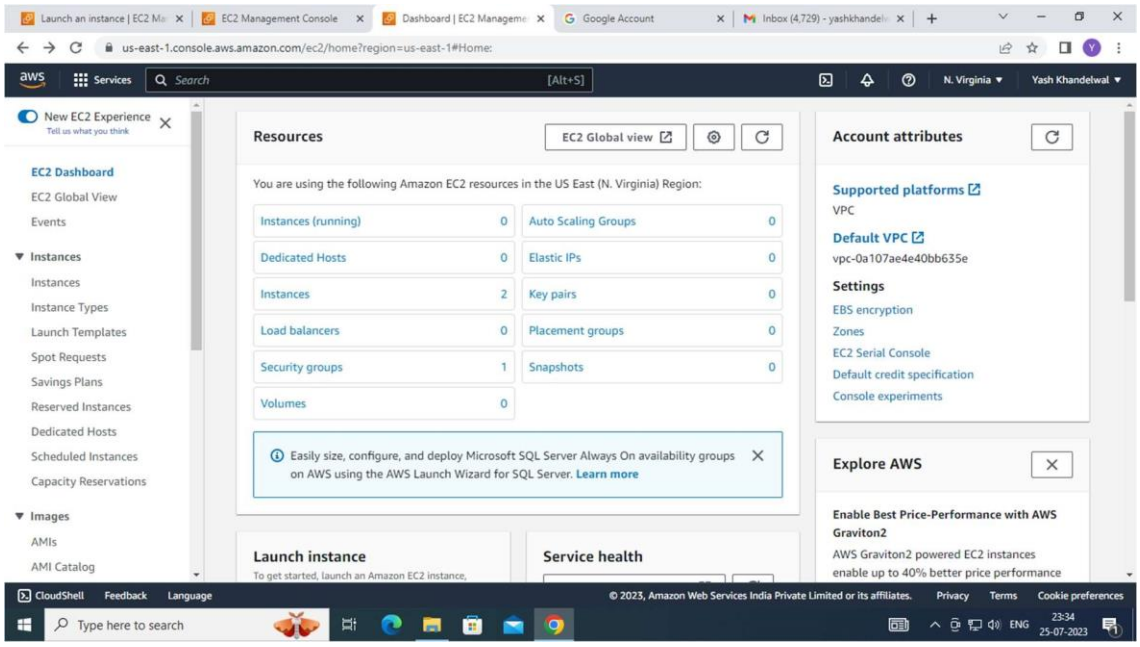
Amazon Elas c Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make webscale cloud compu ng easier for developers.

STEPS-

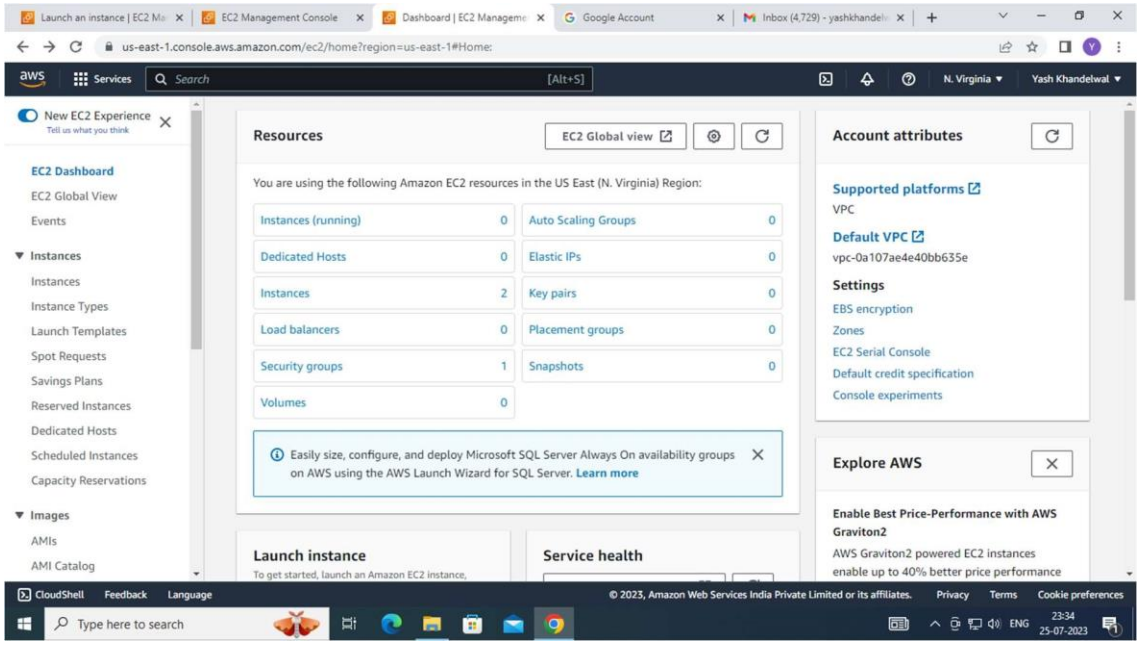
LOGIN TO AWS

ACCOUNT, THEN

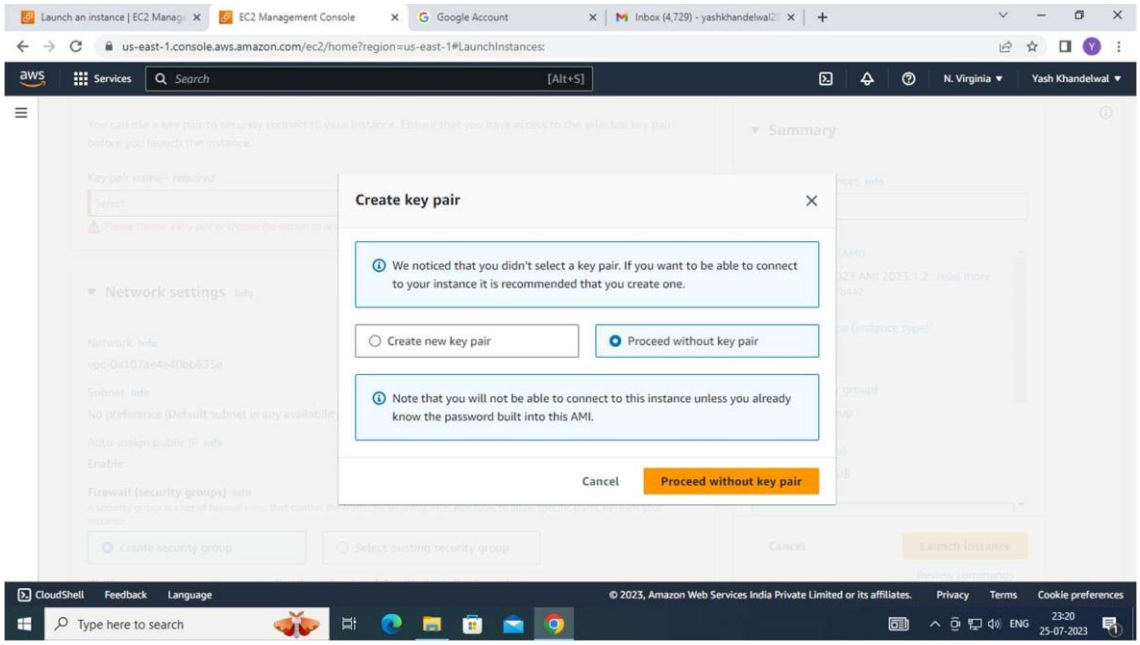
SEARCH EC2 .



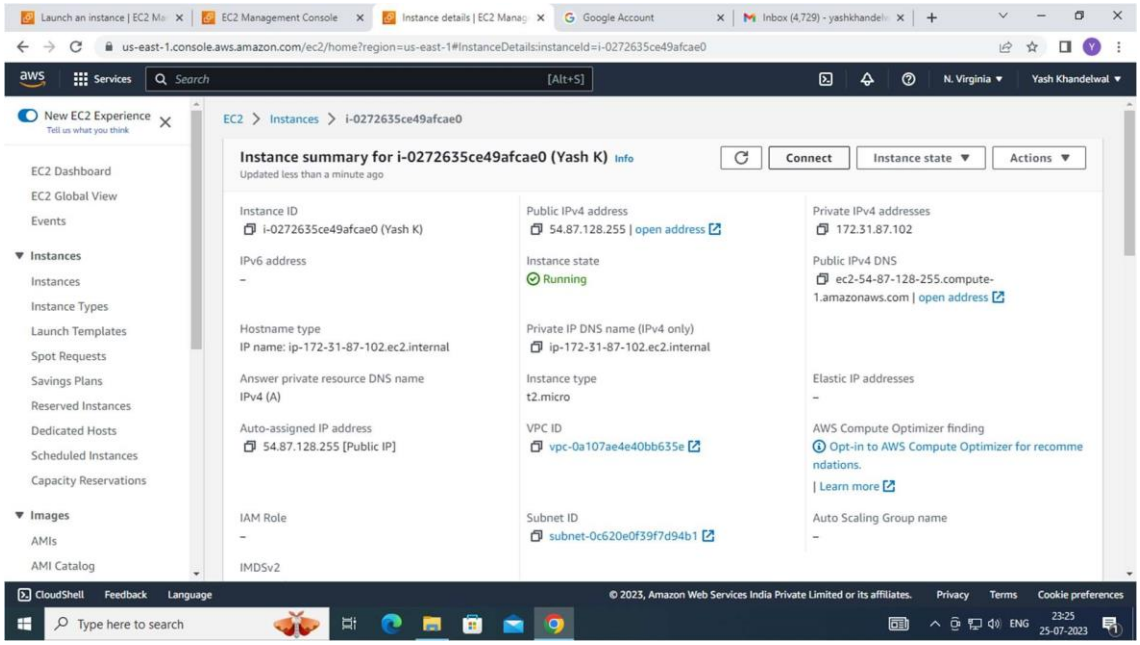
NOW CLICK ON LAUNCH / CREATE NEW INSTANCES.



Select whether you want to go with or without key pair.

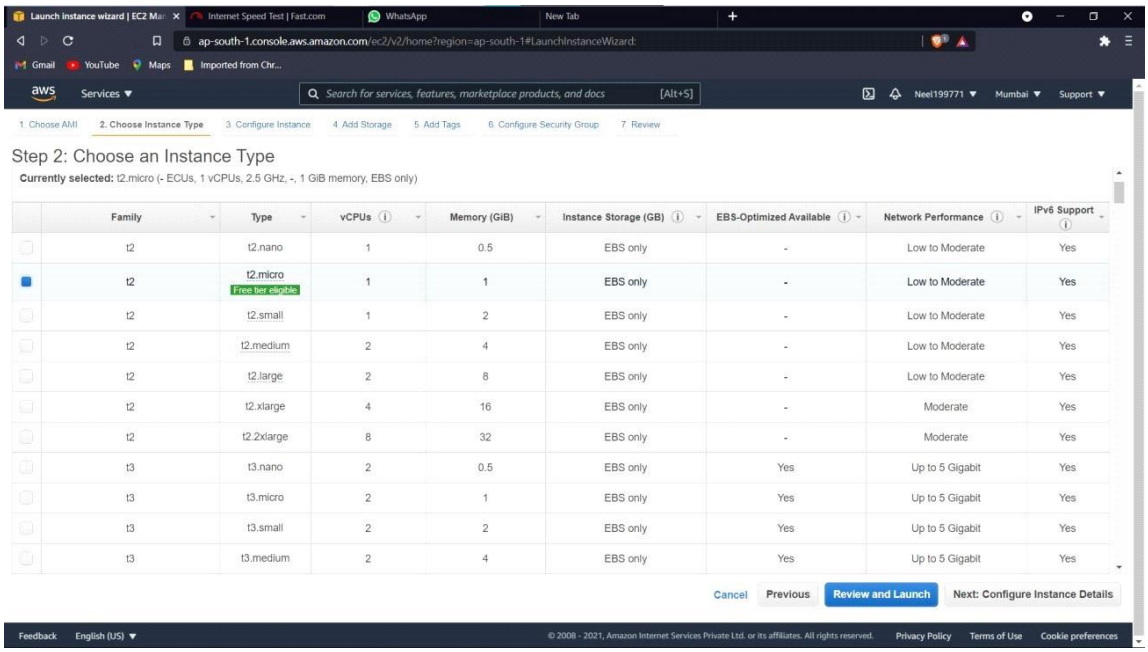


Wait for the success conforma on.

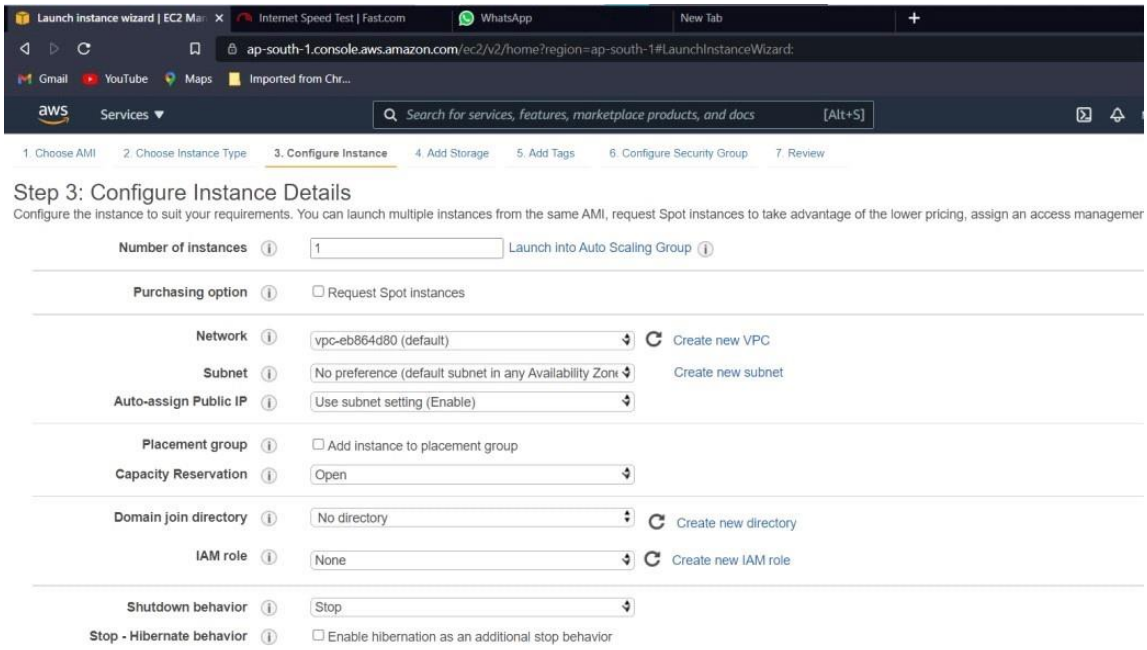


Choose any machine you want to create here I am crea ng UBUNTU(free er).

Click on T2 micro (free er one)



Click on NEXT, then Again Click Next.

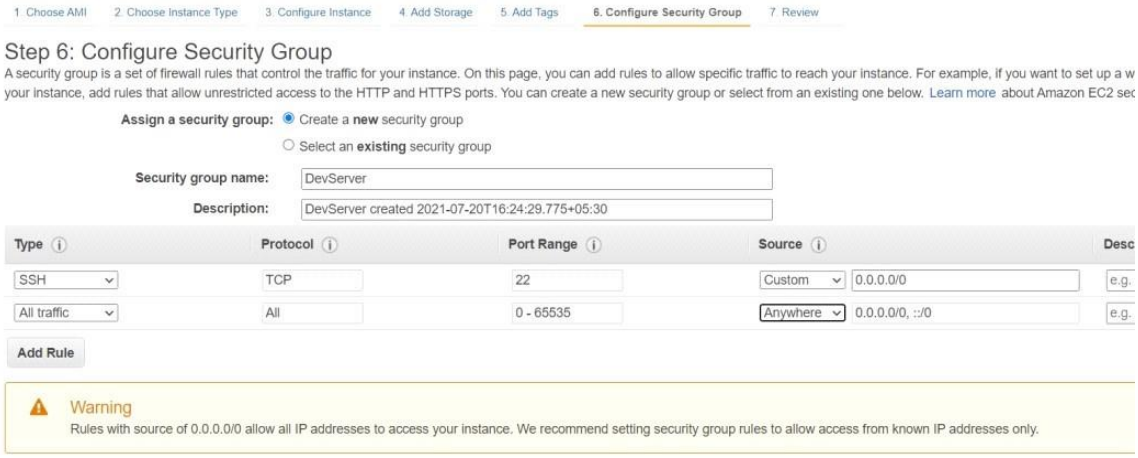


THEN CREATE A KEY PAIR BY ANY NAME AND DOWNLOAD IT. THEN CLICK NEXT

Now add security group ALL TRAFFIC ,

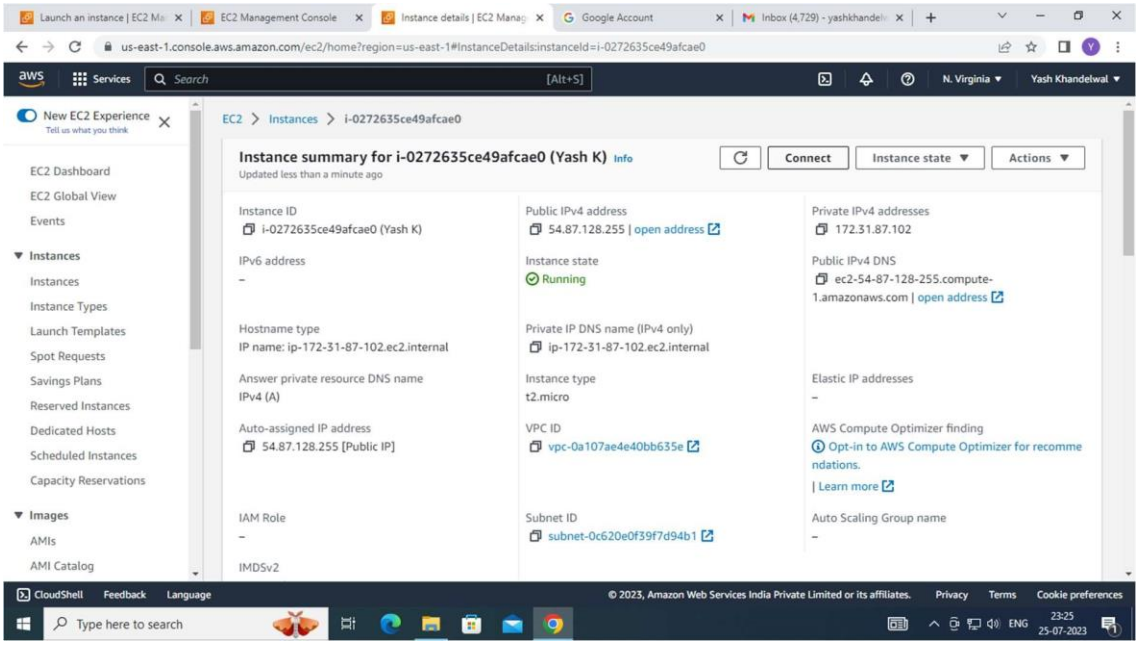
PROTOCOL – ALL,

SOURCE- ANYWHERE.

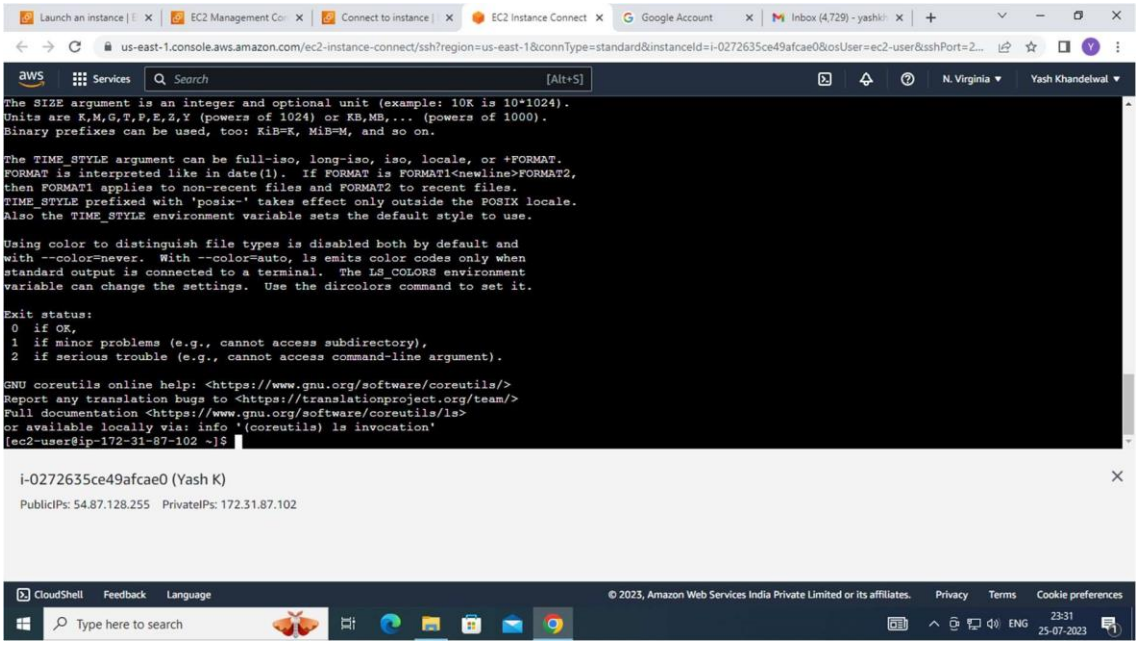


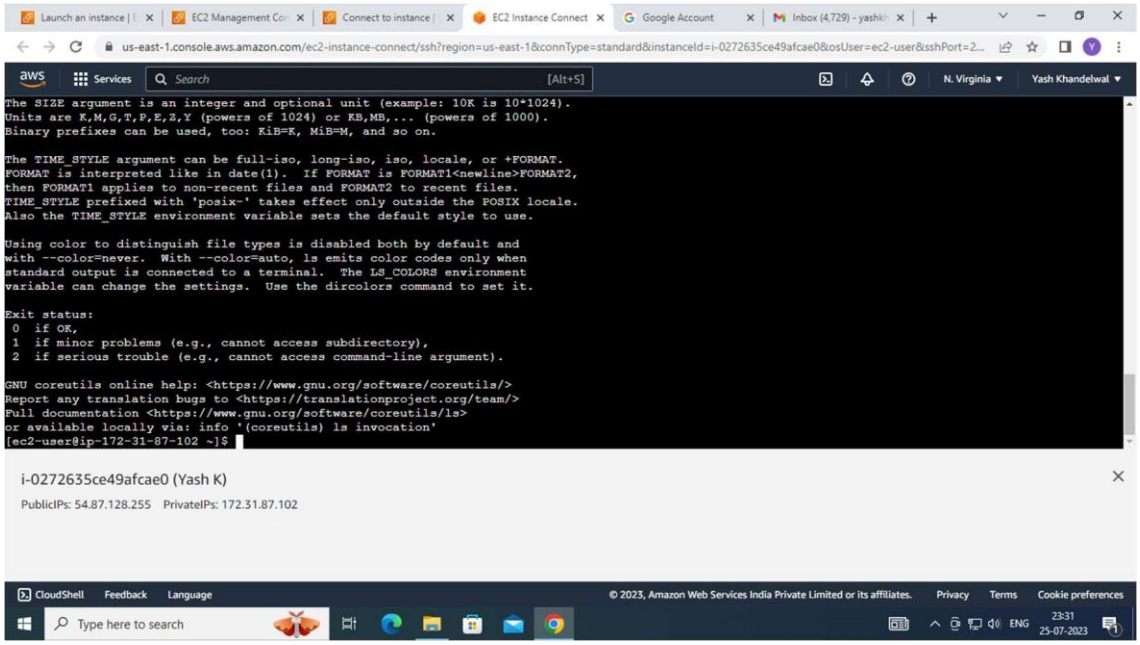
NOW WAIT TILL THE STATUS CHECK IS 2/2 and Instance is running.

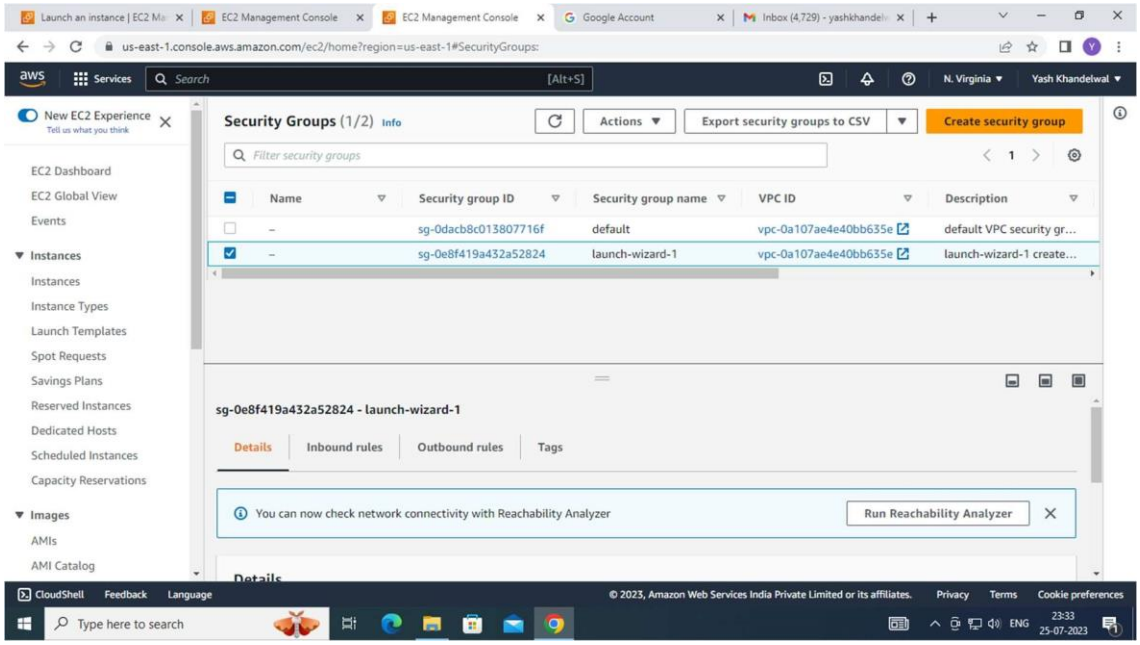
Once Check is complete click on launch instances.



FOLLOW SOME BASIC LINUX COMMANDS AS SHOWN BELOW-







At last terminate the instance.

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

ASSIGNMENT-2

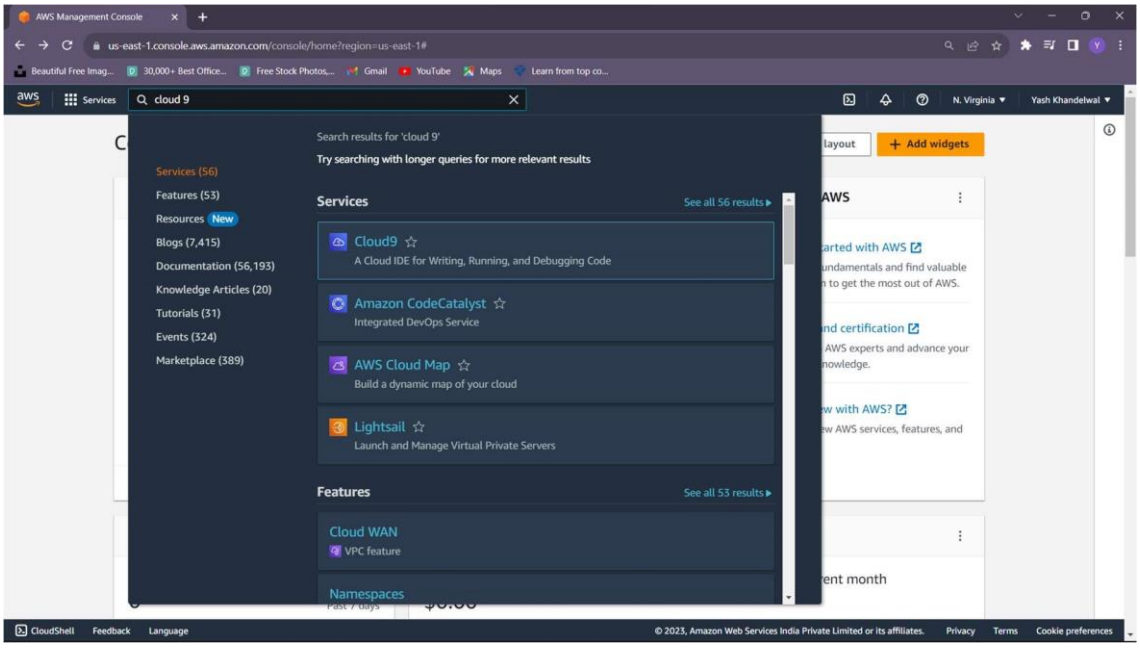
AIM- : To understand the benefits of Cloud Infrastructure and Setup AWS Cloud9 IDE, Launch AWS Cloud9 IDE and Perform Collabora on Demonstra on.

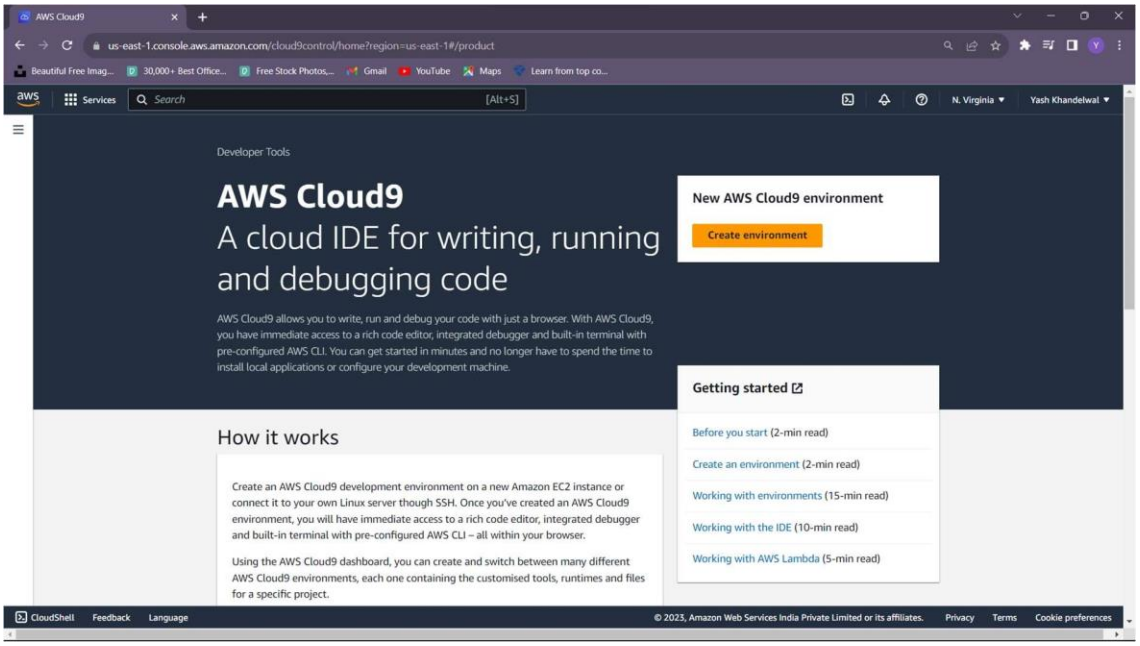
Theory-

Cloud9 IDE is an Online IDE, published as open source from version 2.0, un l version 3.0. It supports mul ple programming languages, including C, C++, PHP, Ruby, Perl, Python, JavaScript with Node.js, and Go. It is wri en almost en rely in JavaScript, and uses Node.js on the back-end.

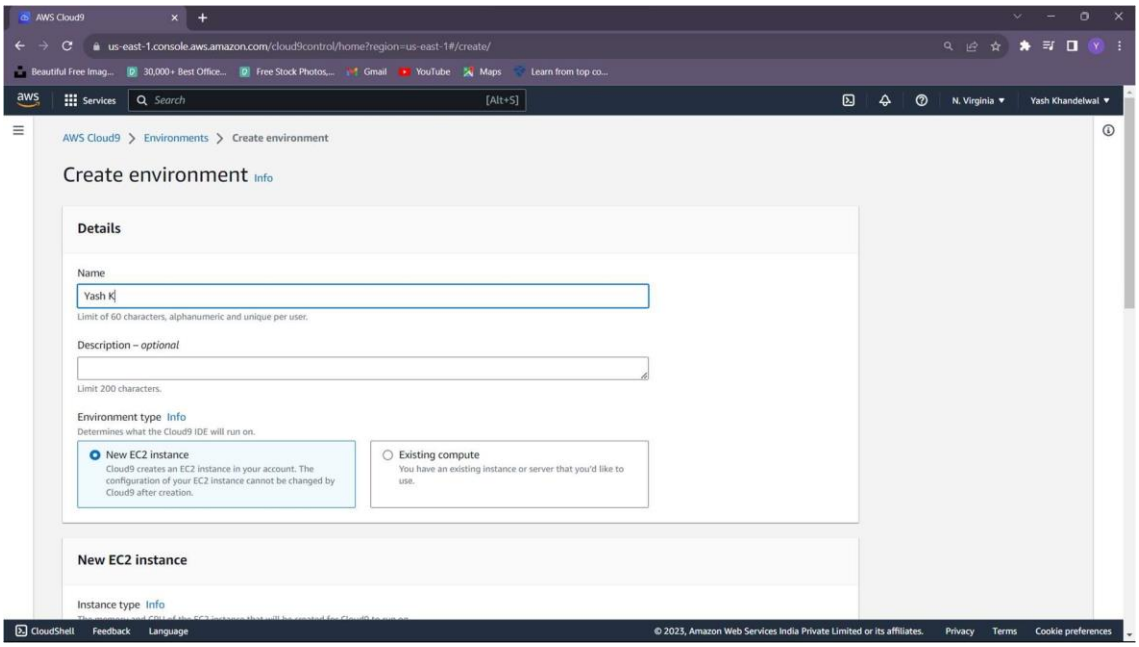
STEPS-

On the Create environment page enter a name for your environment. To add a descrip on to your environment, enter it in the Descrip on field. For Environment type, choose New EC2 instance to create an Amazon EC2 environment.

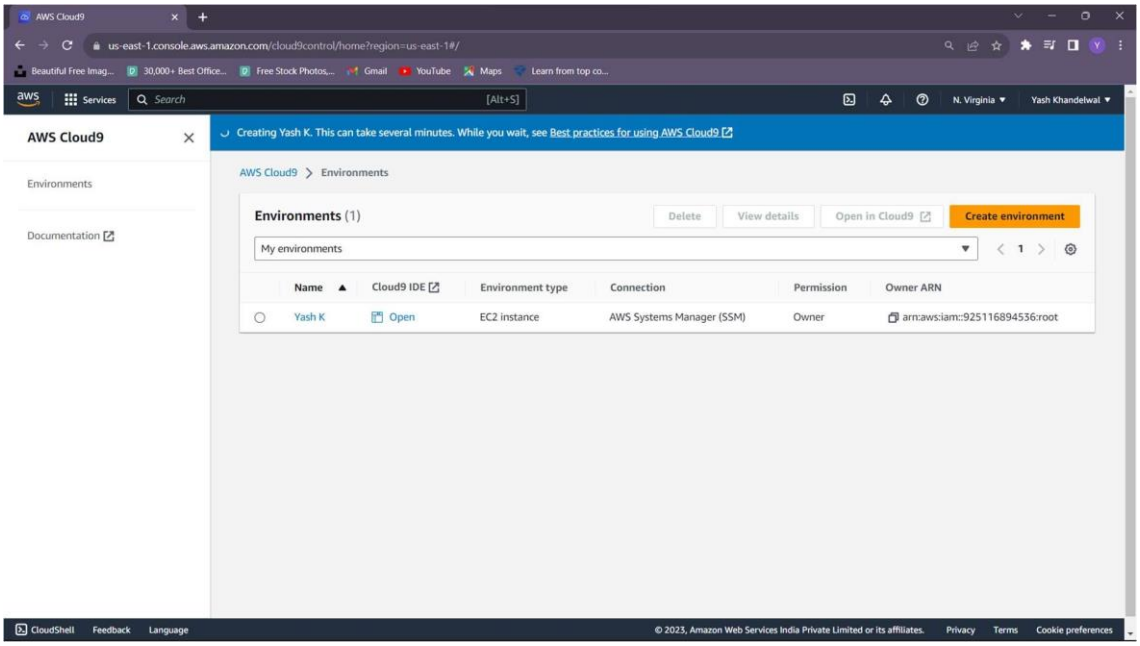




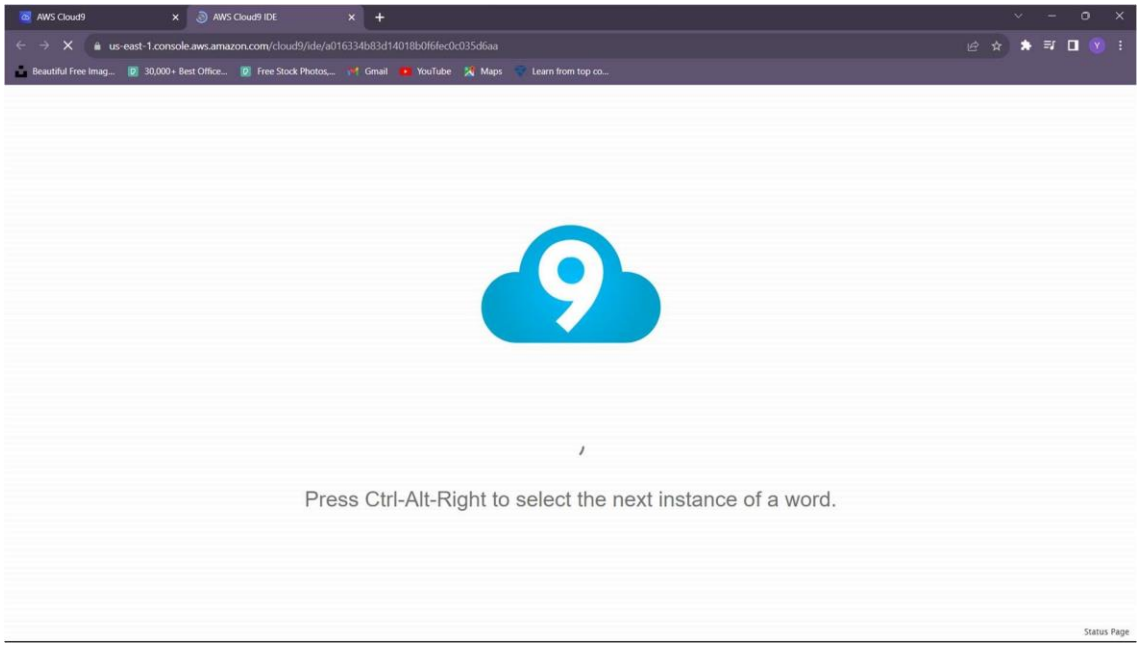
On the Create environment page enter a name for your environment. To add a descrip on to your environment, enter it in the Descrip on field. For Environment type, choose New EC2 instance to create an Amazon EC2 environment.



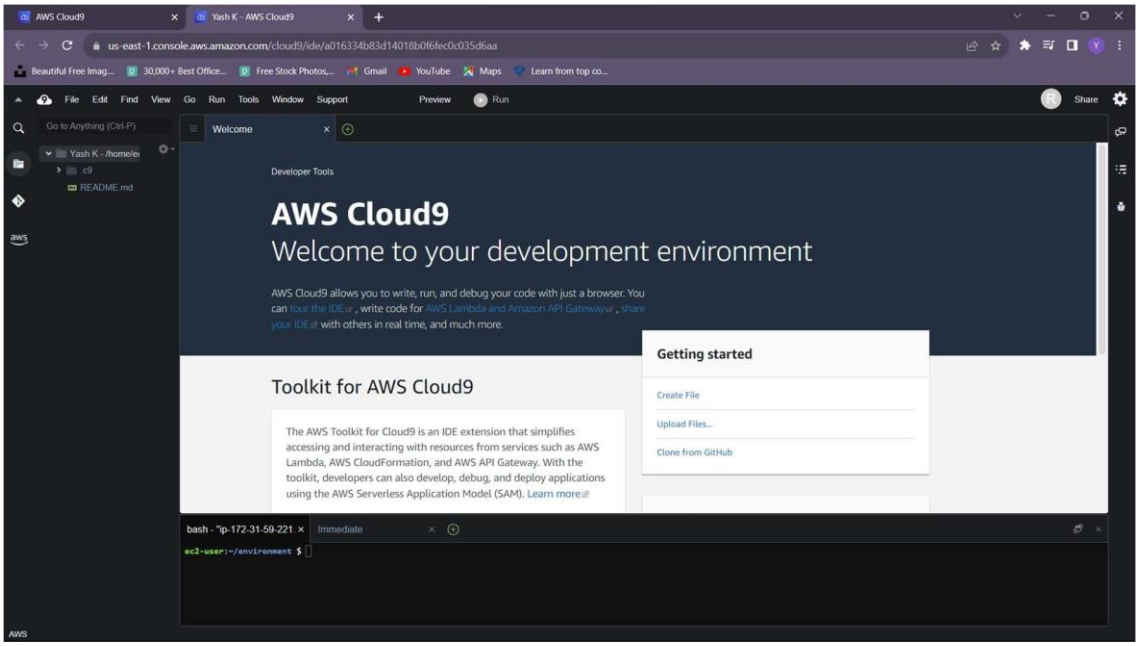
Now click on Create Environment.



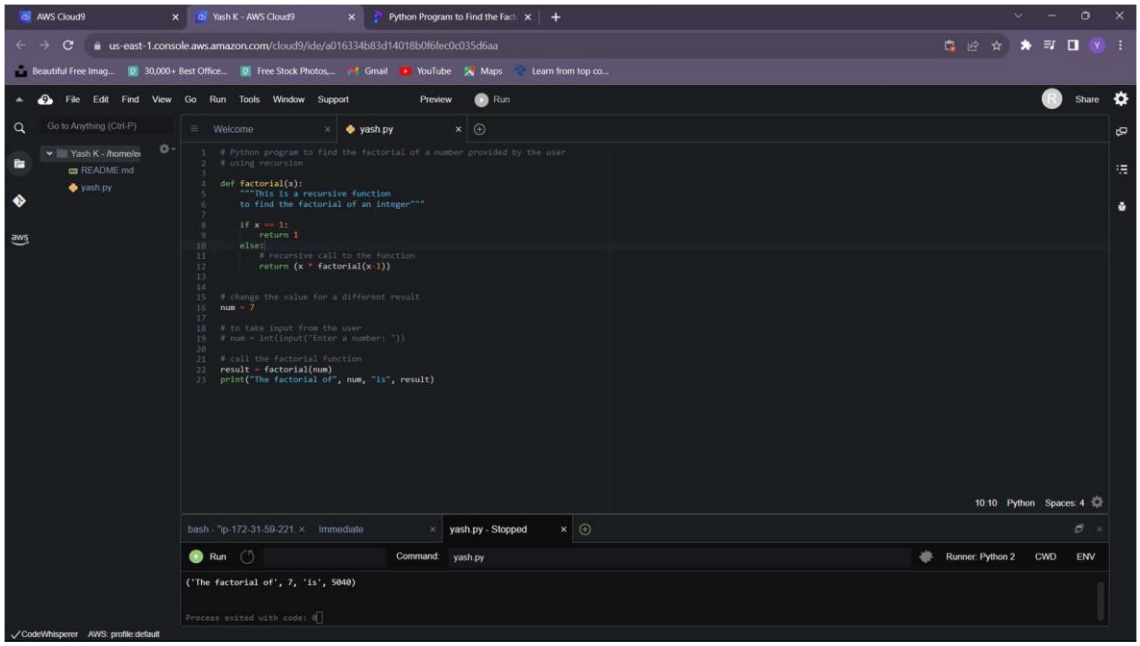
Check cloud9 instance that you have recently created



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



You can select any language that you want to execute



LO Mapped: LO1

Conclusion: Understood the fundamentals of Cloud Compu ng and are fully proficient with Cloud based DevOps solu on deployment op ons to meet our business requirements.

ASSIGNMENT No. – 3

Aim: To build your application using AWS CodeBuild and Deploy on S3/SEBS using AWS CodePipeline.

LO Mapped: LO1, LO2

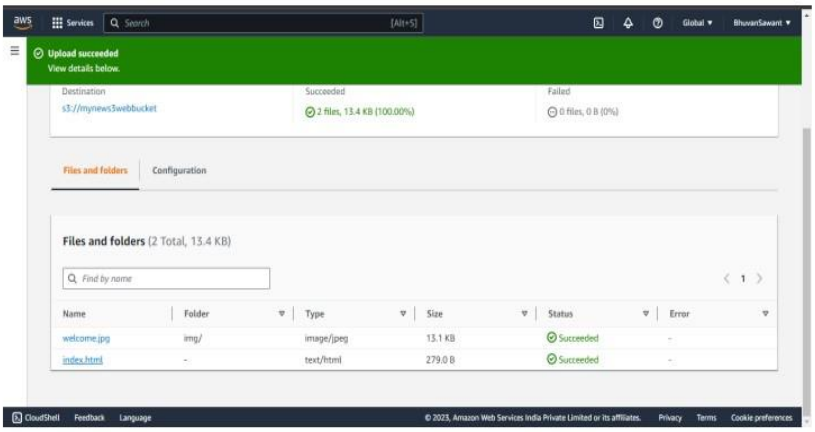
Theory:

Amazon Simple Storage Service (Amazon S3) is an obiect 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.

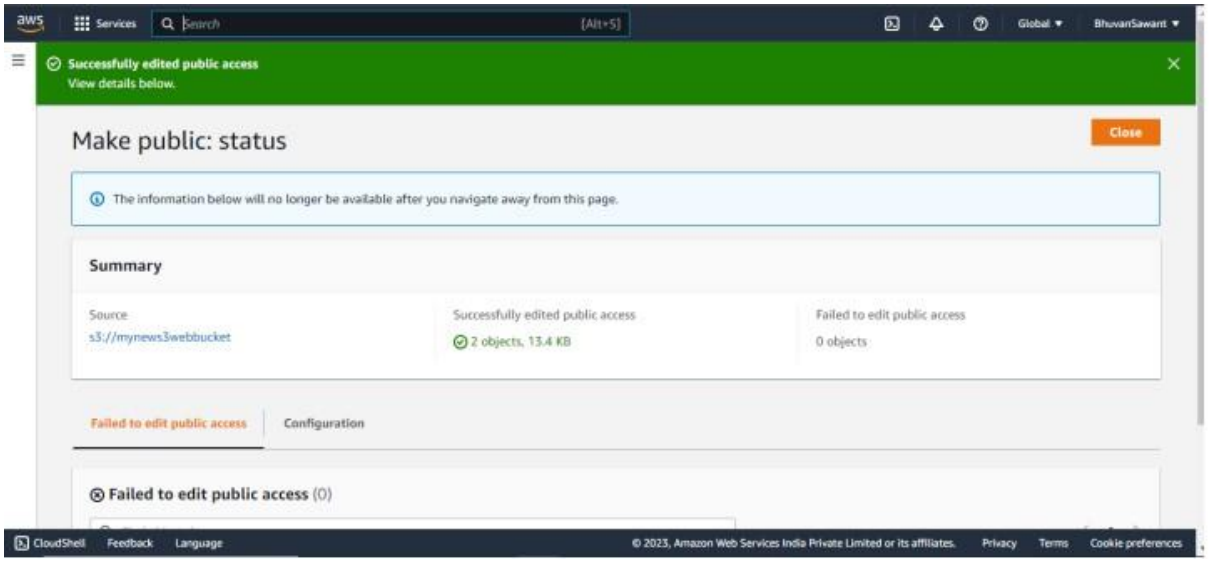
Some of the benefits of AWS S3 are:

- Durability: S3 provides 99.999999999 percent durability.
- Low cost: S3 lets you store data in a range of "storage classes." These classes are based on the frequency and immediacy you require in accessing files.
- Scalability: S3 charges you only for what resources you actually use, and there are no hidden fees or overage charges. You can scale your storage resources to easily meet your organization's ever-changing demands.
- Availability: S3 offers 99.99 percent availability of objects
- Security: S3 offers an impressive range of access management tools and encry'ption features that provide top-notch security.
- Flexibility: S3 is ideal for a wide range of uses like data storage, data backup, software delivery, data archiving, disaster recovery, website hosting, mobile applications, IoT devices, and much more.
- Simple data transfer: You don't have to be an IT genius to execute data transfers on S3. The service revolves around simplicity and ease of use.

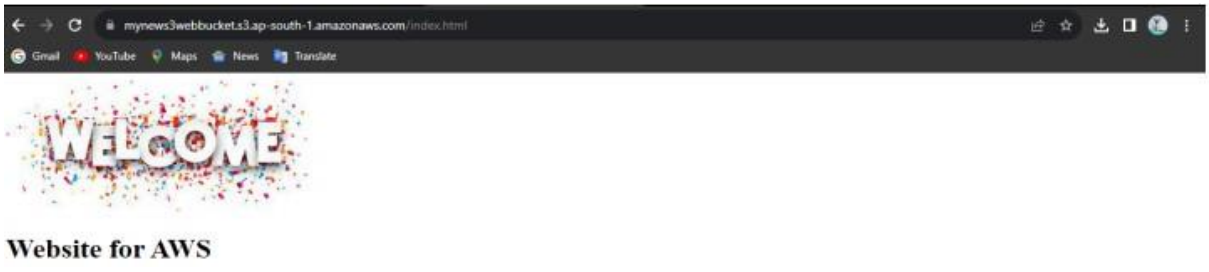
First login to your AWS account.



Then create a new S3 bucket.



After that host a static web page using the S3 bucket.



Conclusion: In this assignment we learnt about AWS S3 bucket and hosted a static web page using the S3 bucket.

Assignment Number: 4

Aim: To study AWS Code Pipeline and deploy web application using Code Pipeline.

LO mapped: LO1, LO2

Theory:

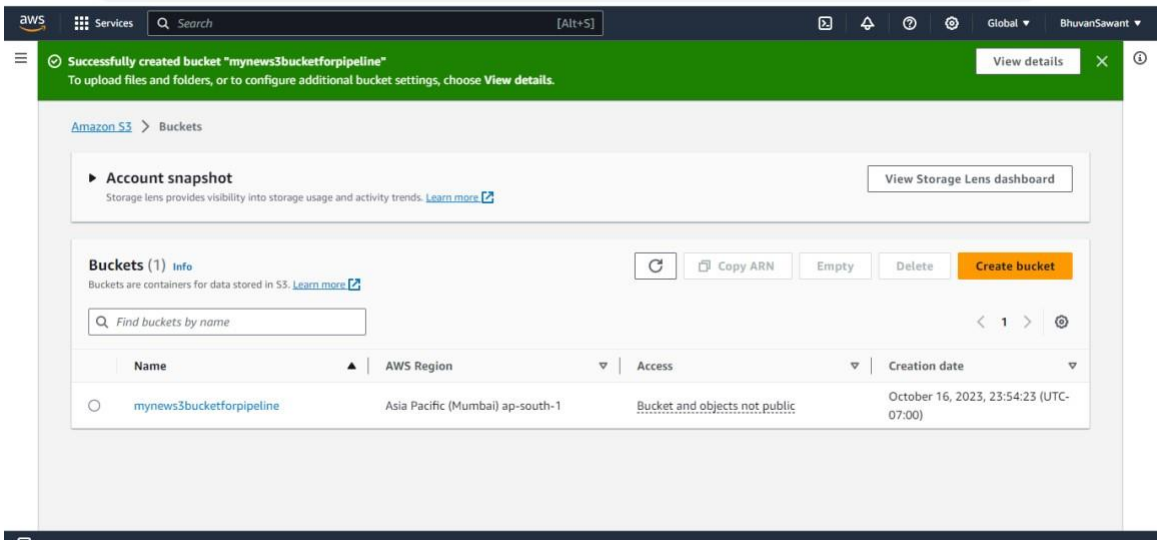
AWS CodePipeline is a continuous integration and continuous delivery (CI/CD) service provided by Amazon Web Services (AWS).

The key aspects of AWS CodePipeline:

Overview:

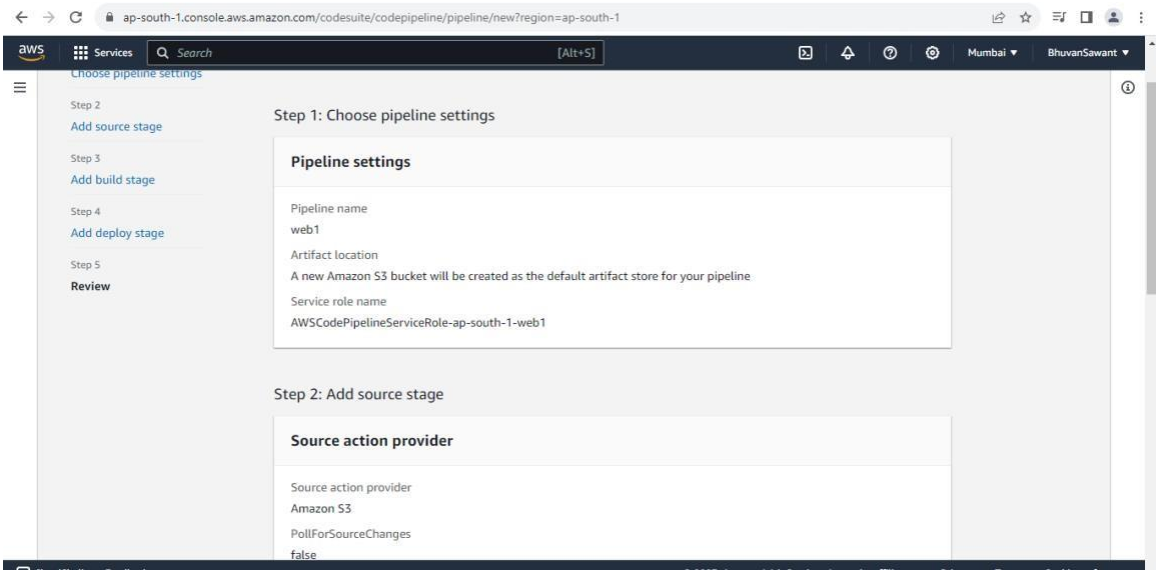
1. **CI/CD Workflow:**

    - AWS CodePipeline facilitates the automation of the build, test, and deployment phases of the release process. It allows you to define a series of stages, each of which can represent a phase in your release pipeline.



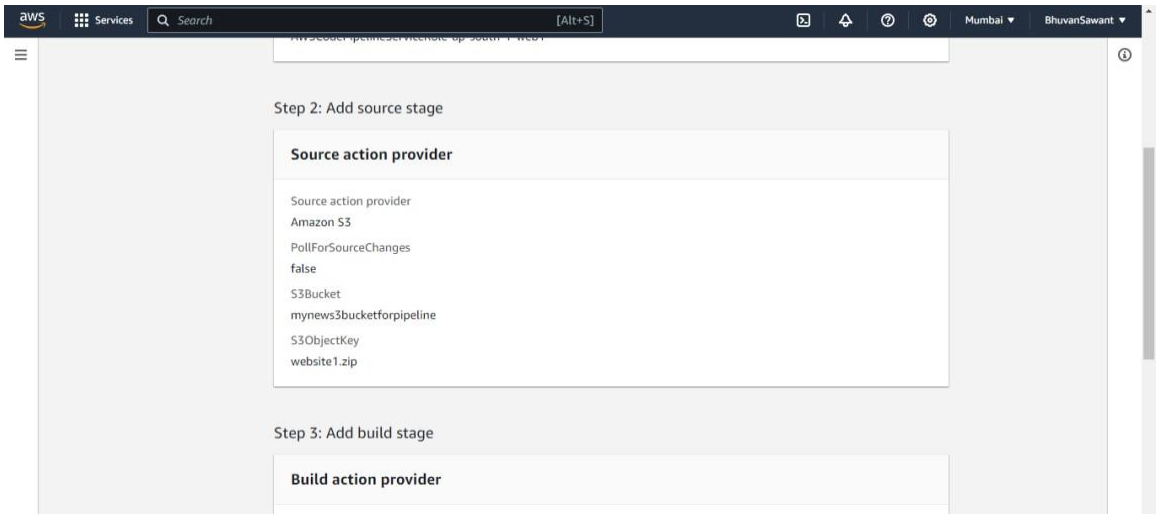
2. **Integration with Other AWS Services:**

    - CodePipeline integrates with various AWS services, such as AWS CodeBuild for building applications, AWS CodeDeploy for automating deployments, and AWS Lambda for running custom actions.

- 

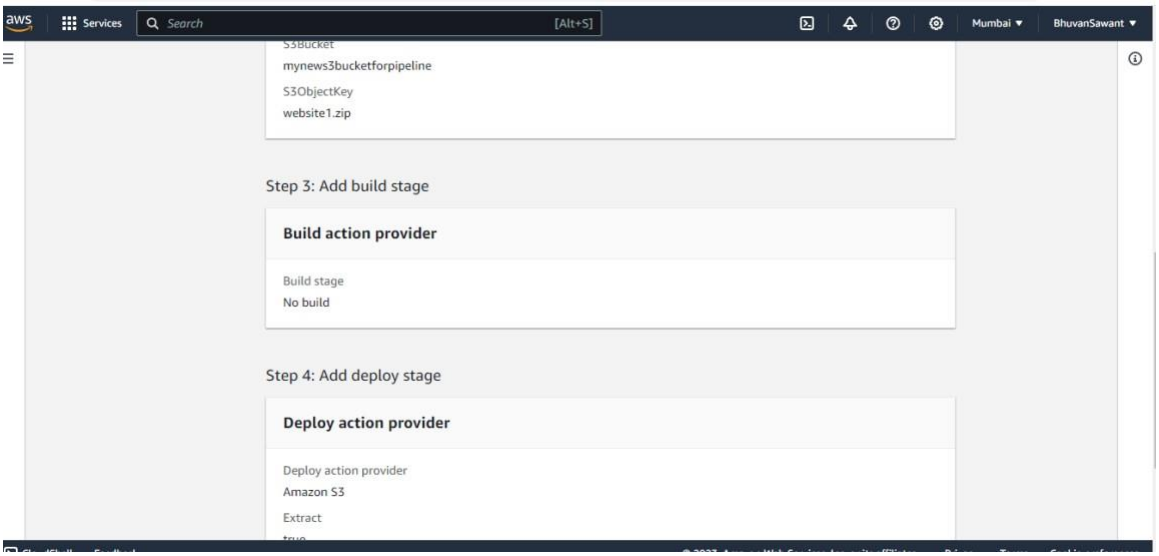
3. **Pipeline Execution:**

    - Pipelines consist of a series of stages, and each stage can have one or more actions. Actions represent a task, such as source code retrieval or deployment to a specific environment.



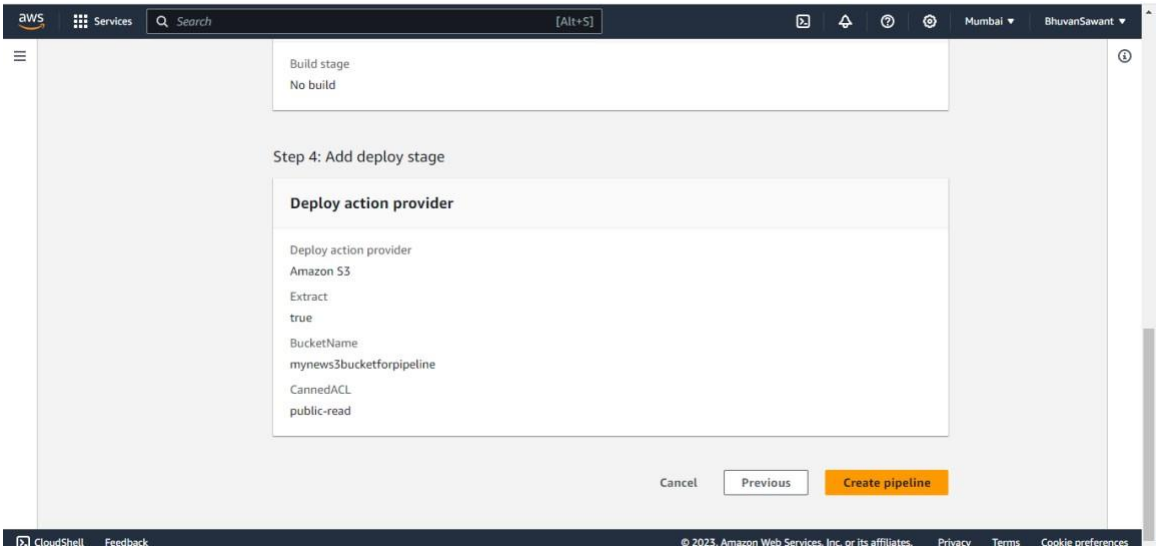
4. **Source Providers:**

    - CodePipeline supports integration with various source code repositories, including AWS CodeCommit, GitHub, and Amazon S3.

- 

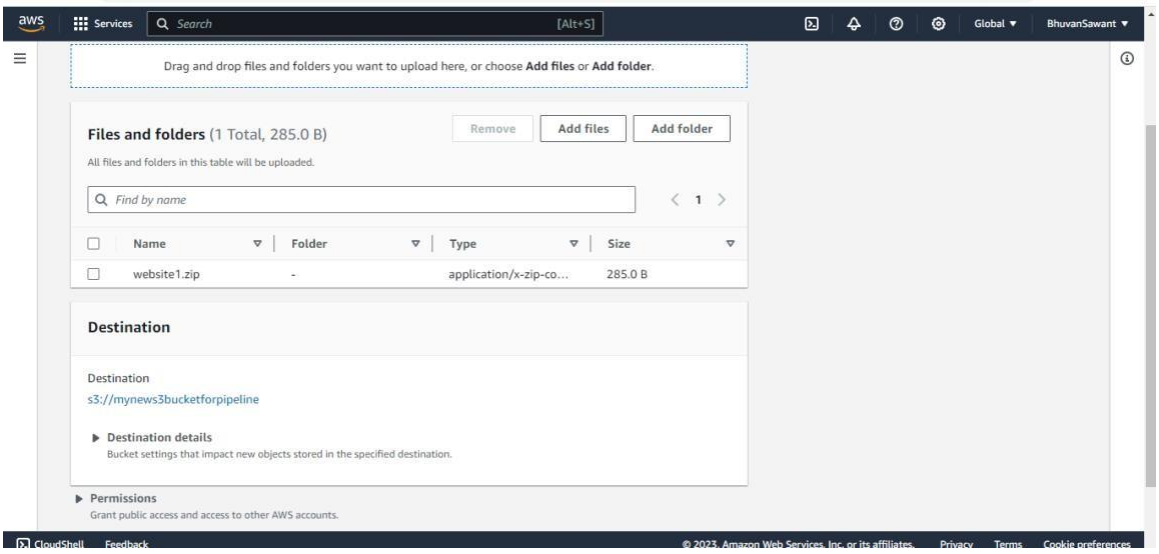
5. **Artifact Management:**

    - CodePipeline uses artifacts to store the files and data needed for each action in a pipeline. Artifacts can be passed between stages to ensure consistency in the deployment process.

    - 

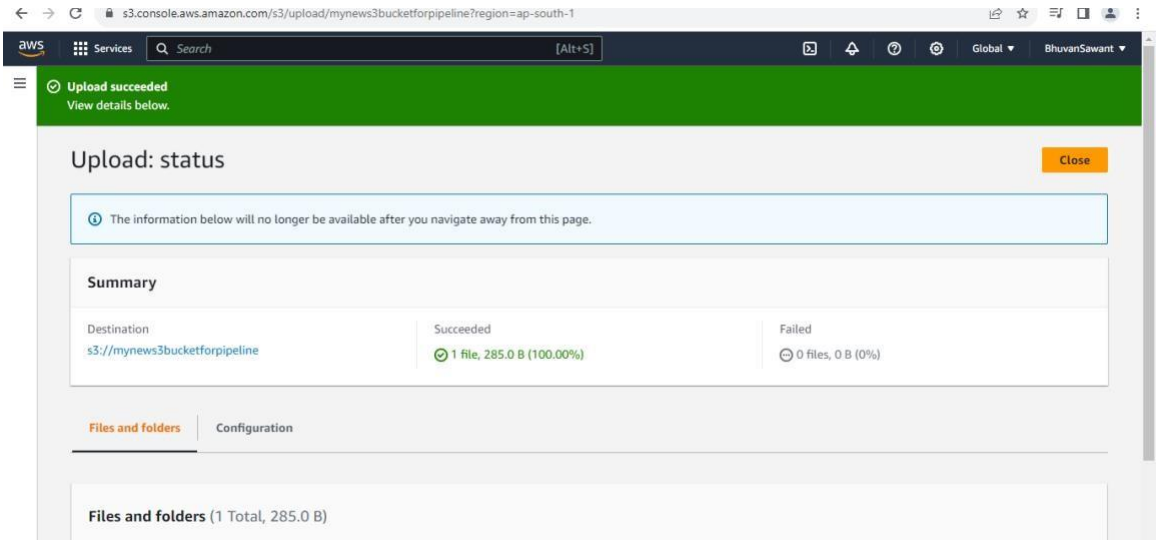
6. **Integration with Third-Party Tools:**

    - Besides AWS services, CodePipeline supports integration with third-party tools. This is achieved through custom actions, which allow you to use external tools and scripts in your pipeline.

- 

7. **Pipeline Visualizations:**

   - CodePipeline provides a visual representation of your release process, making it easy to understand and monitor the status of each stage and action.

- 

Key Concepts:

1. **Pipeline:**

   • A pipeline is a series of stages that represents your release process. Each stage can contain one or more actions.

2. **Stage:**

   • A stage is a logical unit in a pipeline, representing a phase in the release process. Stages are executed sequentially.

3. **Action:**

   • An action represents a task within a stage. Actions can include tasks such as building code, deploying to a test environment, or running tests.

4. **Artifact:**

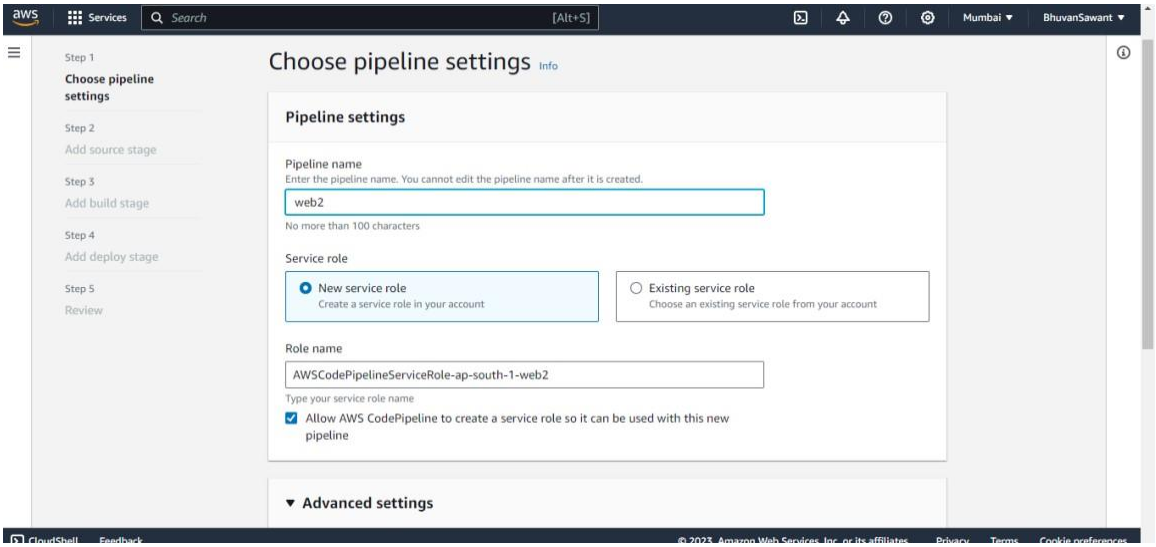
   • Artifacts are the files and data that are produced as a result of an action. They are used to pass information between stages in a pipeline.

To deploy web application using CodePipeline here are the following steps to be followed:

1. **Set Up Source Stage:**

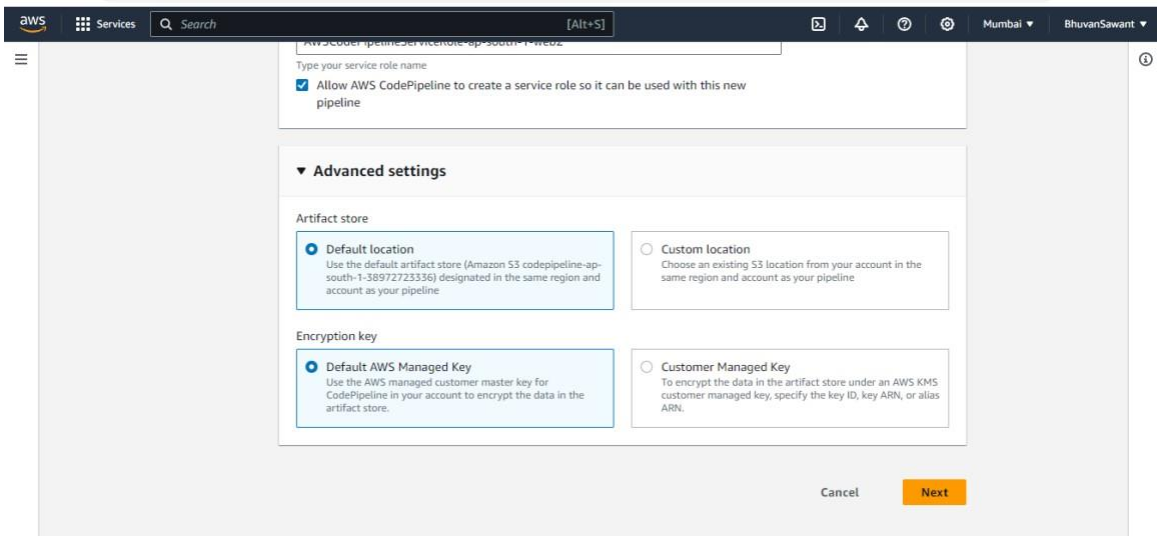
   • Configure a source stage in AWS CodePipeline, linking to your version control system (e.g., CodeCommit, GitHub).

   •



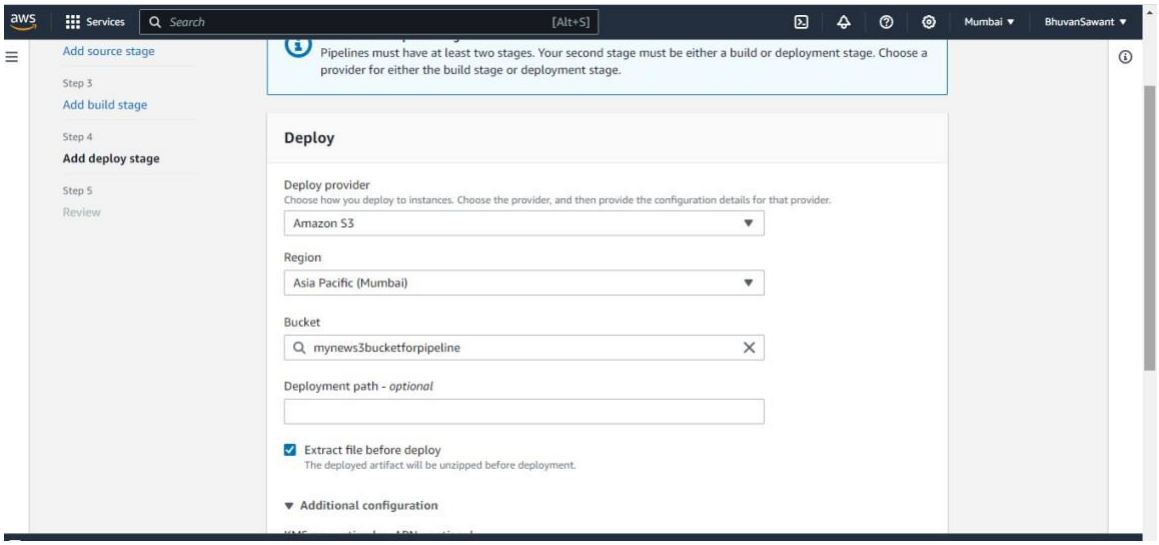
2. **Configure Build Stage:**

   • Set up a build stage using AWS CodeBuild to compile, test, and package your web application.



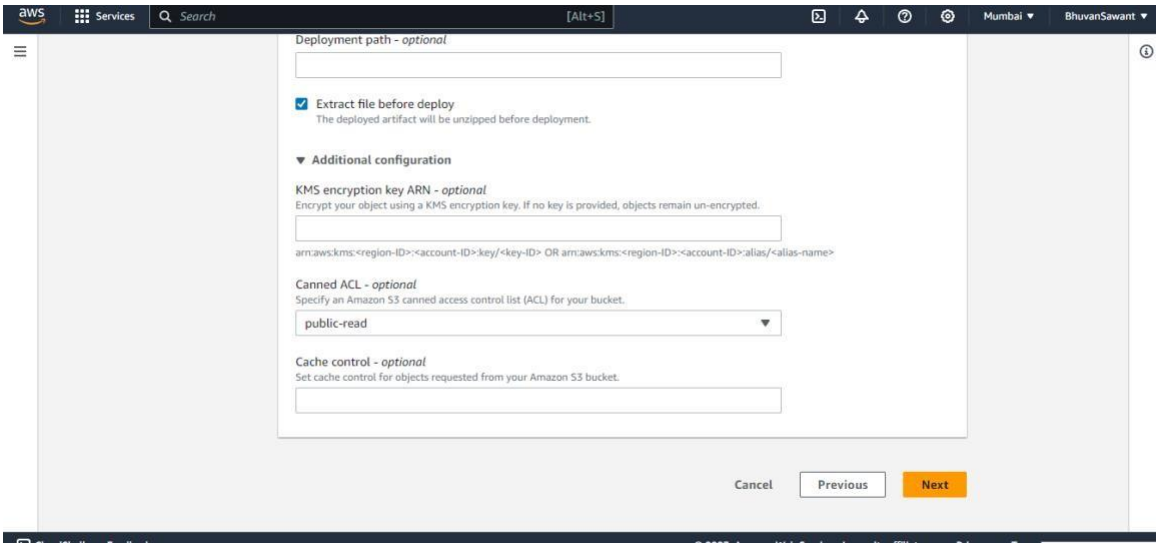
3. **Define Deployment Stage:**

   - Create a deployment stage using AWS CodeDeploy or another deployment provider to deploy your application to target environments.



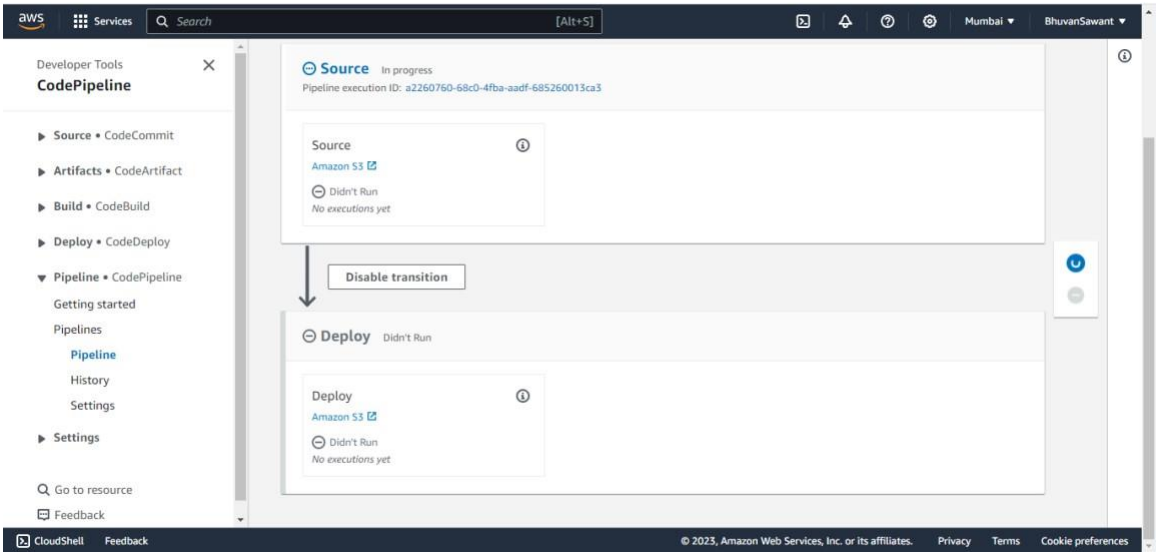
4. **Configure Approval (Optional):**

   - Optionally, add a manual approval stage to review and approve deployments before proceeding to the next stage.

- 

5. **Artifact Passing:**

   - Ensure proper passing of artifacts between stages to maintain consistency in the deployment process.

- 

6. **Add Monitoring (Optional):**

   • Integrate monitoring tools (e.g., AWS CloudWatch) to track the performance and health of your application during and after deployment.

7. **Configure Notifications (Optional):**

   • Set up notifications using AWS SNS or other services to receive alerts about pipeline events and status changes.

8. **Test and Validate:**

- Test the pipeline by triggering a build, ensuring that each stage executes successfully, and the application deploys as expected.

9. **Modify Pipeline as Needed:**

   - Make adjustments to the pipeline configuration based on the specific requirements of your web application and deployment process.

10. **Continuous Improvement:**

    - Implement continuous improvement practices, such as monitoring feedback, optimizing build and deployment scripts, and iterating on the pipeline structure.

Conclusion: By this assignment we learned how to host static web page through codepipeline.

Assignment No. 5

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

Kubernetes nodes run several services to ensure the proper operation of containerized applications. Here are the key services and their roles:

1. **Kubelet:**
    - Role: The Kubelet is an agent running on each node, responsible for managing and maintaining the containers running on the node. It communicates with the Kubernetes API server to ensure that the containers are in the desired state.
2. **Container Runtime:**
    - Role: This is the software responsible for running containers, such as Docker or containerd. It ensures that containers are isolated and that the processes inside them have limited access to resources.
3. **Kube Proxy:**
    - Role: Kube Proxy maintains network rules on nodes. It enables communication between different pods and services, implementing network policies defined in Kubernetes.
4. **CRI (Container Runtime Interface):**
    - Role: CRI is an interface between Kubernetes and the container runtime. It standardizes the way Kubernetes interacts with different container runtimes, ensuring compatibility.
5. **Node Problem Detector:**
    - Role: Node Problem Detector is responsible for detecting and reporting node-level issues, such as hardware or operating system problems. It helps with node recovery and maintenance.
6. **Container Storage Interface (CSI):**

Role: CSI allows storage systems to be seamlessly integrated with Kubernetes. It enables dynamic provisioning and management of persistent storage for pods.

7. **Sysctld:**
    - Role: Sysctld enforces system-level policies by managing kernel parameters and ensuring they align with Kubernetes security and performance requirements.
8. **Time Synchronization:**
    - Role: Accurate time synchronization is essential for various Kubernetes features, like certificate management and coordination among nodes. NTP (Network Time Protocol) or similar services ensure time synchronization.
9. **OS-Level Monitoring and Logging Agents:**
    - Role: These agents collect data on the node's performance, resource usage, and logs. Popular agents include Prometheus for monitoring and Fluentd or Logstash for log collection.
10. **OS Update Agents:**
    - Role: OS update agents ensure that the underlying operating system remains up to date with the latest security patches and updates.

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

A Pod Disruption Budget (PDB) is a resource in Kubernetes used to define policies that control the disruption of pods during maintenance operations, such as node draining or scaling down a deployment. PDBs help ensure high availability of applications by setting constraints on how many pods of a certain type can be disrupted at a given time.

The key elements of a PDB include:

- **minAvailable:** This field specifies the minimum number of pods that must be available at all times. It prevents excessive disruption by ensuring that a minimum number of healthy pods are maintained.

-

- **maxUnavailable:** This field allows a certain number or percentage of pods to be unavailable during disruptions. It provides flexibility while ensuring some level of redundancy.
- **Selector:** PDBs are associated with pods using labels and selectors, specifying which pods are subject to the budget.

By using PDBs, you can control the impact of maintenance operations on applications, avoiding scenarios where too many pods are disrupted simultaneously, which could lead to service downtime or instability. **Q.3 What is the role of Load Balance in Kubernetes?**

Load balancing in Kubernetes is a critical component for ensuring the availability, scalability, and reliability of applications. Here are the key roles of load balancing in Kubernetes:

1. **Service Discovery:**
    - Load balancers provide a stable endpoint (IP or DNS name) for services. Clients can connect to the service without needing to know the specific pod's IP address, allowing for dynamic scaling and failover.
2. **Traffic Distribution:**
    - Load balancers distribute incoming traffic to multiple pods or replicas of a service. This ensures even distribution of workloads, preventing overloading of specific pods and enhancing application performance.
3. **High Availability:**
    - Load balancers can detect unhealthy pods or nodes and redirect traffic away from them. This ensures that even if some pods fail, the service remains available and operational.
4. **Scalability:**
    - As traffic increases, load balancers can route requests to new pods or nodes. This elasticity allows applications to handle increased load without manual intervention.

5. **Session Persistence:**

   Some load balancers support session affinity, ensuring that requests from the same client are consistently directed to the same pod. This is important for stateful applications.

6. **Security:**

   - Load balancers can enforce security policies, such as rate limiting or access control, to protect services from malicious traffic.

7. **External Connectivity:**

   - In cloud environments, load balancers often provide external IP addresses, allowing applications to be exposed to the internet securely. This is particularly important for web services.

8. **Global Load Balancing:**

   - Some load balancers support global load balancing, distributing traffic across multiple clusters in different geographical regions for improved performance and resilience.

Load balancing in Kubernetes can be implemented using various technologies, including Ingress controllers, service types like NodePort or LoadBalancer, and cloud-managed load balancers from providers like AWS, Google Cloud, and Azure. It plays a crucial role in managing the complexity of containerized applications and ensuring their reliability and scalability.

Diptanshu Mishra Roll No:78

Assignment Number: 6

Aim: To build, change, and destroy AWS / GCP / Microsoft Azure / Digital Ocean infrastructure Using Terraform. **LO mapped:** LO1 and LO3

Theory:

To build change, and destroy AWS infrastructure Using Terraform we would use following tools:

1. IAM (AWS SERVICE)
2. EC2(only for AMI id)
3. VS code
4. Terraform installation in Local Machine via Environment variables

IAM Service:

AWS Identity and Access Management (IAM) acts like a guardian for your AWS resources. It lets you create personalized "passes" for users and roles, specifying what they're allowed to do. You can group users together for easier management. IAM also keeps a record of actions taken, ensuring security and accountability. Essentially, IAM safeguards your AWS "club" by controlling who can access what.

STEPS TO CREATE AN IAM USER: I.
In AWS search for IAM

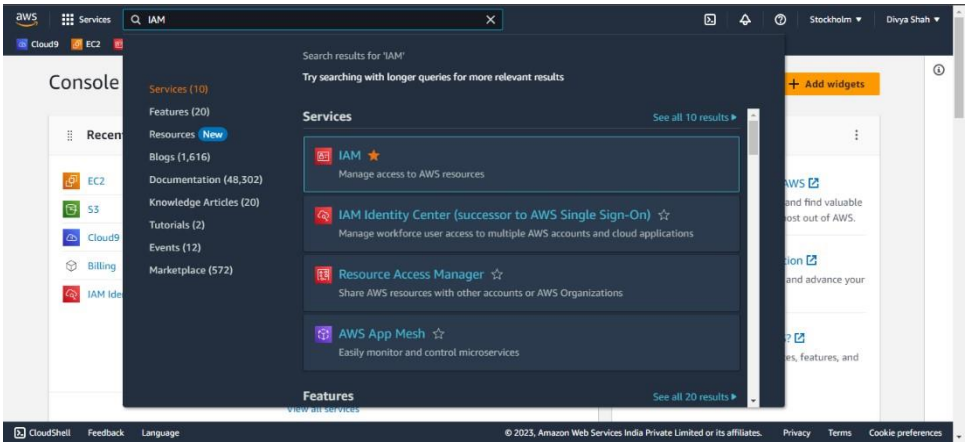


Fig 1

Diptanshu Mishra Roll No:78

II. The dashboard of IAM will look like this (Fig. 2)

Now click on “Users” present in the left panel to create an IAM User:

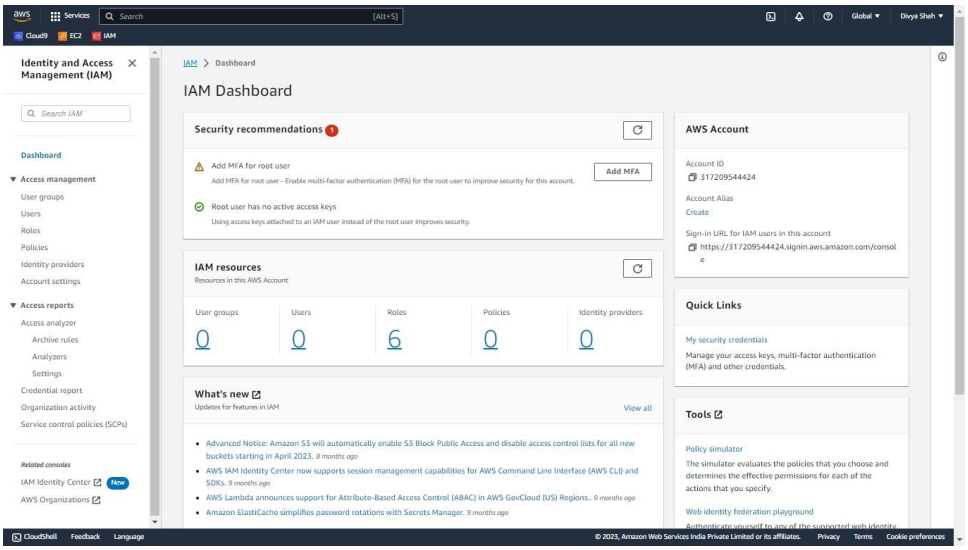


Fig. 2

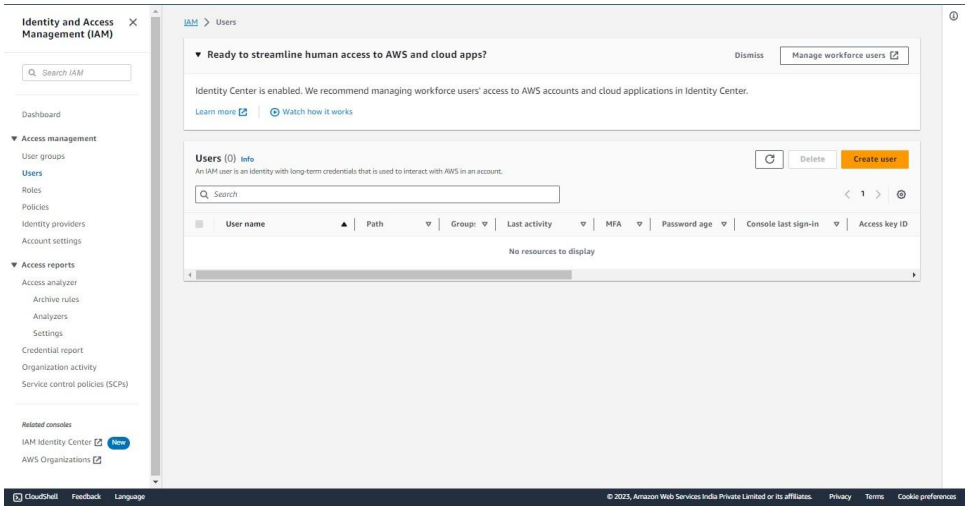


Fig.3

III. Provide a “Username” and click next:

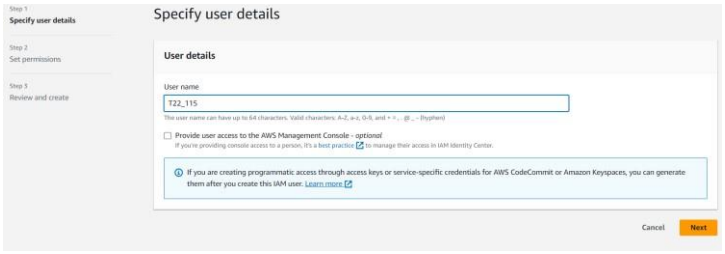


Fig. 4

Diptanshu Mishra Roll No:78

IV. Here for permissions click on “Attach policies directly” and select “Administrator Access”. Then click on next as follows:

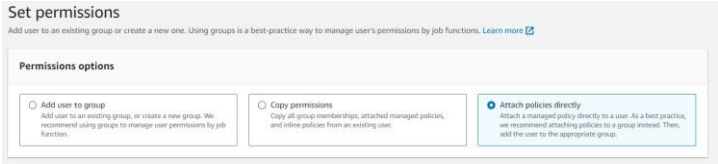


Fig. 5

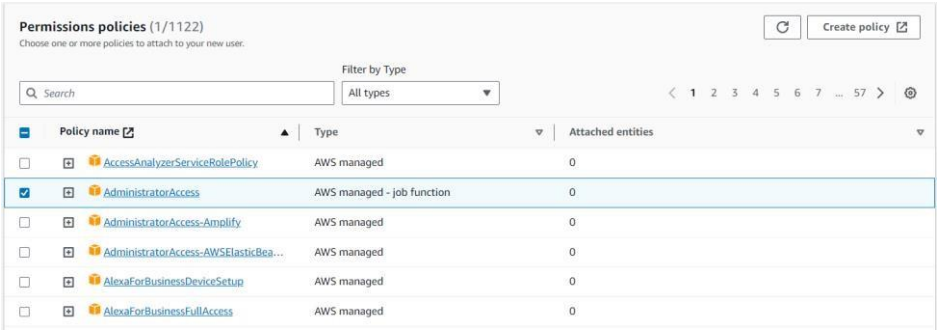


Fig. 6

V. Now click on “Create user”:

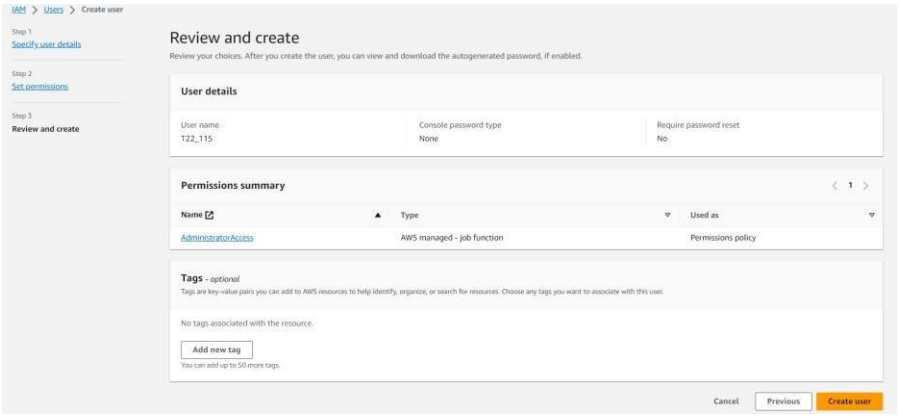


Fig. 7

VI. After successful IAM user creation go to user and click on “Username” and then create an access key for it as follows:

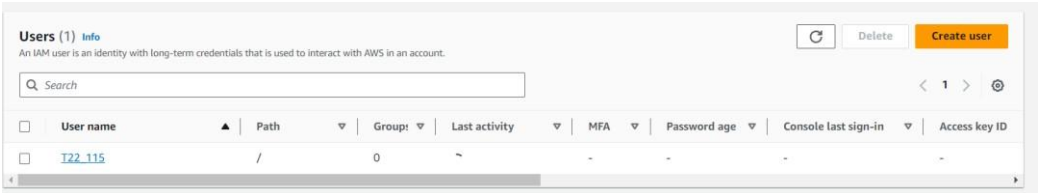


Fig.8

Diptanshu Mishra Roll No:78

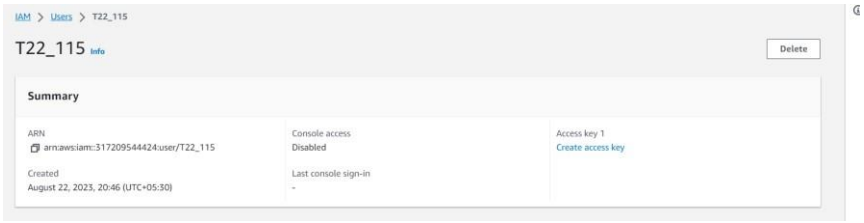


Fig.9

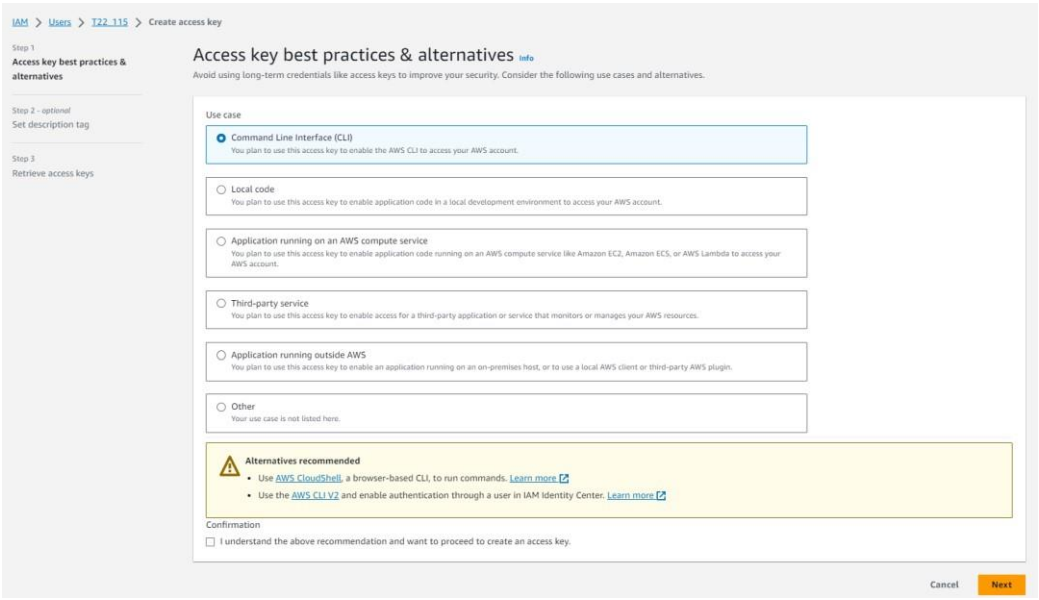
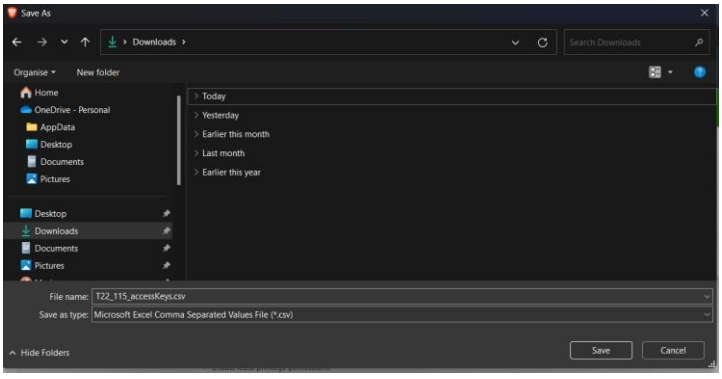


Fig.10

The below description is recommended to be kept empty. After this click on “Generate a key”:

Fig.11

VII. Now we need to Download the .csv file for copying the Access key as well as Secret access key:



Diptanshu Mishra Roll No:78

Fig. 12

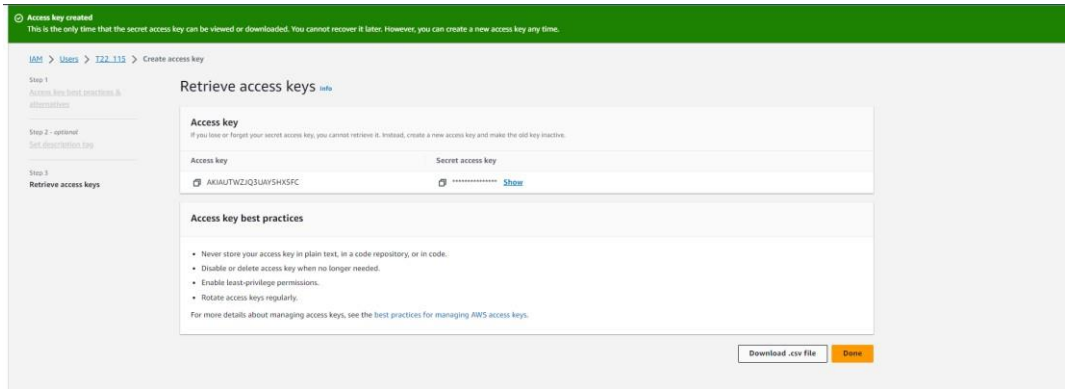


Fig.13

VIII. Now we need to open vs code an create a folder for .tf file operations. You can follow procedure as follows for creating .tf file with following code:

Fig.14

IX. Now we need to install terraform from chrome to set it in our environment variables:

Note: You need to download "386" for local windows machine.

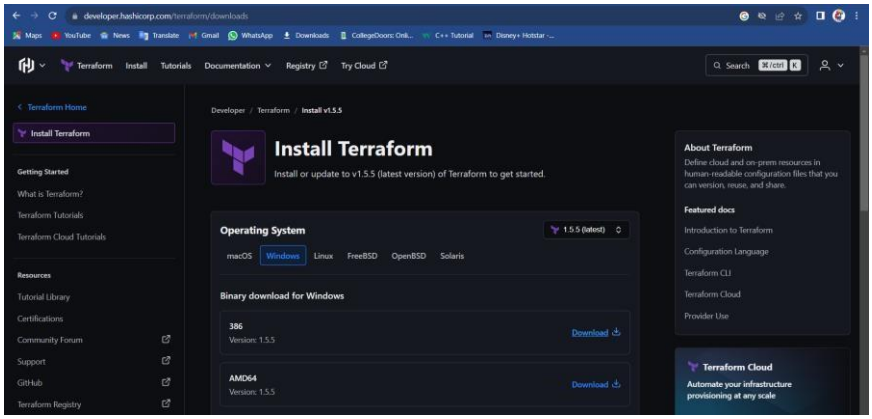


Fig.15

Copy path of downloaded Terraform and set it in environment variables as follow:

Diptanshu Mishra Roll No:78

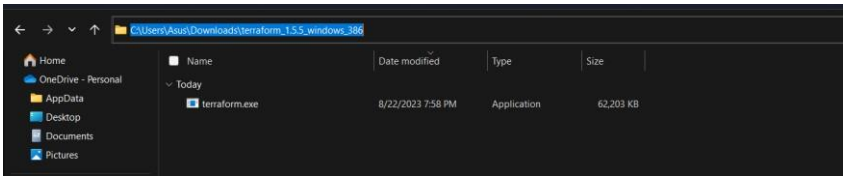


Fig.16

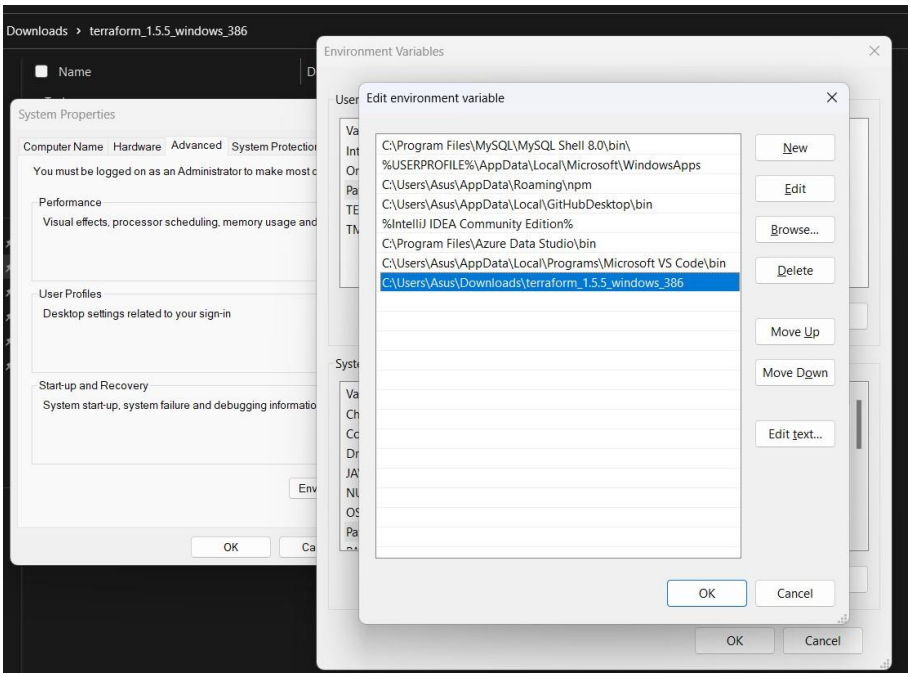


Fig.17

X. Now open cmd to location where you have saved you .tf file:

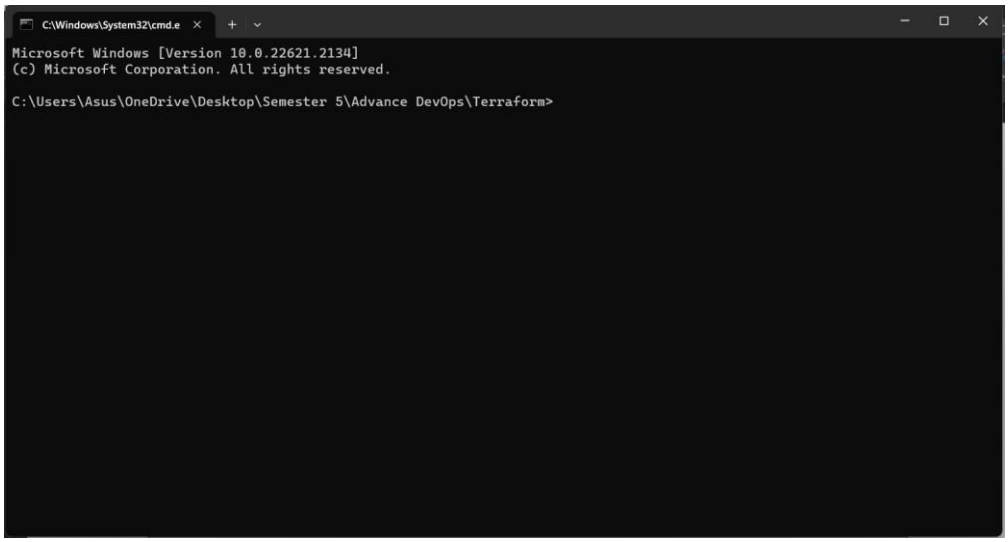


Fig. 18

XI. Now run command:

$$command: terraform\ init$$

Diptanshu Mishra Roll No:78

```
Microsoft Windows [Version 10.0.22621.2134]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Asus\OneDrive\Desktop\Semester 5\Advance DevOps\Terraform>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:\Users\Asus\OneDrive\Desktop\Semester 5\Advance DevOps\Terraform>
```

Fig. 19

XII. Now run command:

$$command: terraform\ plan$$

```
      + outpost_arn                          = (known after apply)
      + password_data                        = (known after apply)
      + placement_group                      = (known after apply)
      + placement_partition_number           = (known after apply)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + 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.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.

C:\Users\Asus\OneDrive\Desktop\Semester 5\Advance DevOps\Terraform>
```

Fig. 20

The new files which are created after above command.

Fig. 21

XIII. Now we need to run the command:

$$command: terraform\ apply$$

```
      + 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)
      + primary_network_interface_id         = (known after apply)
      + private_dns                          = (known after apply)
      + private_ip                           = (known after apply)
      + public_dns                           = (known after apply)
      + public_ip                            = (known after apply)
      + secondary_private_ips                = (known after apply)
      + security_groups                      = (known after apply)
      + source_dest_check                    = true
      + 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:
```

Fig. 22

Diptanshu Mishra Roll No:78

```
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.myT22_115: Creating...
aws_instance.myT22_115: Still creating... [10s elapsed]
aws_instance.myT22_115: Still creating... [20s elapsed]
aws_instance.myT22_115: Creation complete after 25s [id=i-0882605e5f3bc53fb]

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

C:\Users\Asus\OneDrive\Desktop\Semester 5\Advance DevOps\Terraform>
```

Fig. 23

XIV. Now after the above command you can see an instance is created in AWS:

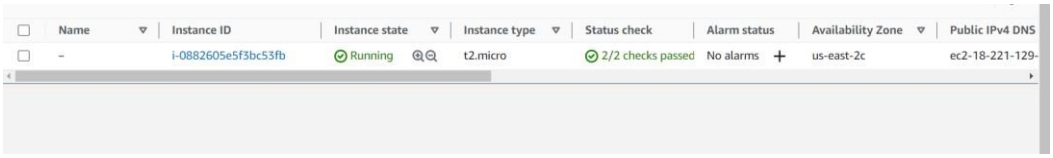


Fig. 24

XV. Now to destroy the created ec2 instance we need to run command:

$$command: terraform\ destroy$$

```
      - root_block_device {
          - delete_on_termination = true -> null
          - device_name           = "/dev/xvda" -> null
          - encrypted             = false -> null
          - iops                  = 100 -> null
          - tags                  = {} -> null
          - throughput            = 0 -> null
          - volume_id             = "vol-065015f5e4beb670c" -> null
          - volume_size           = 8 -> null
          - volume_type           = "gp2" -> null
        }
    }

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

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

  Enter a value: yes

aws_instance.myT22_115: Destroying... [id=i-0882605e5f3bc53fb]
aws_instance.myT22_115: Still destroying... [id=i-0882605e5f3bc53fb, 10s elapsed]
aws_instance.myT22_115: Still destroying... [id=i-0882605e5f3bc53fb, 20s elapsed]
aws_instance.myT22_115: Still destroying... [id=i-0882605e5f3bc53fb, 30s elapsed]
aws_instance.myT22_115: Destruction complete after 32s

Destroy complete! Resources: 1 destroyed.
```

Fig. 25

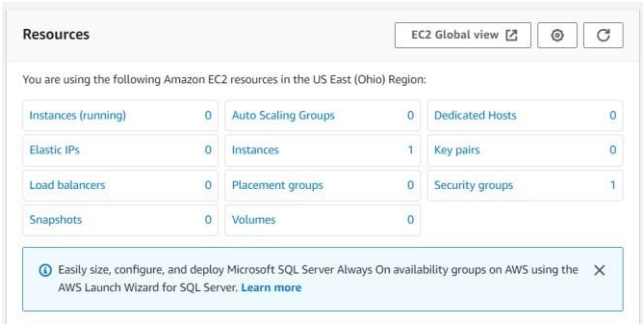


Fig. 26

Note: Do Delete your IAM User before logging off from your AWS account and any other sort of instance or program running.

Diptanshu Mishra Roll No:78

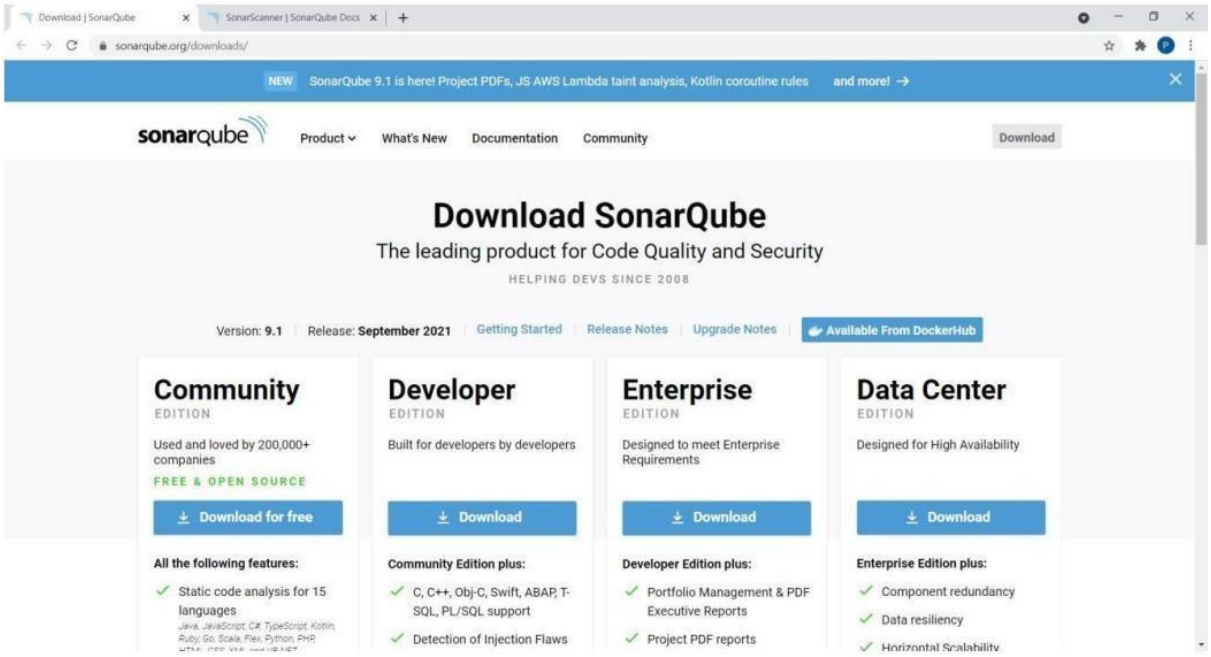
Conclusion: By this assignment we created, changed and destroyed AWS infrastructure using terraform.

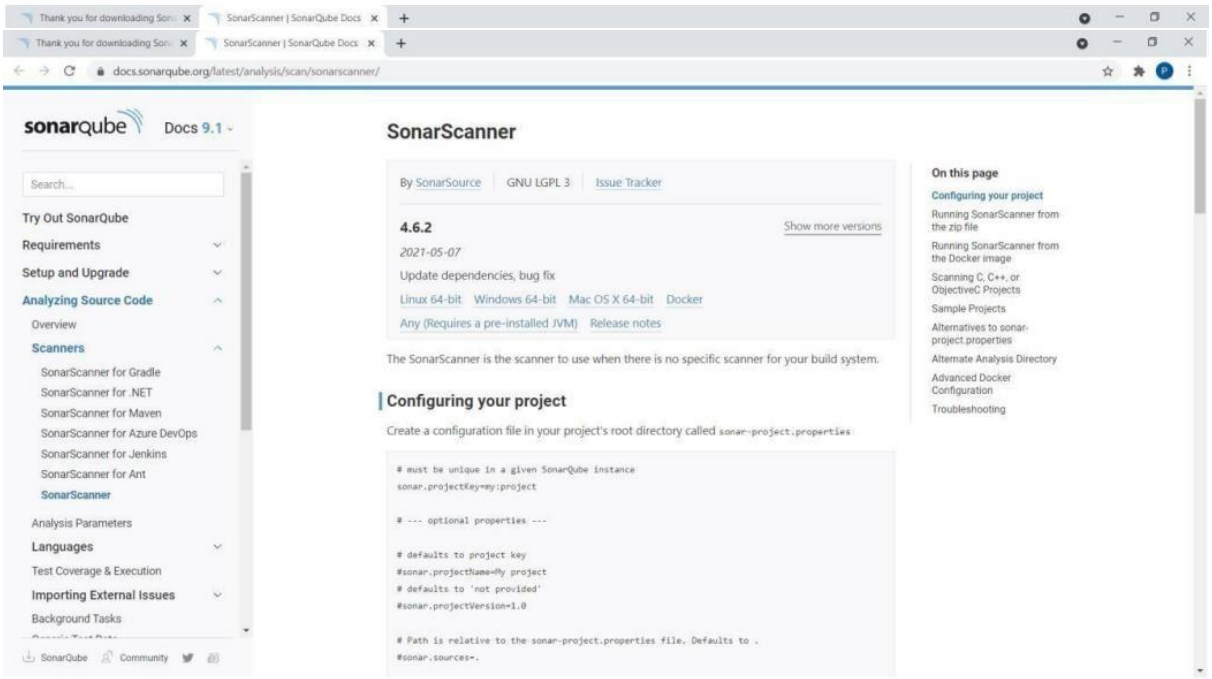
Assignment 7

Aim: TO perform static analysis on python programs using SonarCube SAST processes

LO Mapped: L04 Theory:

Download and Sonar Scanner

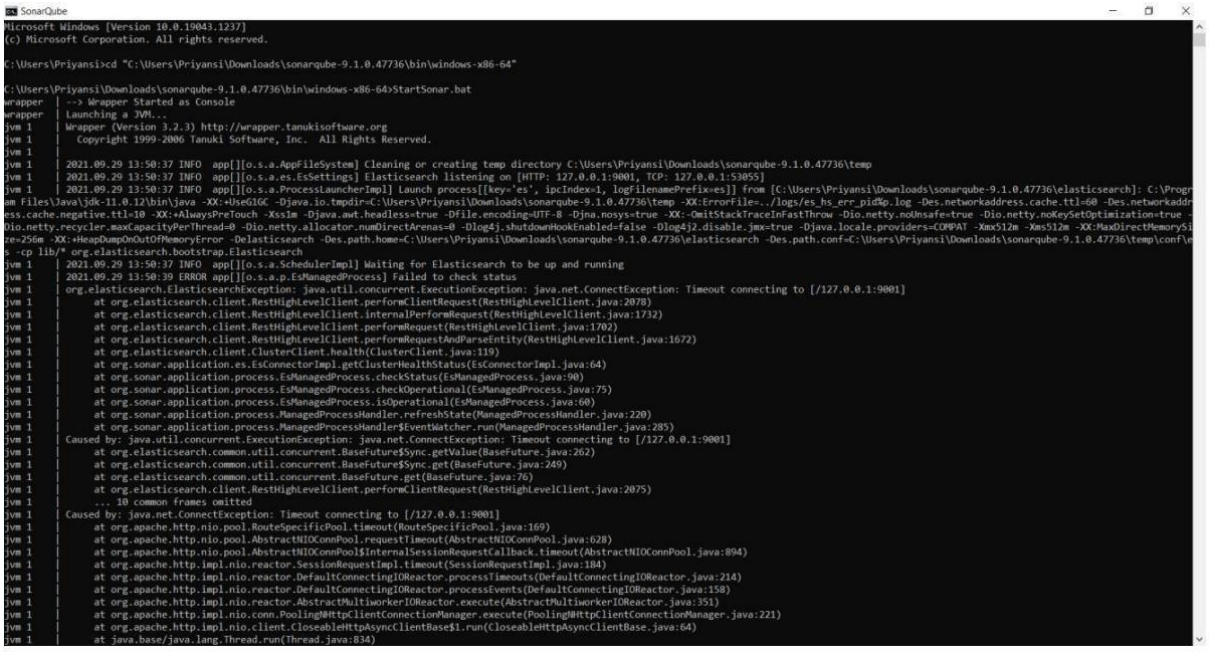


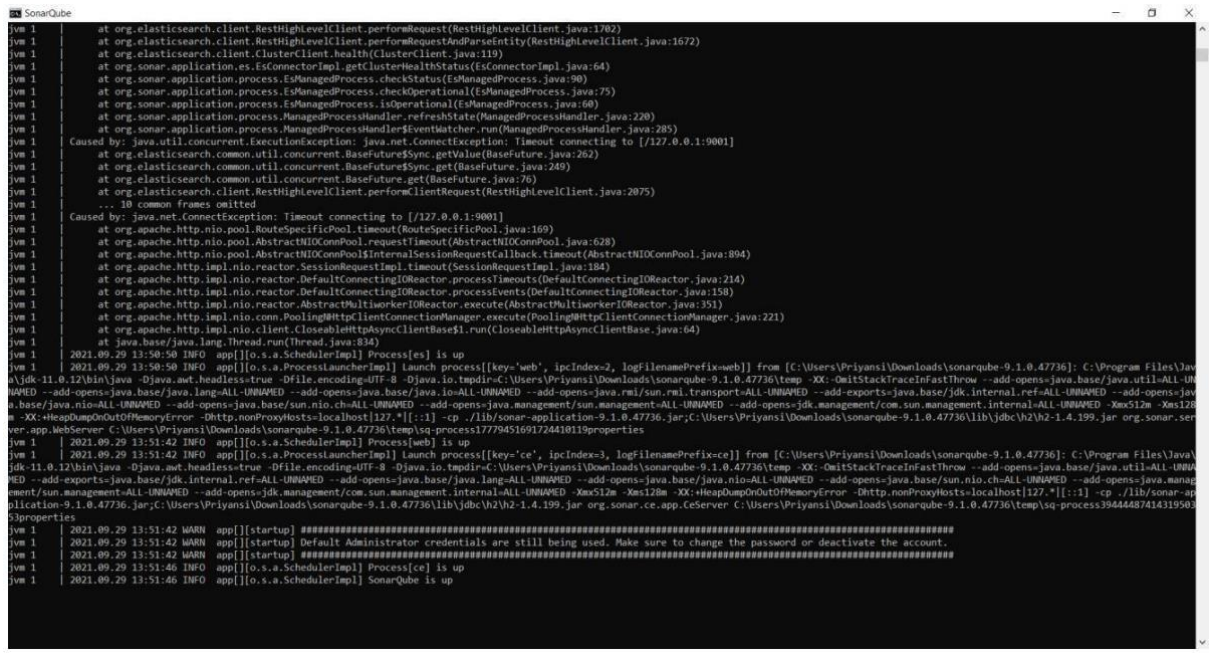


After downloading, set Environment Variables. Add "sonarqube-9.1.0.47736\bin" to Path.

Open command prompt. Run commands:

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

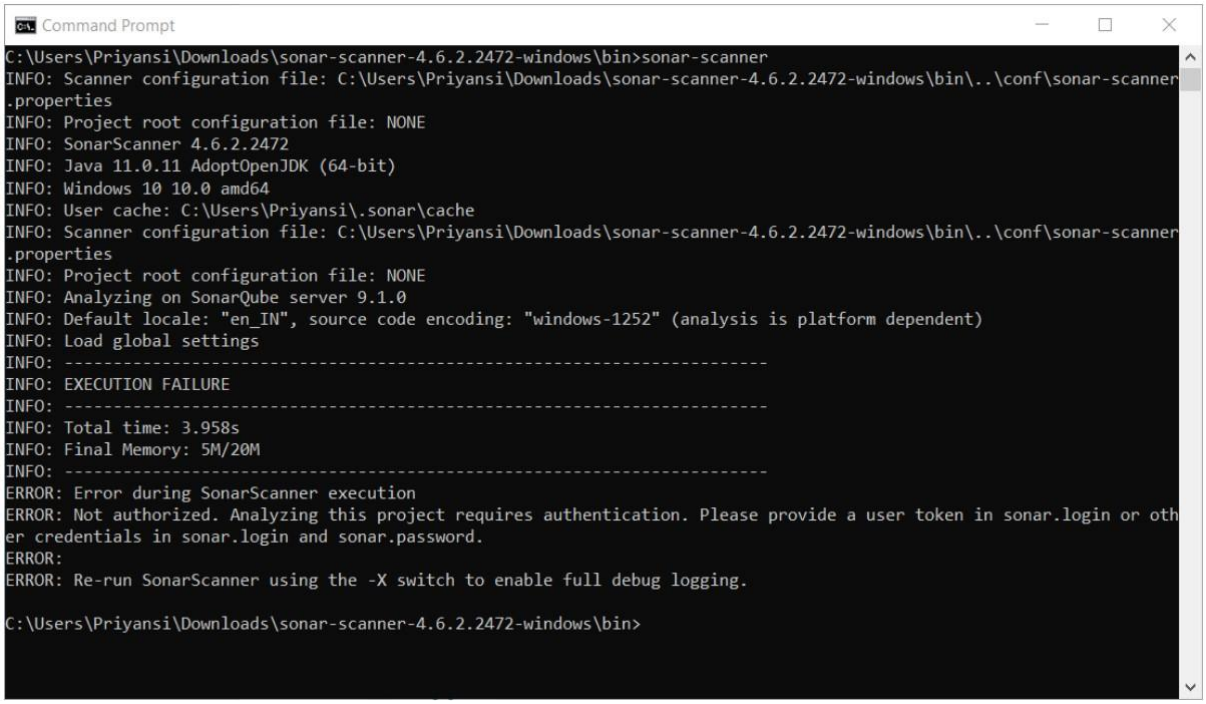




Open another command prompt. Run command: e

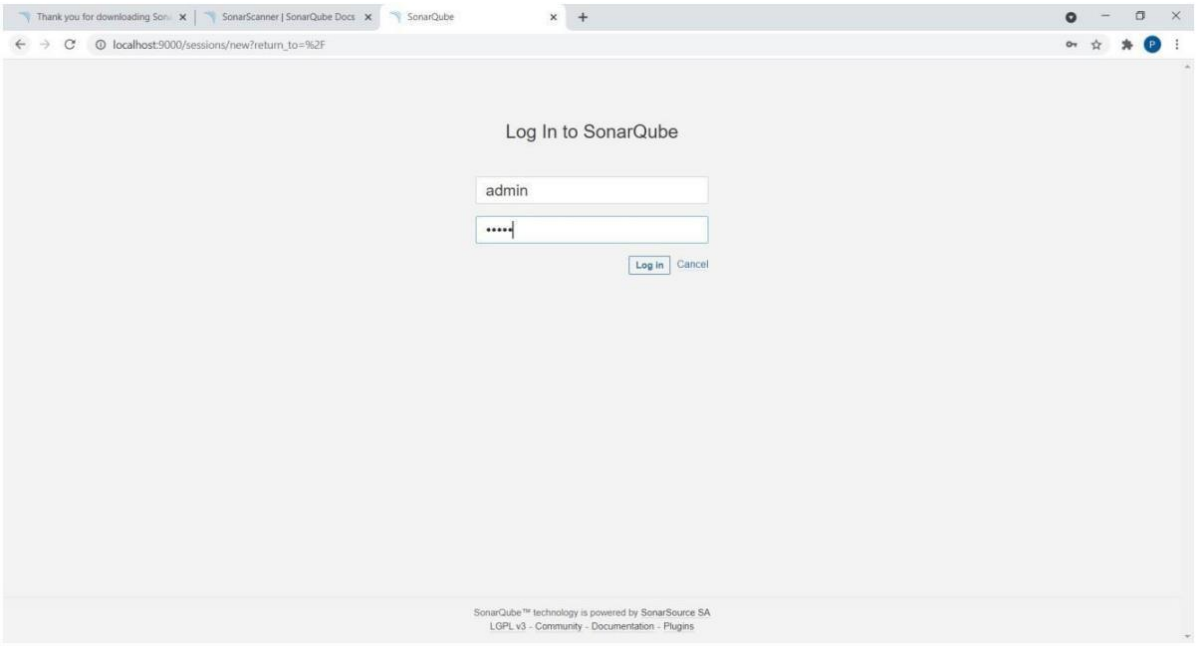
cd "sonar-scanner-4.6.2.2472-windows\bin"

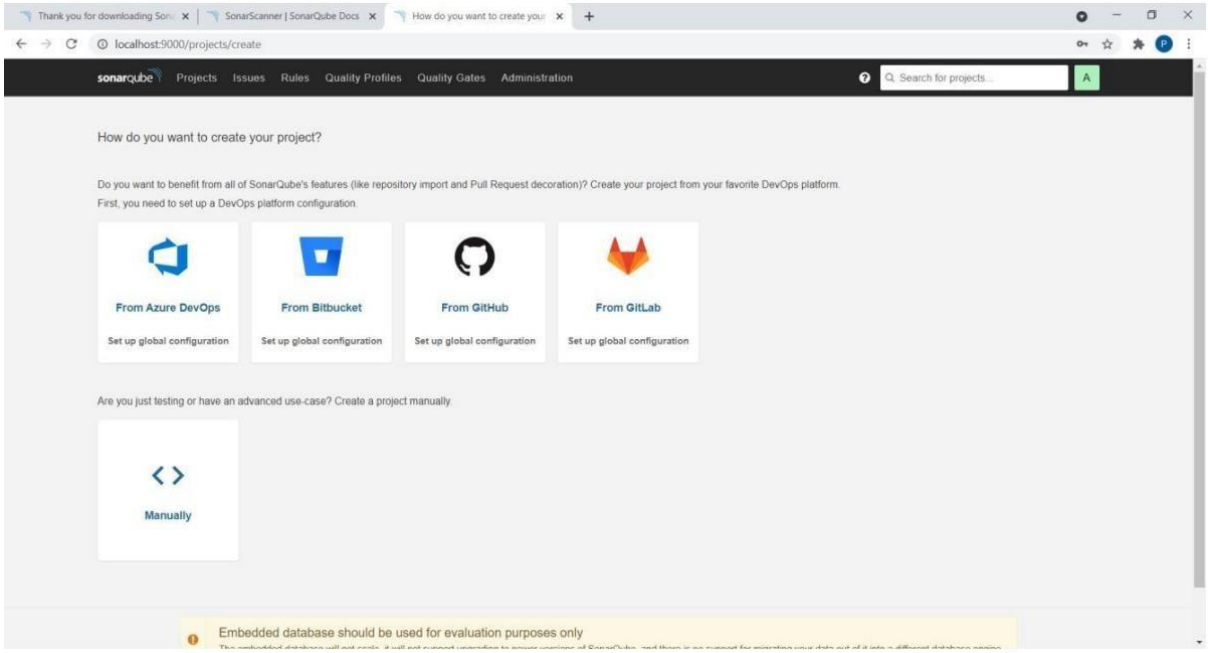
sonar-scanner



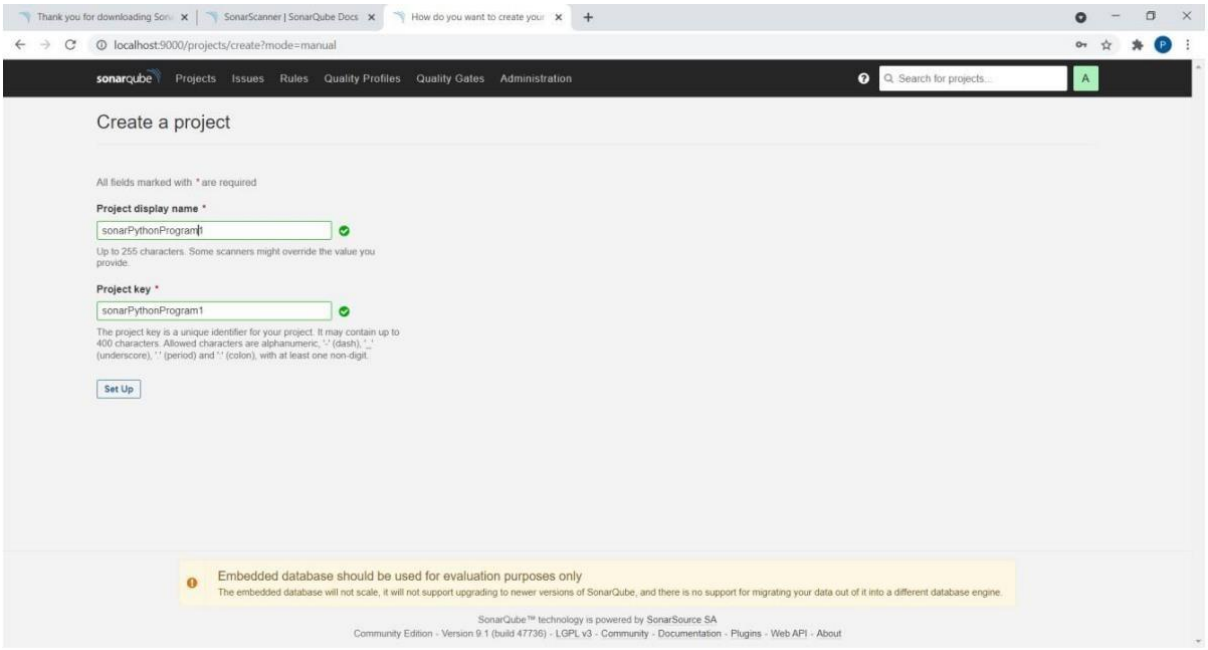
Server up and running on localhost:9000

Login using credentials as User: admin and Password: admin and Set a new password

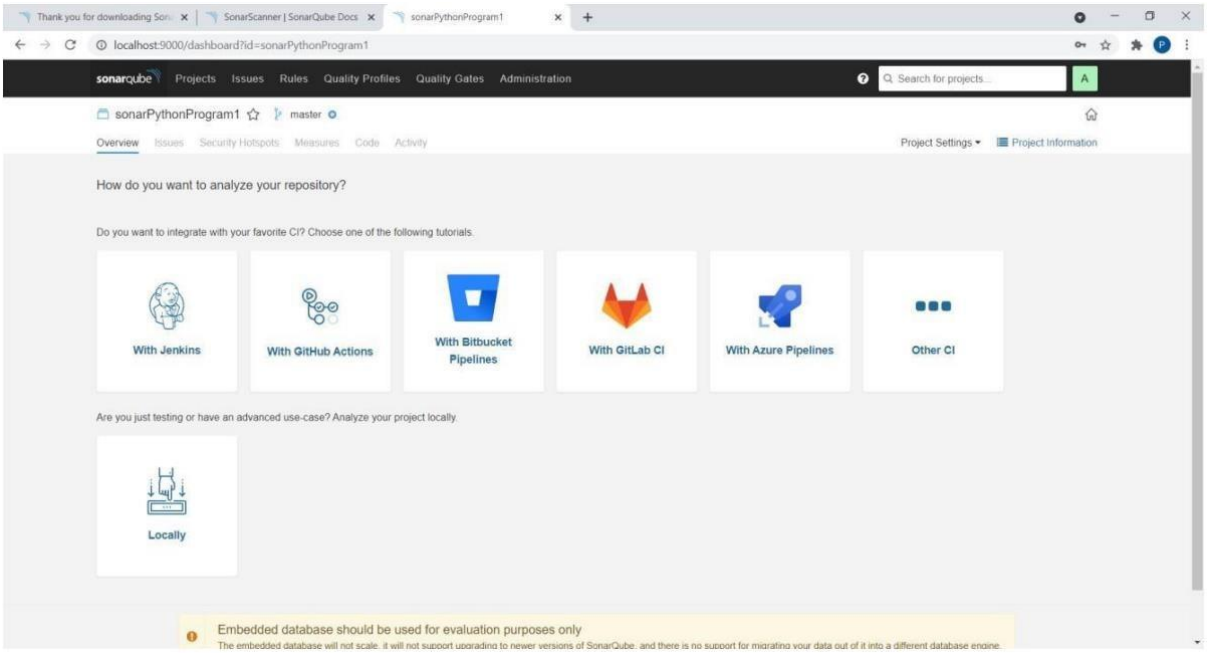




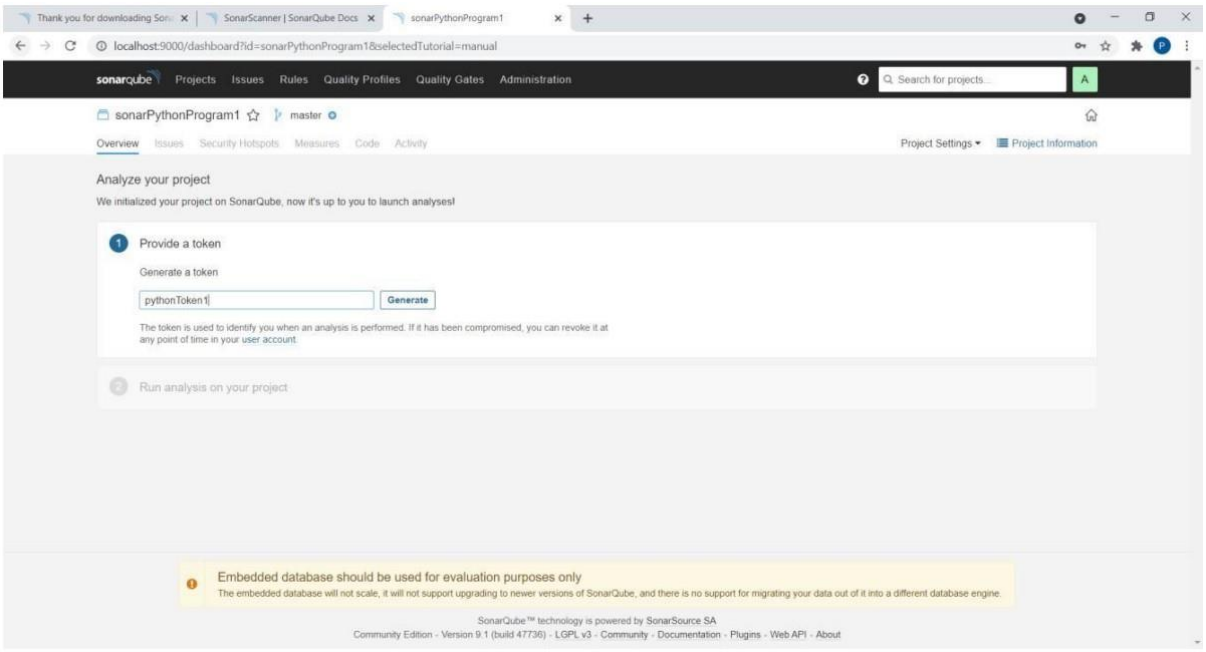
Click on Create a project Manually.



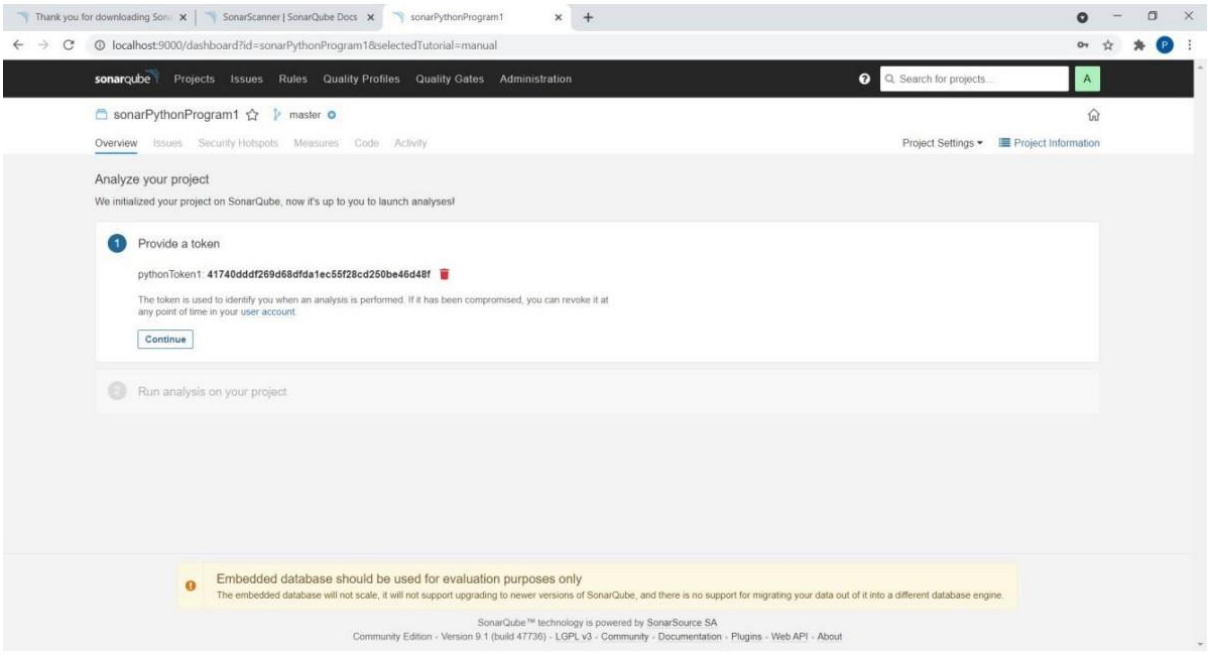
Give any Project display name.



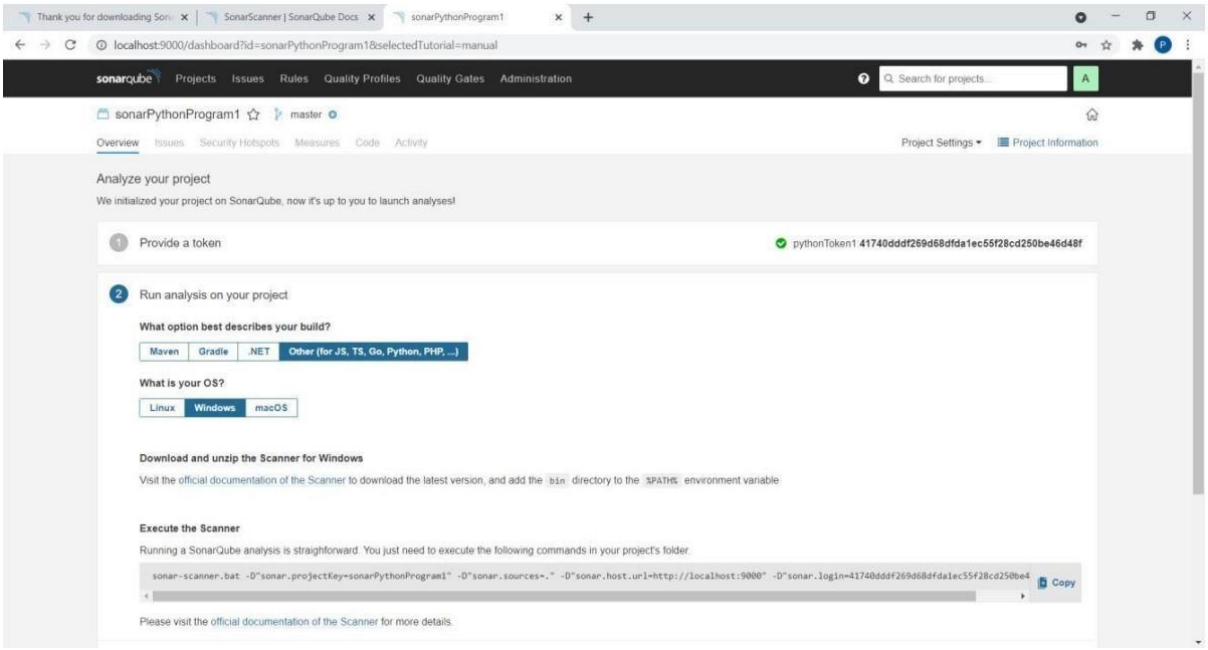
Click on Locally.



Give any name to token and click on Generate.



Click on Continue.



Save a Python program in a

folder. class Solution(object): def

romanTolnt(self, s):

roman = num = while i < len(s): if

i+l<len(s) and s[i:i+2] in roman:

num+=roman[s[i:i+2]]

else: #print(i)

num+=roman[s[i]]

return num obl =

Solution()

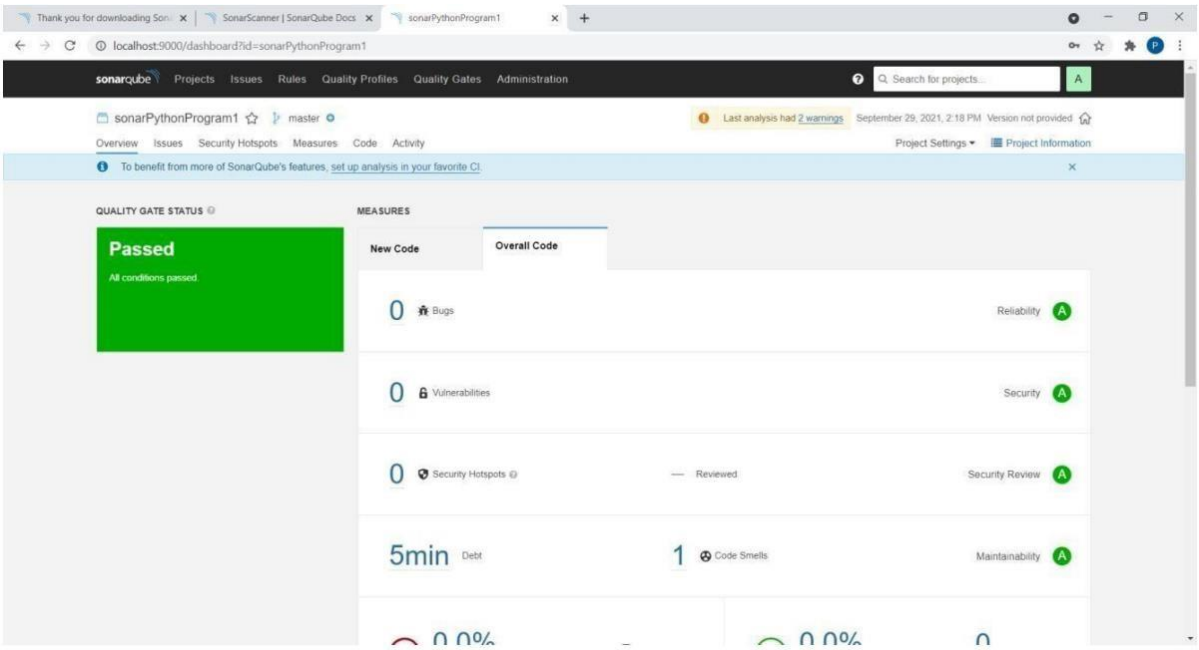
print(obl l"))

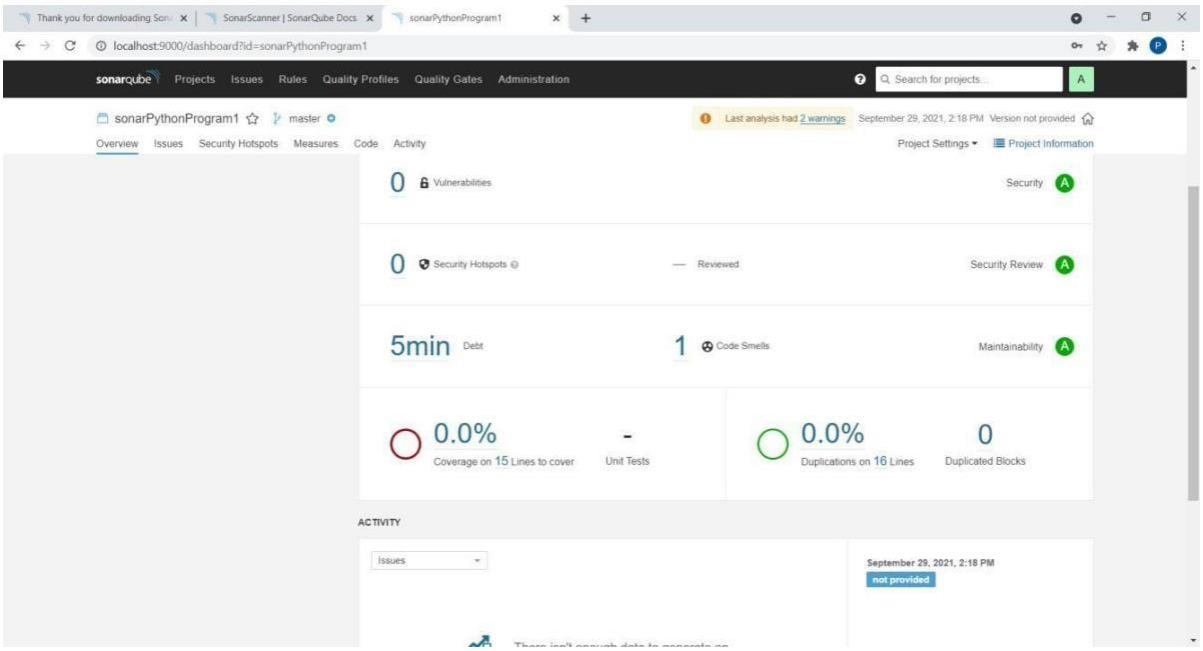
print(obl . l")

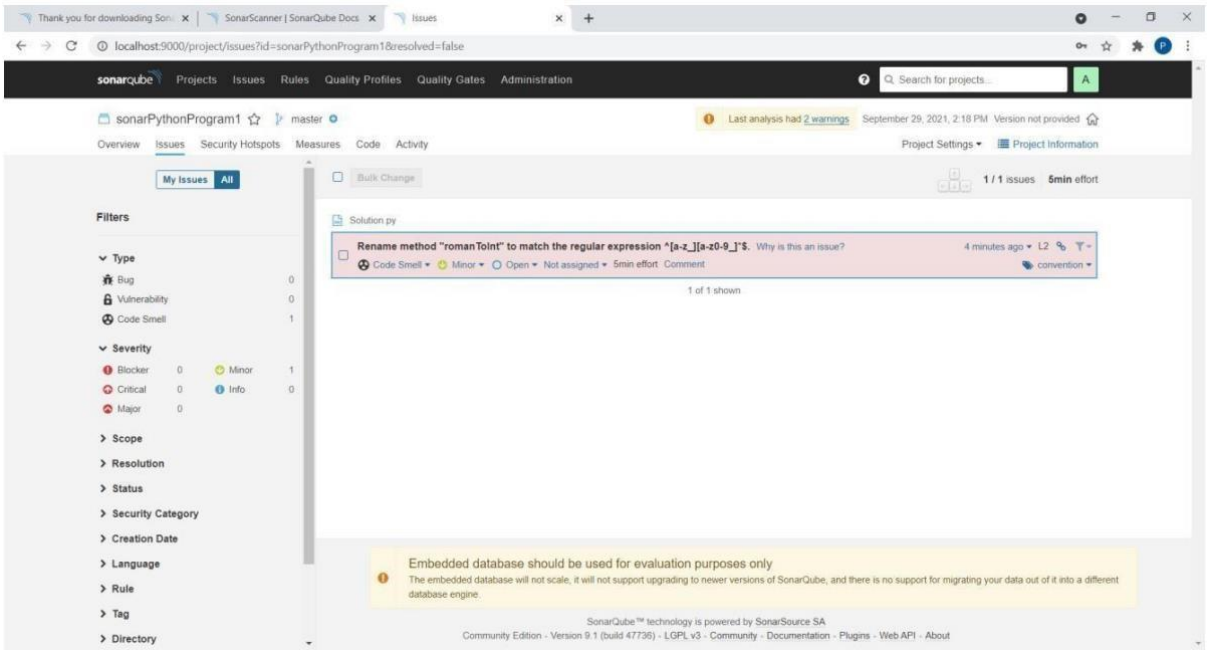
Open command prompt in this folder and Run program using copied command.

sonar-scanner.bat -D"sonar.projectKey=<YourDisplayName>" -
D"sonar.sources=." -D"sonar.host.url=http://localhost:9000" -
D"sonar.login=<YourTokenGeneratedID>'

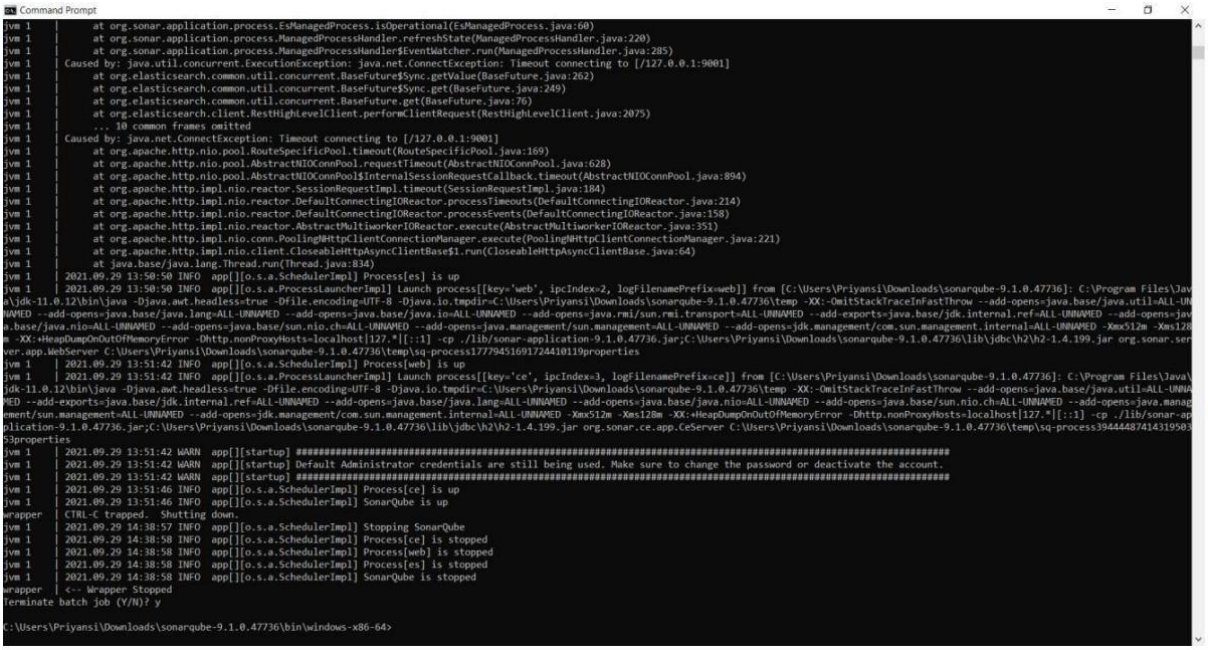
Given below is the inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities.


Thank you for downloading Son
SonarScanner | SonarQube Docs
sonarPythonProgram1
localhost:9000/dashboard?id=sonarPythonProgram1
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
A
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





Press "Ctrl + C" to stop the server.



Conclusion:

In this assignment we implemented static analysis on python programs using SonarCube SAST processes.

Assignment 8

Aim: To understand Continuous Monitoring using Nagios.

LO Mapped: LO5

Theory:

Login to Aws.

Create Ec2 instances on Aws account of any linux os

Then Run the following command in SS

```
ubuntu@ip-172-31-13-219:~$ sudo apt update
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://security.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:5 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe amd64 Packages [8628 kB]
Get:6 http://security.ubuntu.com/ubuntu focal-security/main amd64 Packages [909 kB]
Get:7 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu focal/universe Translation-en [5124 kB]
```

cd /usr/src/

sudo wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz

```
ubuntu@ip-172-31-13-219:/usr/src$ sudo wget https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz
--2021-10-05 10:24:05--  https://github.com/NagiosEnterprises/nagioscore/archive/nagios-4.4.2.tar.gz
Resolving github.com (github.com)... 13.234.176.102
Connecting to github.com (github.com)|13.234.176.102|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2 [following]
--2021-10-05 10:24:05--  https://codeload.github.com/NagiosEnterprises/nagioscore/tar.gz/nagios-4.4.2
Resolving codeload.github.com (codeload.github.com)... 13.233.43.20
Connecting to codeload.github.com (codeload.github.com)|13.233.43.20|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/x-gzip]
Saving to: 'nagios-4.4.2.tar.gz'

nagios-4.4.2.tar.gz                     [    <=>                                                 ]  10.78M  16.9MB/s    in 0.6s

2021-10-05 10:24:06 (16.9 MB/s) - 'nagios-4.4.2.tar.gz' saved [11301457]
```

sudo tar zxf nagios-*.tar.gz

cd nagioscore-nagios-*/

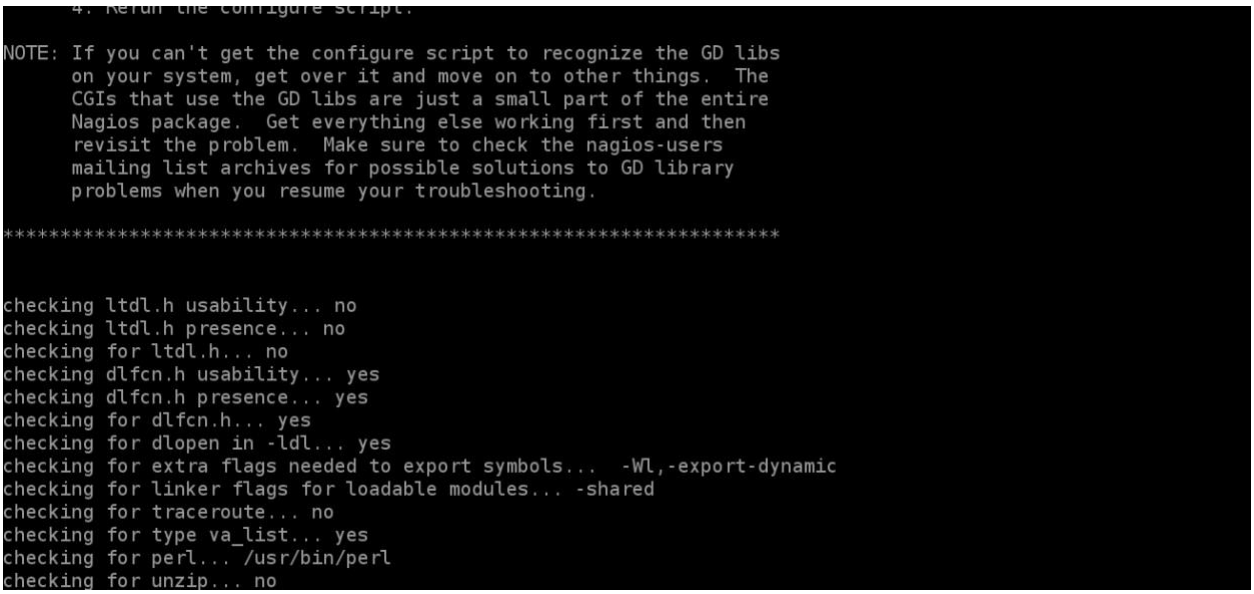
```
ubuntu@ip-172-31-13-219:/usr/src$ cd nagioscore-nagios-*/
ubuntu@ip-172-31-13-219:/usr/src/nagioscore-nagios-4.4.2$
```

Now finally run the following command

sudo ./configure --with-httpd-conf=/etc/apache2/sites-enabled

if error comes install c compiler on the linux by following this link

https://linuxize.com/post/how-to-install-gcc-compiler-on-ubuntu-18-04/ Finally-



Now let us install plugins by

sudo wget -O nagios-plugins.tar.gz https://github.com/nagios-plugins/nagiosplugins/archive/release-2.2.1.tar.g

then

sudo tar zxf nagios-plugins.tar.gz

then

cd nagios-plugins-release-2.2.1 finally start

and then check status of nagi os

sudo systemctl start nagios

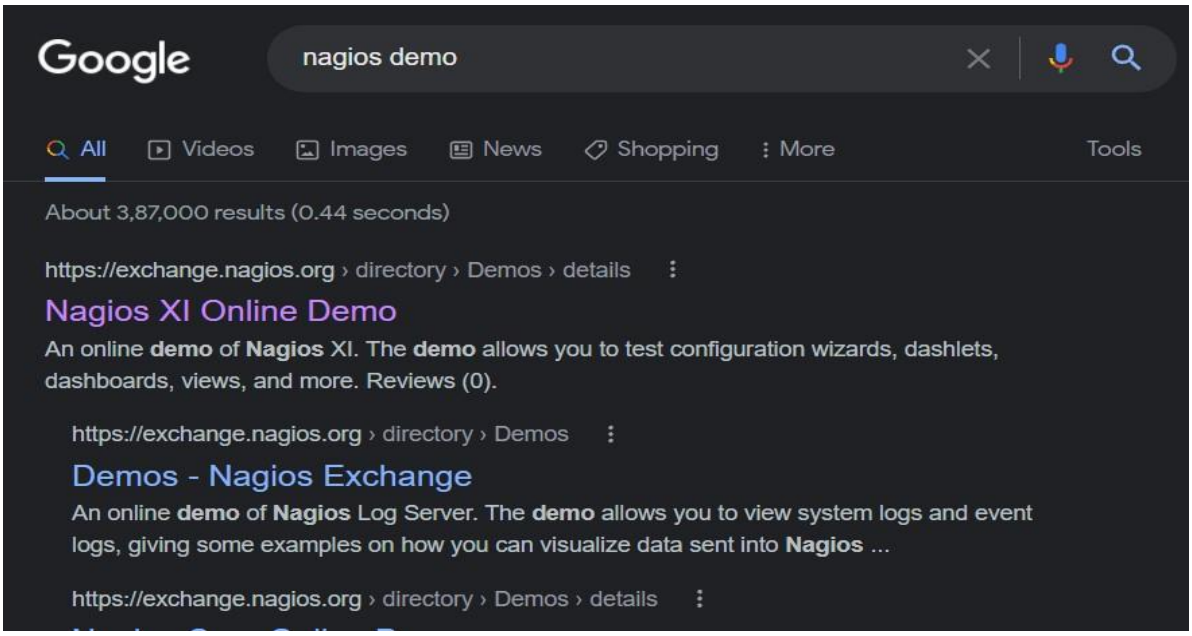
sudo systemctl status nagios

```
* nagios.service - Nagios Core 4.4.2
   Loaded: loaded (/lib/systemd/system/nagios.service; enabled; vendor preset: enabl
   Active: active (running) since Fri 2018-11-16 14:54:21 PST; 1s ago
     Docs: https://www.nagios.org/documentation
  Process: 18294 ExecStopPost=/bin/rm -f /usr/local/nagios/var/rw/nagios.cmd (code=e
  Process: 18293 ExecStop=/bin/kill -s TERM ${MAINPID} (code=exited, status=0/SUCCES
  Process: 18315 ExecStart=/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nag
  Process: 18313 ExecStartPre=/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/
 Main PID: 18325 (nagios)
    Tasks: 6 (limit: 2319)
   CGroup: /system.slice/nagios.service
```

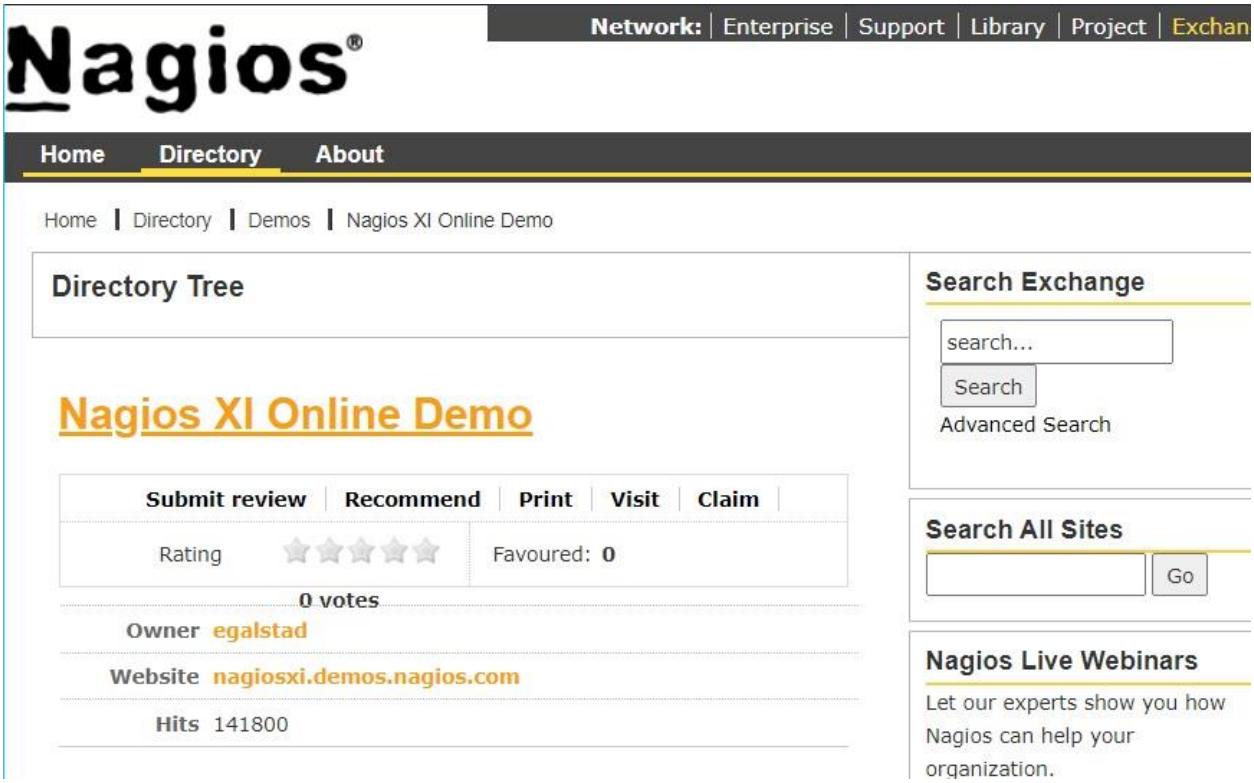
Steps-

Go to google.com, Search NagiOs Demo

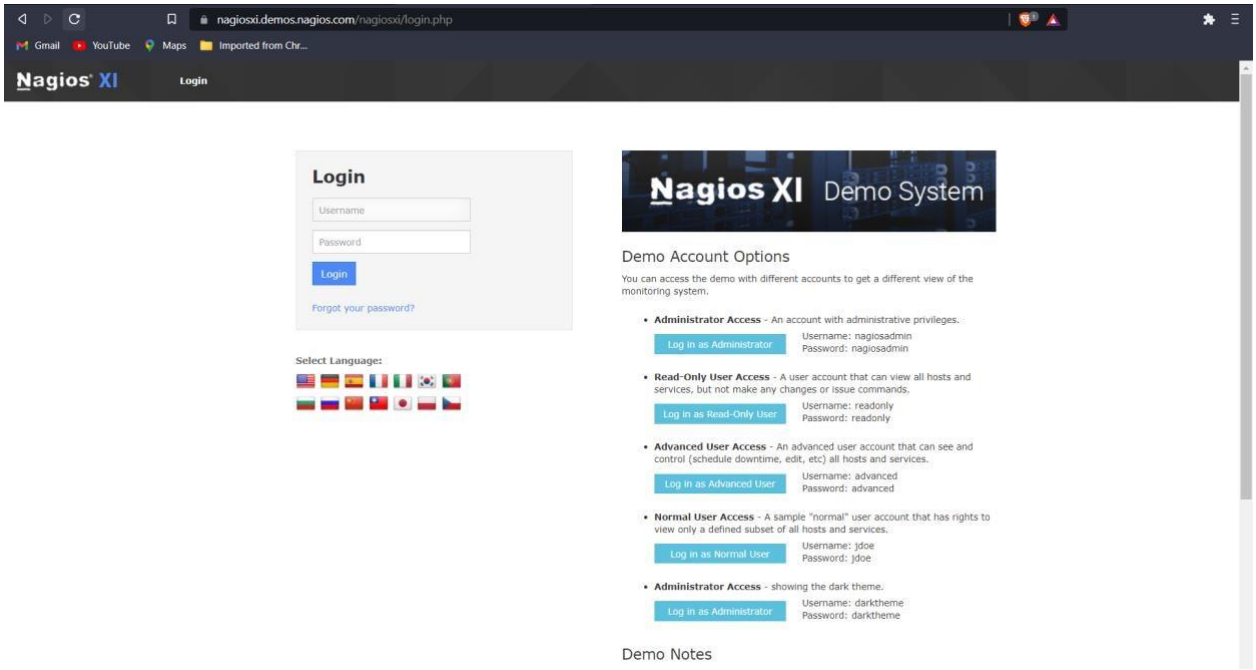
Click on the first link shown below



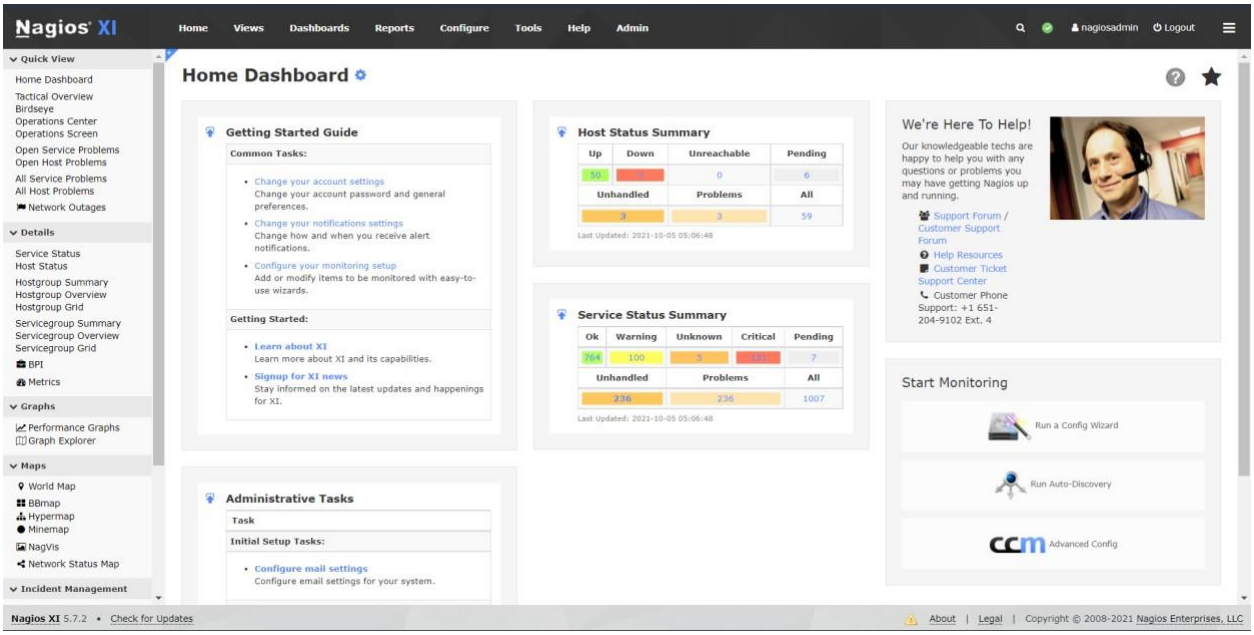
Now click on the website-



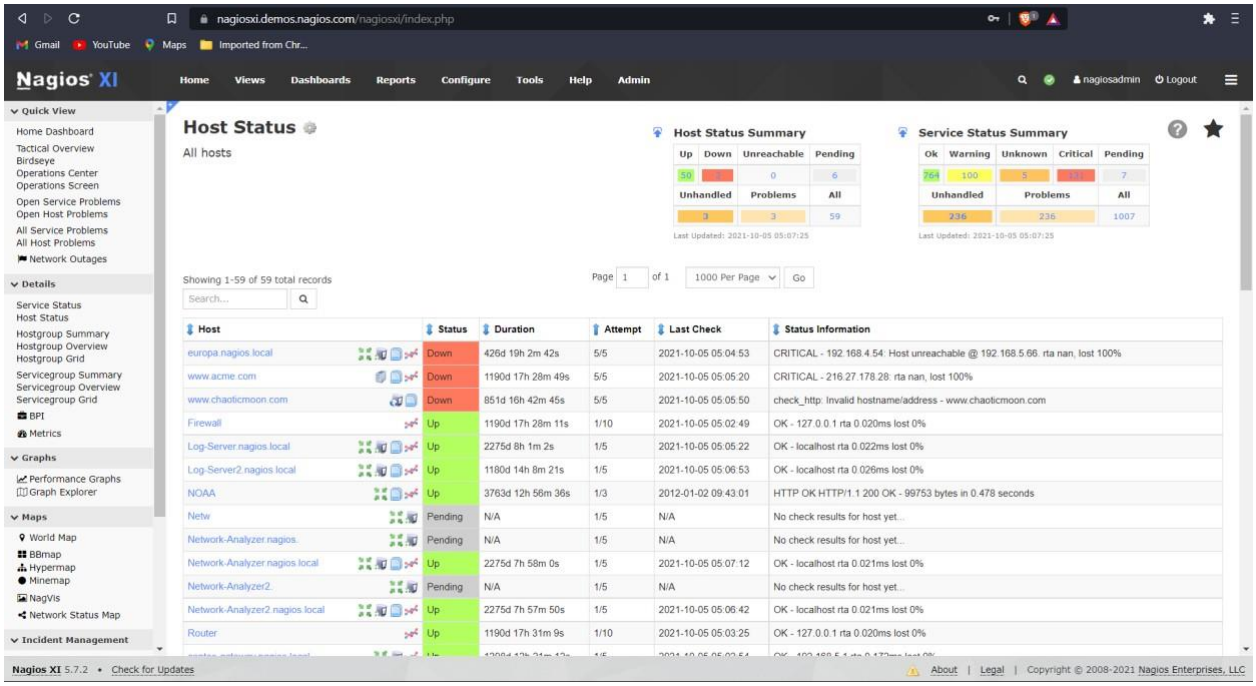
Now click on login as administrator



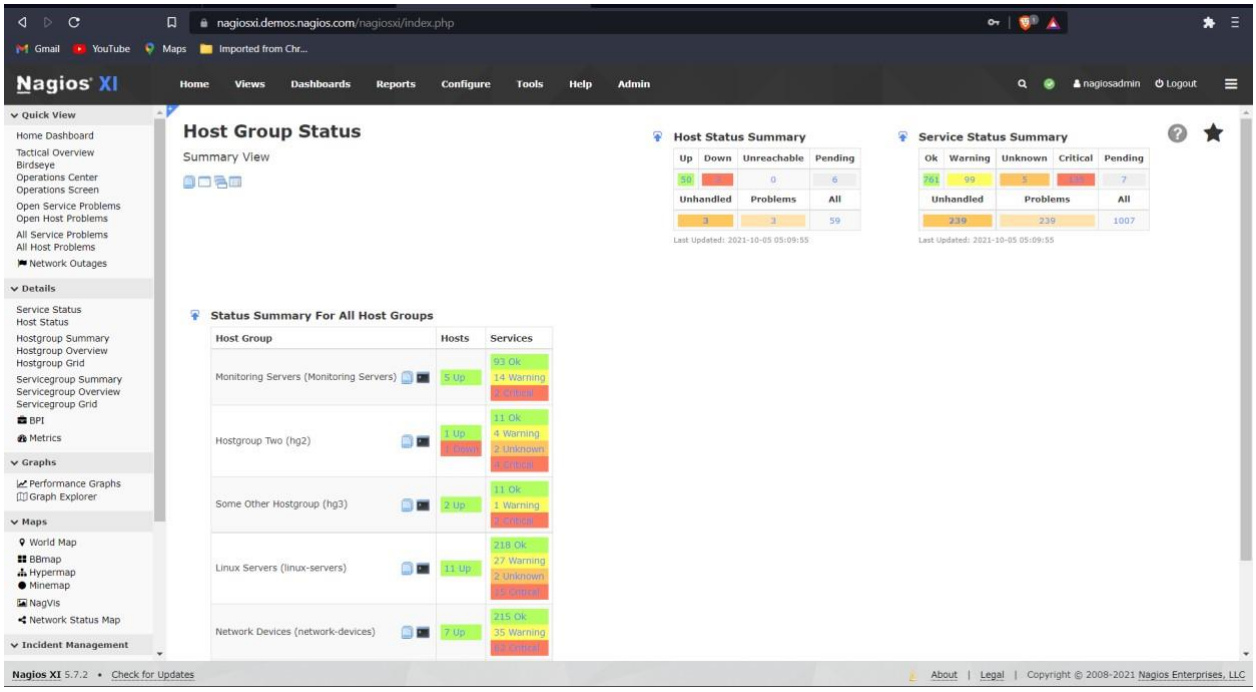
It will have interface like this



Now click on Host status-



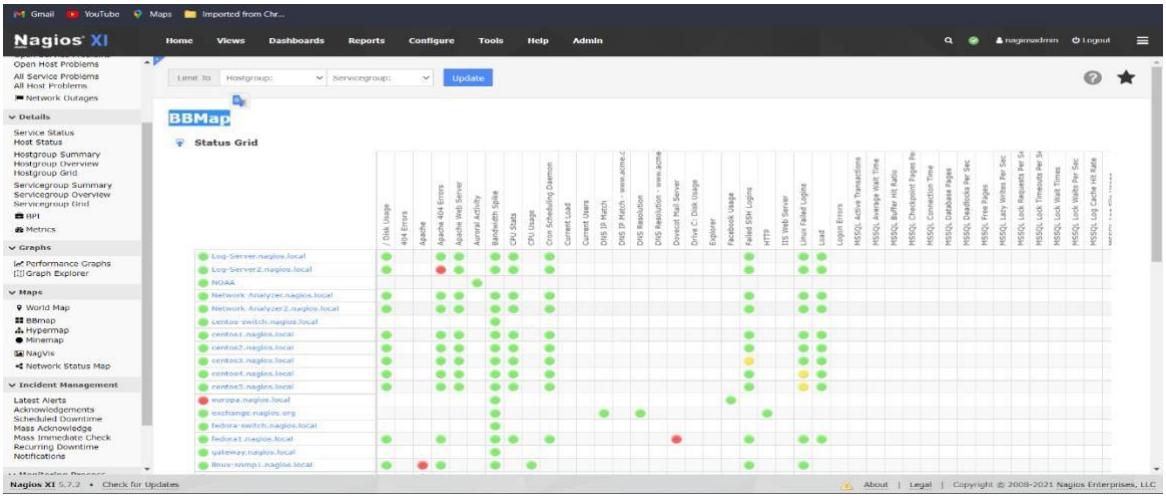
In the above image one can see Host Status Summary and Service Status Summary also how many host are up, down and also errors in detail Now click on Host Group Status.



Here we can see Status Summary For All Host Groups

Now we click on BBMap

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



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

Conclusion: In this assignment we learned about how to continuously monitor Nagios commands

ASSIGNMENT 9

AIM: To understand Lambda Function and create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.

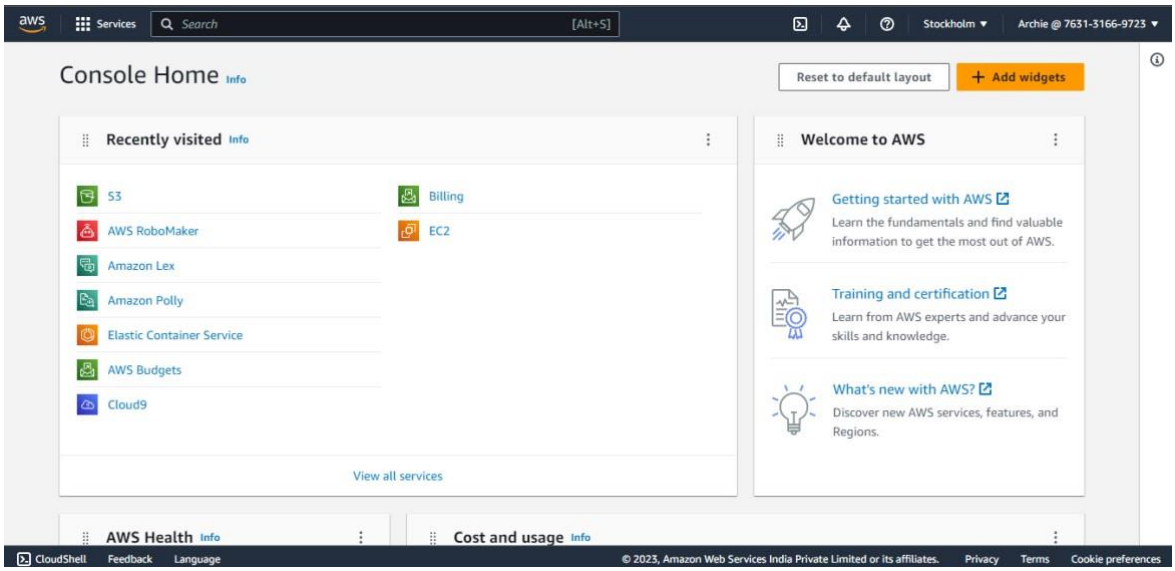
LO MAPPED: LO1, LO5

THEORY:

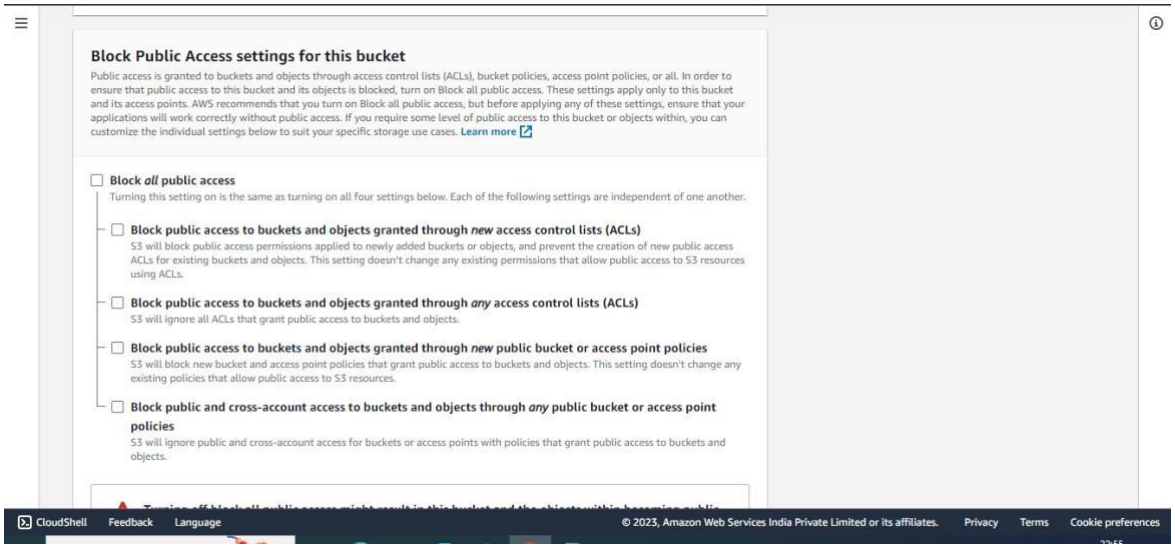
You can use Lambda to process event notifications from Amazon Simple Storage Service. Amazon S3 can send an event to a Lambda function when an object is created or deleted. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.

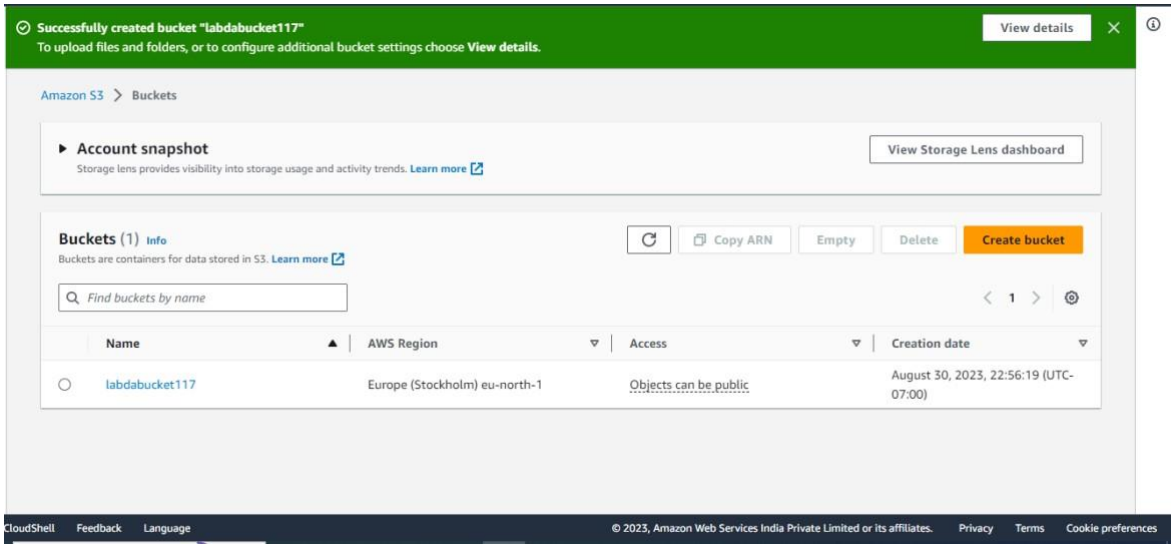
STEPS TO FOLLOW:

1. Log in as IAM User

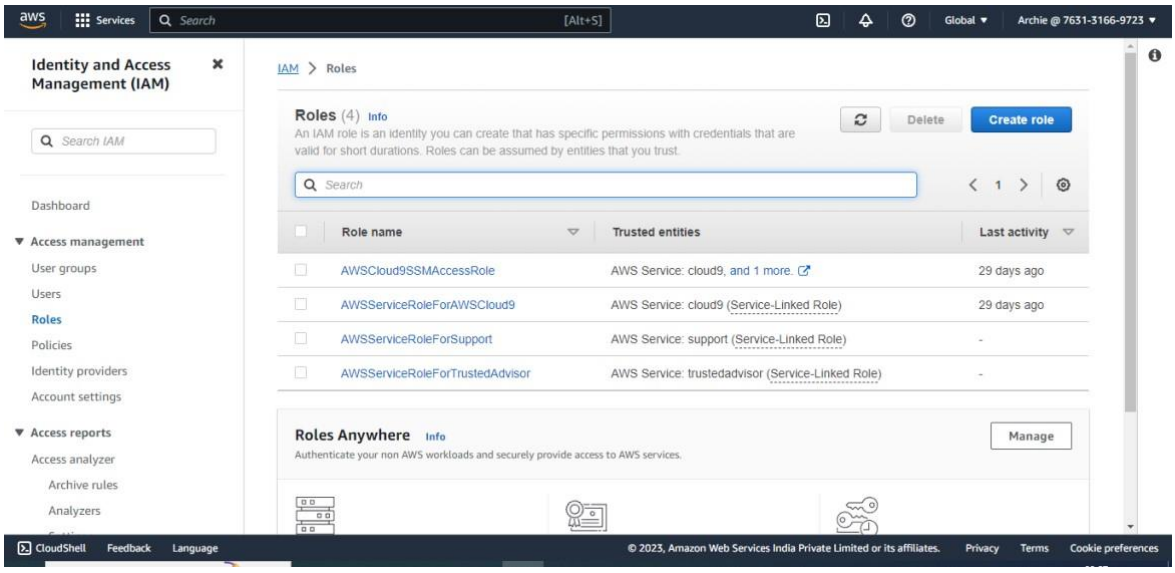


2. Create a S3 bucket and Enable the "Block all public access"

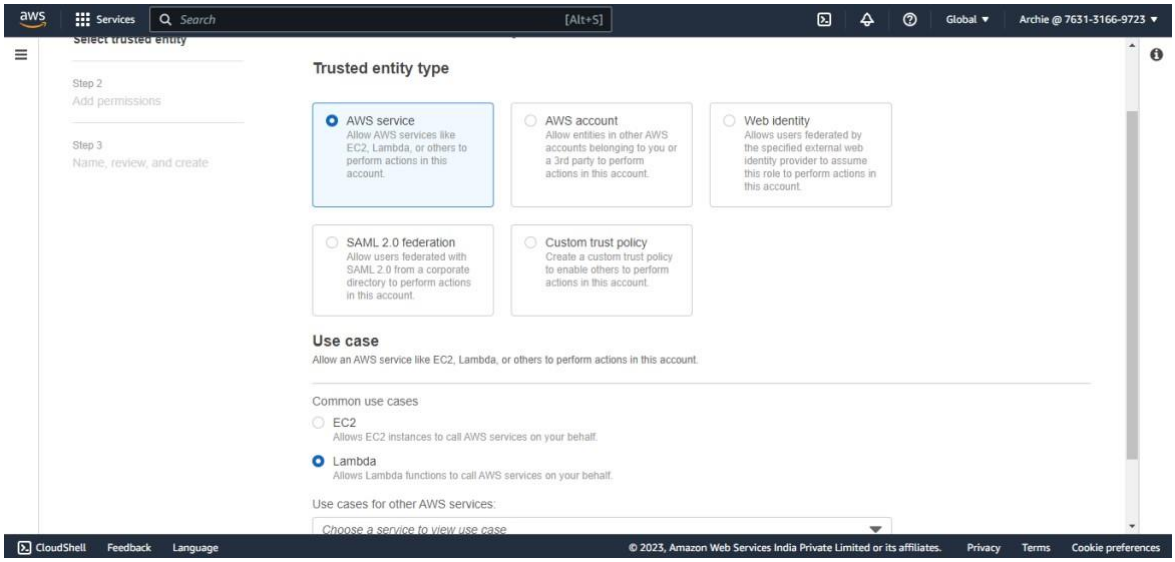




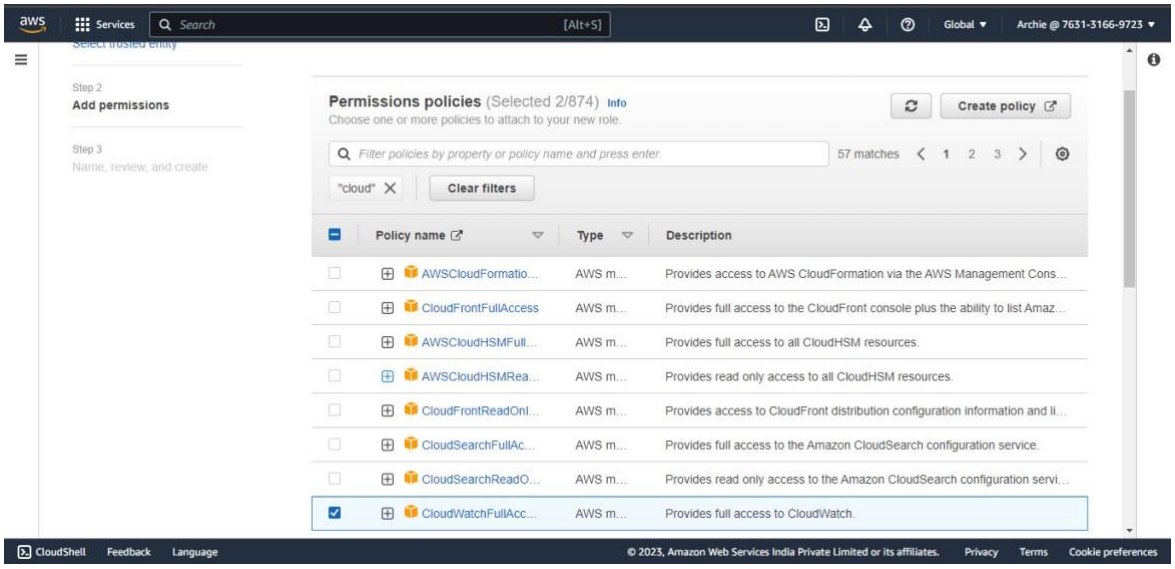
3. Search IAM on the console and go to "Roles". Click on Create Roles

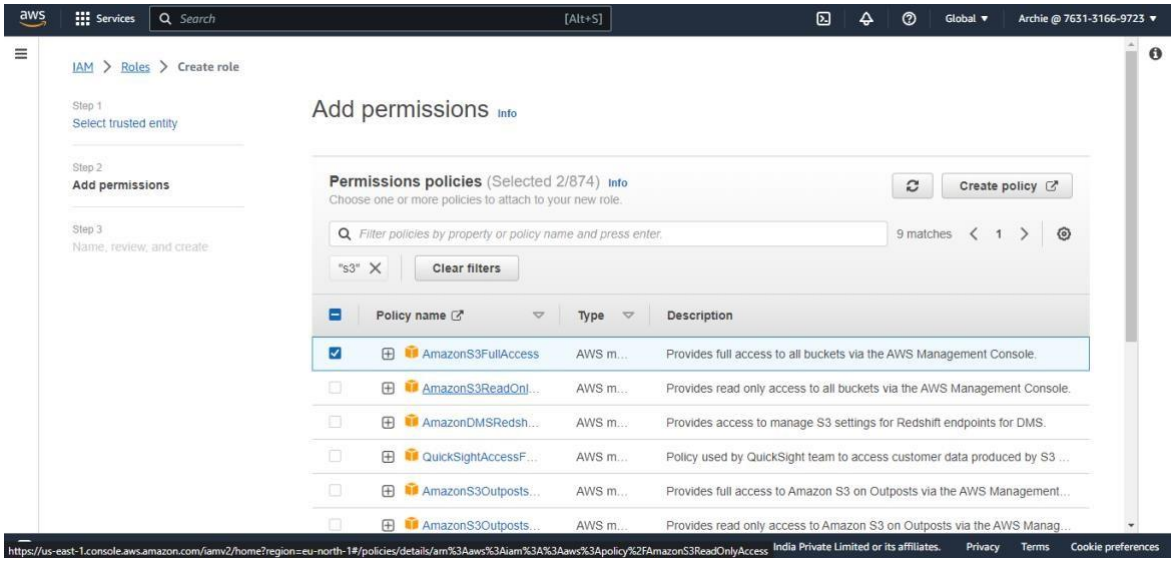


4. Select the options of "AWS service" and "lambda"

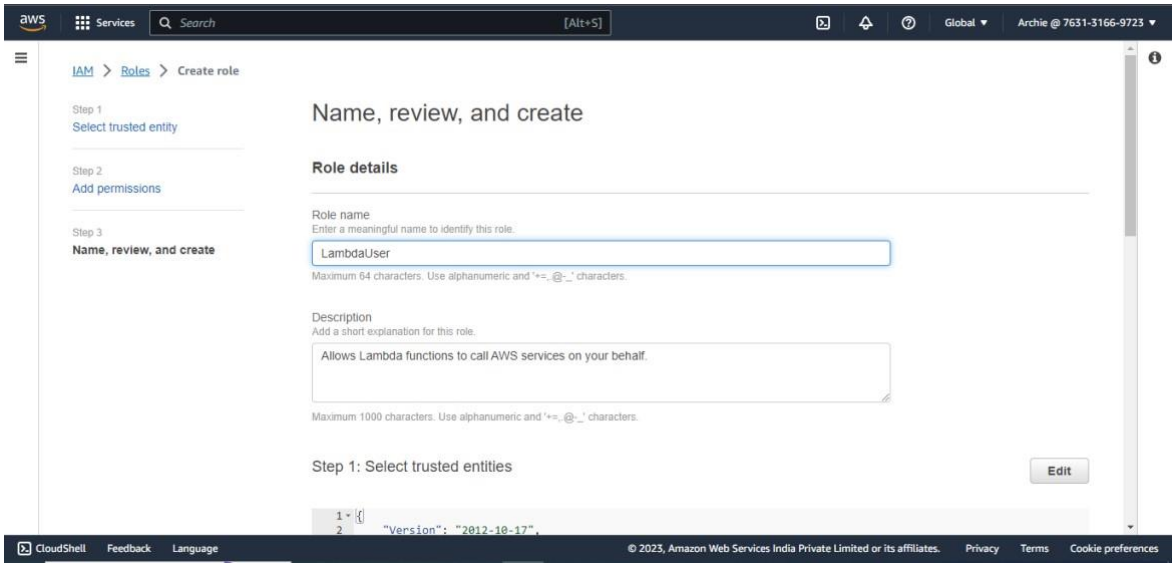


5. Enable the "CloudWatchFullAccess" and "AmazonS3FullAccess"

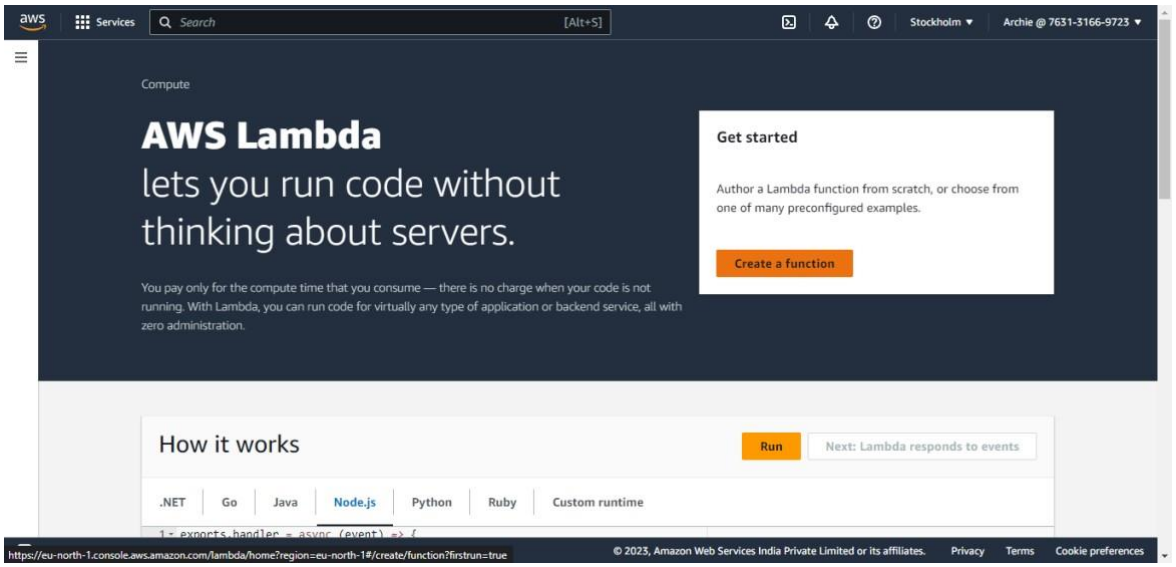




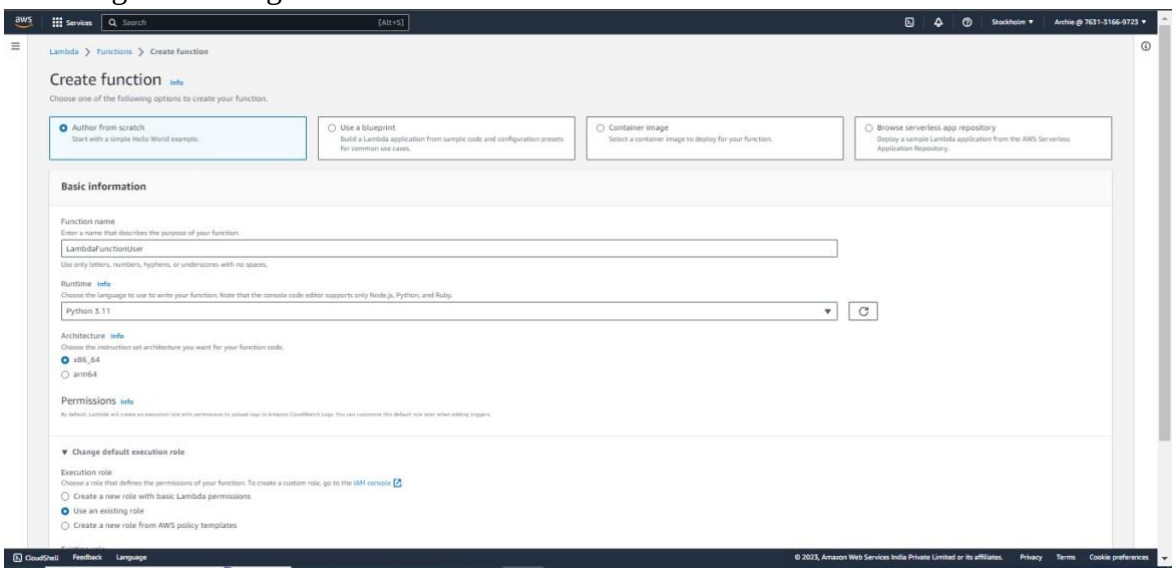
6. Click on Create Role and you will be redirected to this dashboard. Give the Role a Name and Click on Done.



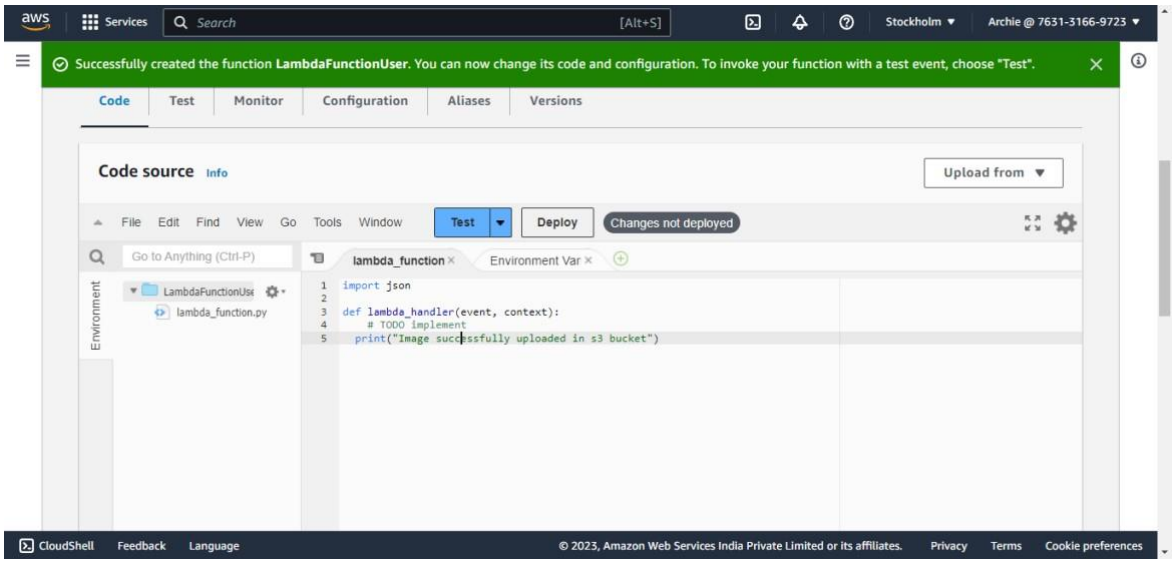
7. Search for Lambda in the console and click on Create Function.



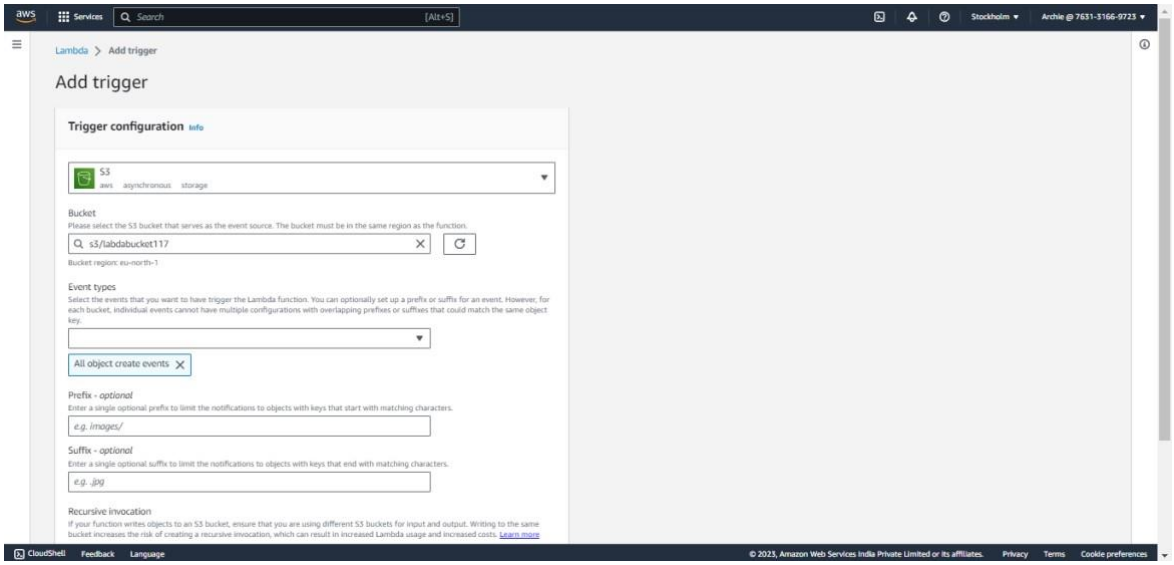
8. Change the settings of the Lambda Function.



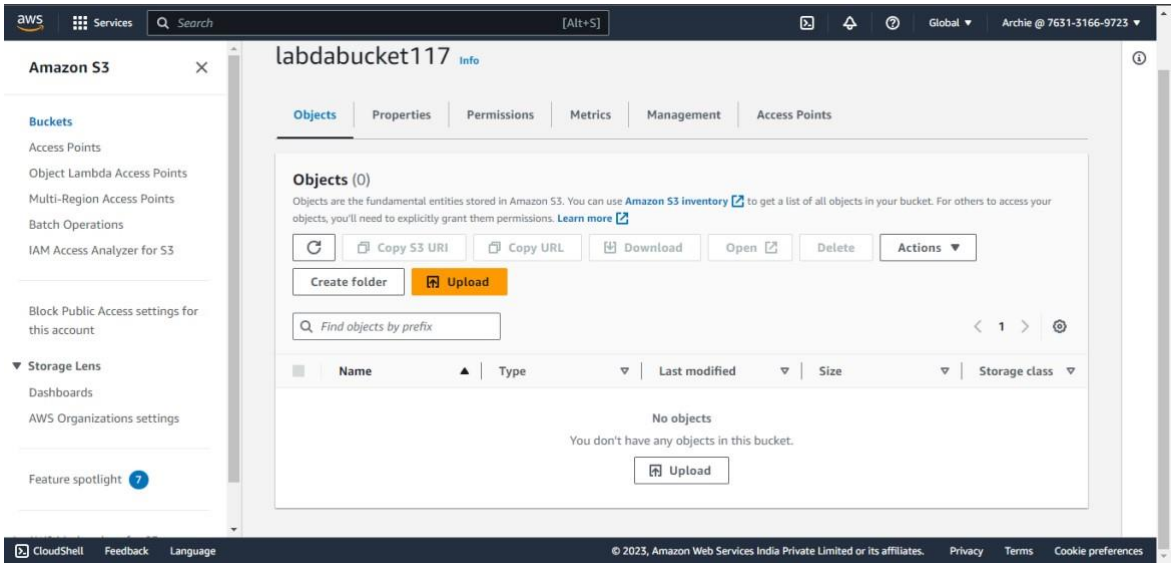
9. Add the python code and click on deploy to save the changes.

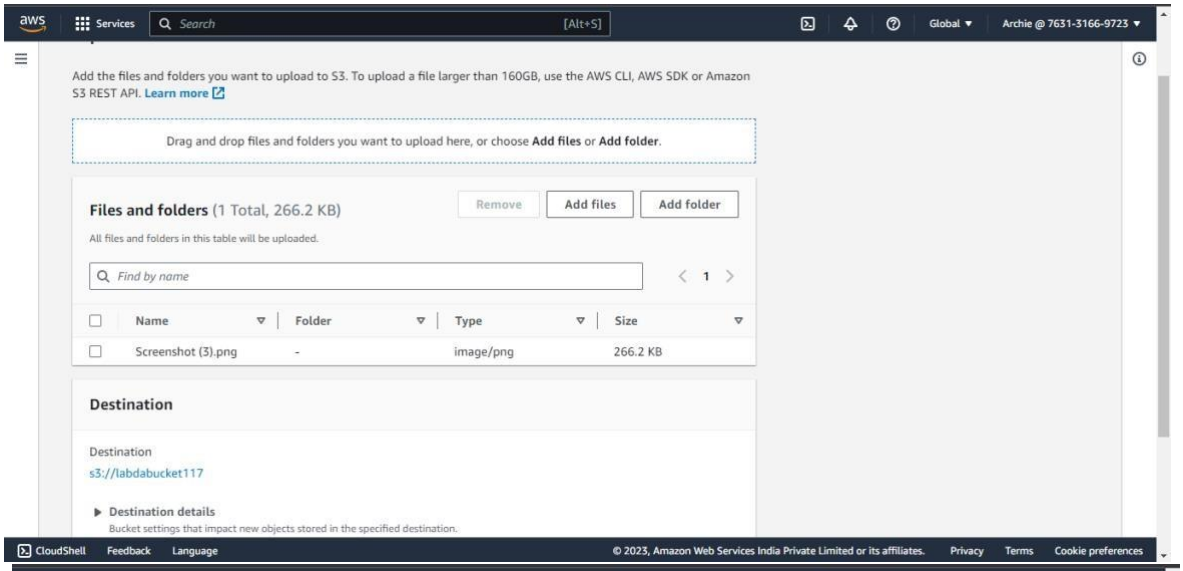


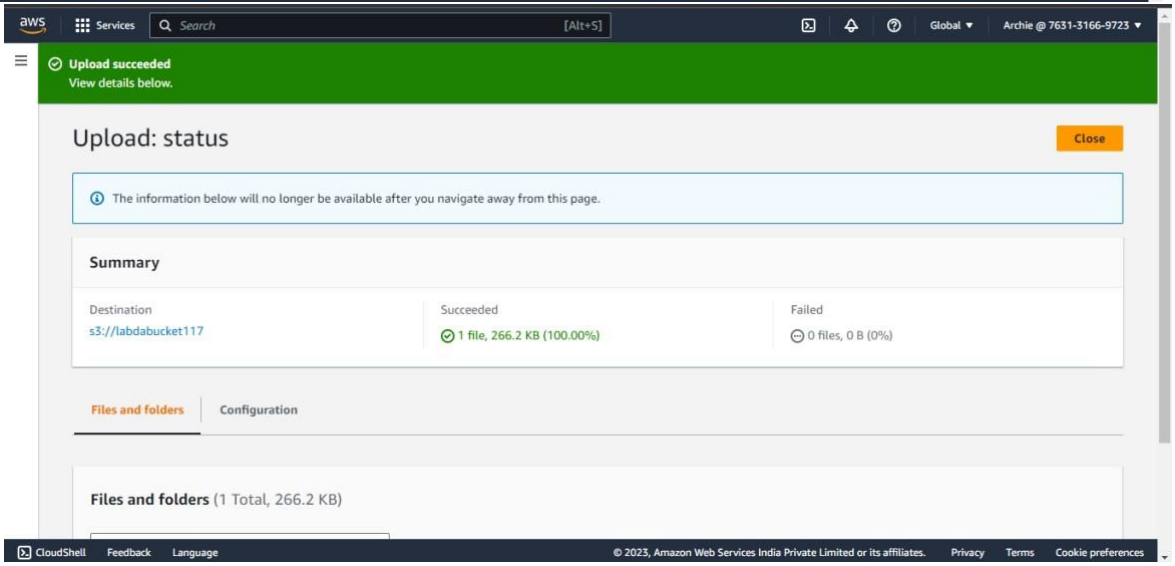
10. Scroll above and click on ADD TRIGGER. Select the following options and click on Done.



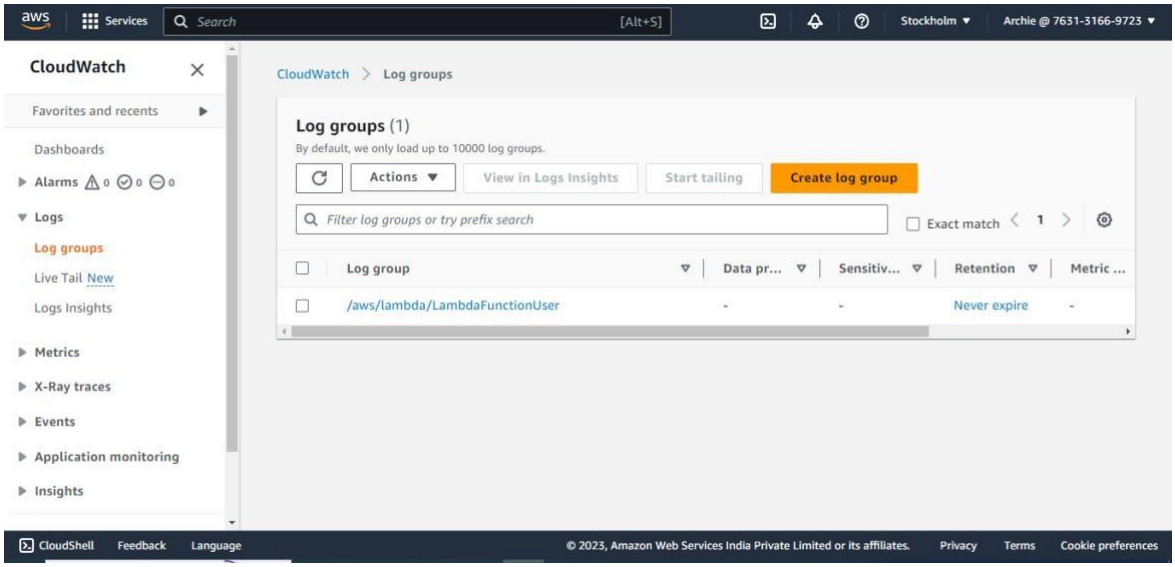
11. Go to S3 bucket and click on Add files. Select a image and click on Upload.



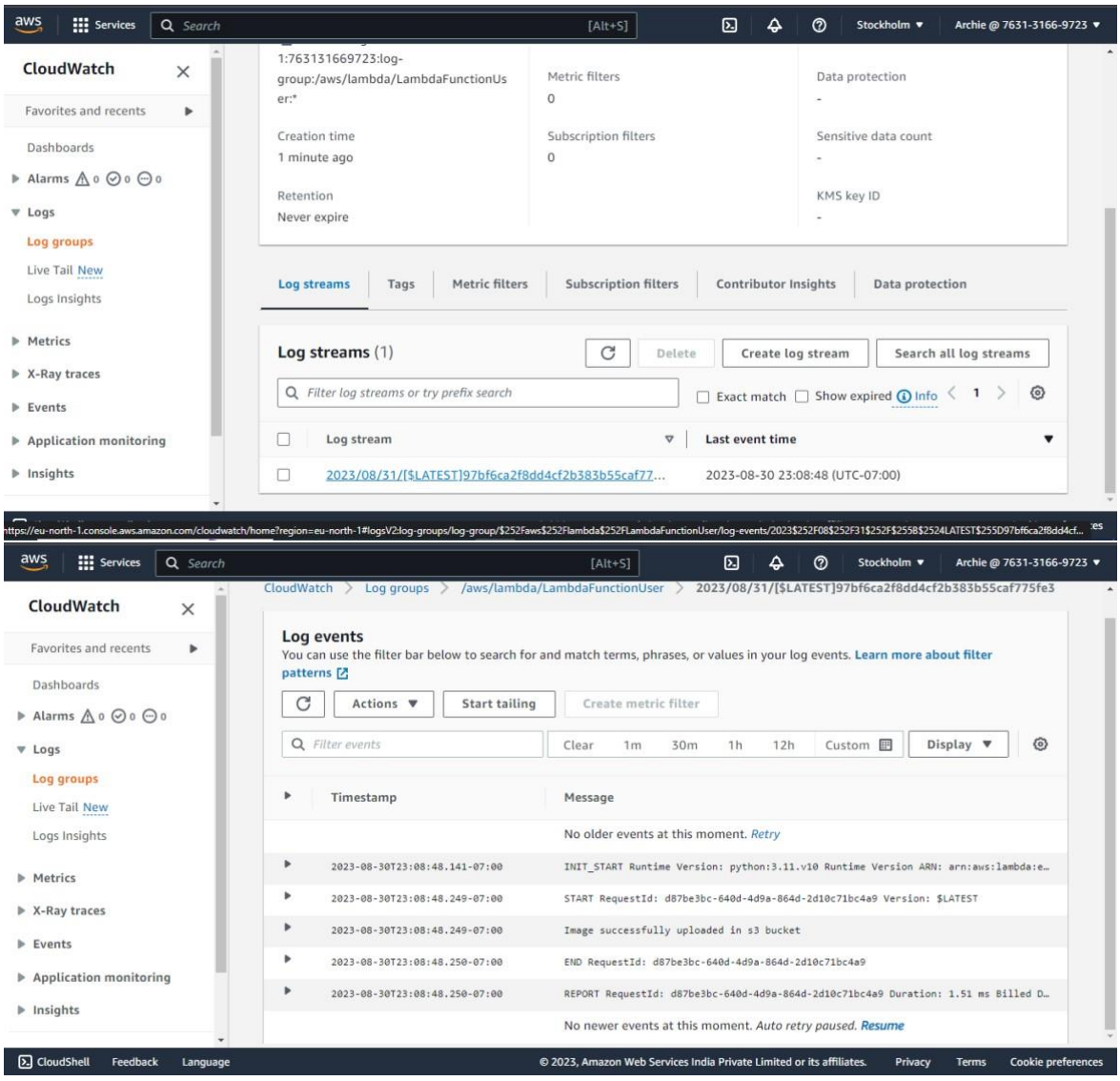




12. Search CloudWatch and go to Log groups. Select the existing Log Group.



13. Click on the link provided and then you will see the message displayed.



CONCLUSION: In this assignment, we learnt how to create a Lambda function which will log "An Image has been added" once you add an object to a specific bucket in S3.

Assignment No. 10

Aim : Students are expected to study AWS Lambda function and create the following Lambda functions using Python:

To add data to Dynamo DB database.

Theory :

Creating an AWS Lambda function to add data to a DynamoDB database using Python is a common use case for serverless applications. You can use AWS Lambda and the AWS SDK for Python (Boto3) to achieve this. Here's a step-bystep guide:

Set up Your Environment:

Ensure you have an AWS account and access to the AWS Management Console.

Install the AWS Command Line Interface (CLI) and configure it with your AWS credentials.

Create a DynamoDB Table:

In the AWS DynamoDB Management Console, create a table with the desired schema. Note the table name and primary key (usually the partition key).

Create a Lambda Function:

Go to the AWS Lambda Management Console.

Click the "Create function" button.

Choose a blueprint or create a custom function.

Configure the function with a name, runtime (Python 3.8 or later is recommended), and execution role.

Write the Lambda Function Code:

In the "Function code" section of the Lambda function, you can write Python code to add data to the DynamoDB table. Here's a simple example using Boto3:

python Copy

code import

os import

boto3

```
# Initialize the AWS SDK
dynamodb = boto3.resource('dynamodb')

def lambda_handler(event, context):
    # Retrieve data to be added to DynamoDB from the event
data = {
        'id': 'unique_id',
        'name': 'John Doe',
        'email': 'john@example.com'
    }

    # Specify the DynamoDB table name
table_name = 'your-dynamodb-table-name'

    # Get the DynamoDB table     table =
dynamodb.Table(table_name)

    # Add the data to DynamoDB
```

```
    table.put_item(Item=data)
```

This code adds a new item to the specified DynamoDB table.

Set Up Required Permissions:

Ensure that the Lambda function's execution role has the necessary permissions to interact with DynamoDB. You can create a role with appropriate policies using AWS IAM.

Test the Function:

You can test the Lambda function by invoking it manually with sample data.

Deploy the Function:

Save and deploy your Lambda function.

Monitor and Troubleshoot:

You can monitor the function's execution and review logs in the AWS CloudWatch Logs to ensure it's working as expected.

Additional Configuration:

Depending on your use case, you can extend the Lambda function to add more complex data to DynamoDB or trigger it using AWS services like S3, API Gateway, or other event sources.

Make sure to replace the example code with your specific DynamoDB table name and data. This Lambda function will add the specified data to the DynamoDB table whenever it's invoked.

Conclusion :

In the following we learnt how to use lambda functions to create mongodb table.

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/controllerv1.0.0/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 full-duplex 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.

WRITTEN ASSIGNMENT 2

Q1. How to deploy lambda function on AWS?

Deploying a Lambda function on AWS involves several steps. Lambda is a serverless compute service that lets you run code without provisioning or managing servers. Here's a step-by-step guide on how to deploy a Lambda function:

Step 1: Create a Lambda Function

1. Log in to the AWS Management Console.
2. Navigate to the Lambda service by searching for it in the AWS Services section.
3. Click the "Create function" button.

Step 2: Configure Your Function

4. Choose "Author from scratch."
5. Fill in the basic information for your function, including the name, runtime, and execution role.
6. For "Execution role," you can create a new role from a template or use an existing role if you have one with the necessary permissions.
7. Click the "Create function" button.

Step 3: Upload Your Code

8. In the "Function code" section, you can upload your code either directly (ZIP file or folder) or by specifying a repository from AWS CodeCommit, Amazon S3, or other options.
9. Configure the handler if needed. The handler is the method that AWS Lambda invokes.

10. Set environment variables if your code requires them.
11. Adjust the runtime settings if necessary.
12. Click the "Deploy" button to upload your code.

Step 4: Define the Trigger

13. In the "Add triggers" section, you can specify event sources that will trigger your Lambda function. This can be an API Gateway, an S3 bucket, an SNS topic, etc.
14. Configure the trigger according to your needs and grant permissions as required.

Step 5: Test Your Function

15. You can test your Lambda function by creating a test event or using a sample test event provided by AWS.
16. Click the "Test" button to run your function and see the results.

Step 6: Monitor and Troubleshoot (Optional)

17. AWS Lambda provides various monitoring and logging options. You can configure CloudWatch alarms, inspect logs, and set up notifications to keep an eye on your function's performance.

Step 7: Save and Deploy the Function

18. After you've configured everything to your satisfaction, click the "Save" button to save your changes.
19. Click the "Deploy" button to make your function live and ready to respond to triggers.

Step 8: Invocation and Scaling

Your Lambda function is now deployed and ready to be invoked by the trigger you configured. AWS Lambda handles the scaling of resources based on the incoming workload automatically.

Q2. What are the deployment options for AWS Lambda?

AWS Lambda offers several deployment options:

1. **Code Upload:** You can directly upload your function code as a ZIP file or a deployment package when creating or updating your Lambda function.

2. **AWS Lambda Layers:** You can use Lambda Layers to separate your function code from its dependencies. This allows you to manage and version common libraries independently and share them across multiple functions.

3. **AWS SAM (Serverless Application Model):** AWS SAM is a framework for building serverless applications. You can define your Lambda functions and their associated resources in a SAM template file, then deploy the entire application using AWS SAM CLI.

4. **AWS CloudFormation:** You can use AWS CloudFormation templates to define and deploy Lambda functions along with other AWS resources. This enables infrastructure as code (IaC) for your serverless applications.

5. **AWS Serverless Application Repository:** You can publish and share serverless applications and Lambda functions using the AWS Serverless Application Repository. Users can deploy your applications directly from the repository.

6. **AWS CodePipeline:** You can integrate AWS Lambda deployment into a continuous integration/continuous deployment (CI/CD) pipeline using AWS CodePipeline. This automates the building, testing, and deployment of your Lambda functions.

7. **Container Image Support**: AWS Lambda now supports deploying functions as container images, allowing you to package your function and dependencies in a Docker container and deploy them as a Lambda function.

These deployment options provide flexibility and enable you to choose the one that best fits your application architecture and development workflow.

Q3 What are the 3 full deployment modes that can be used for AWS?

In the context of AWS Lambda, there are three primary deployment modes:

1. **Automatic Deployment:** AWS Lambda provides automatic deployment when you directly upload your function code as a ZIP

file or specify a deployment package. This is the most common and straightforward deployment method, and it's suitable for most use cases.

2. Infrastructure as Code (IaC) with CloudFormation: AWS CloudFormation is a service that allows you to define and provision AWS infrastructure and resources in a template. You can use CloudFormation to define your Lambda functions, their associated resources, and any other AWS resources your application needs. When you create or update the CloudFormation stack, it deploys the Lambda functions and other resources defined in the template. This approach is more suitable for complex serverless applications and infrastructure orchestration.

3. **Serverless Application Model (AWS SAM):** AWS SAM is an opensource framework for building serverless applications. It extends AWS CloudFormation to provide a simplified way to define serverless resources, including Lambda functions. You can use AWS SAM to define your functions and related resources in a SAM template, and then deploy the entire application using AWS SAM CLI. This approach is a balance between the simplicity of direct deployment and the power of CloudFormation for more complex applications.

These deployment modes offer different levels of control and abstraction, allowing you to choose the one that best matches your deployment needs and development workflow.

Q4. What are the 3 components of AWS Lambda?

AWS Lambda consists of three primary components:

1. **Function Code:** This is the core component of AWS Lambda. It's the code that you want to run in response to events. You can write your code in various supported programming languages (e.g., Node.js, Python, Java, C#, etc.). You can package your code along with its dependencies into a deployment package, which you upload to Lambda.

2. **Event Sources:** Event sources are triggers that invoke your Lambda function. AWS Lambda supports various event sources, such as Amazon S3, Amazon DynamoDB, Amazon SNS, AWS API Gateway, and more. When an event occurs in one of these services, it triggers the execution of your Lambda function.

3. **Execution Environment:** AWS Lambda manages the execution environment for your function. It automatically scales and provisions the necessary infrastructure to run your code. You don't need to worry about server provisioning or resource management. AWS Lambda also provides runtime support for various programming languages, so your code runs in an isolated environment with the necessary resources to execute.